Advancing Natural Language Processing in Political Science

Dissertation von Sandra Wankmüller

München 2022

Advancing Natural Language Processing in Political Science

Inaugural-Dissertation
zur Erlangung des Doktorgrades an der Sozialwissenschaftlichen Fakultät
der Ludwig-Maximilians-Universität München

vorgelegt von
Sandra Wankmüller

2022

Erstgutachter:	Prof. Dr. Paul W. Thurner
Zweitgutachter:	Prof. Dr. Christian Heumann
Tag der mündlichen Prüfung:	2.12.2022

I dedicate this thesis to my love, Moritz.

Acknowledgements

Five years ago, in a Graduate Center workshop for new doctoral students, the instructor said: "Writing a dissertation is like running a marathon." "Very good.", I thought, "I am (or rather used to be) fairly good at long-distance running. In fact, in everything I do, I prefer the slow, steady pace of endurance runs compared to the fast racing in sprints. So, I should be fine with that." Yet imagining running a marathon is not the same as really running a marathon. Doing is much harder than imagining. And what I did not realize then was that this marathon (just like probably any marathon) is not one you do alone. You run it alone, but you do not accomplish it alone. And so, on June 11th, 2022, at 06:14 in the morning, happy and exhausted and the finishing line for this marathon just in sight, I thank all those that played an important role along the way.

I am grateful to Prof. Dr. Paul W. Thurner for setting me on track, for giving helpful advice, for letting me explore numerous paths and sideways, for providing guidance, for the kindness and generosity I have experienced, and for safely navigating me *all* this way (including the difficult parts). Paul, I thank you from the bottom of my heart!

I thank Prof. Dr. Christian Heumann for taking me on, for his valuable feedback and input, for sharing his ideas, and for the helpful answers to my numerous questions. I learned something new about statistics (or life in general) in every conversation. I am truly grateful to you!

I thank Prof. Dr. Mario Haim for readily and happily agreeing to take on the role of the third examiner in the disputation.

I thank (in chronological order) Prof. Dr. Susumu Shikano, Prof. Dr. Martin Elff, Prof. Dr. Simon Munzert, apl. Prof. Dr. Michael Hermann, Prof. Dr. Philip Leifeld, Dr. Michael Stoffel, Prof. Dr. Paul W. Thurner, Dr. Thomas Däubler, Prof. Dr. Christian Heumann, Dr. James Lo, Assoc. Prof. Dr. Pablo Barberá, Prof. Dr. Peer Kröger, and Dr. Matthias Aßenmacher for giving excellent courses on methods, statistics, machine learning, quantitative text analysis, and natural language processing,—and thus teaching me the core skills I required for this marathon.

I am grateful to the *Studienstiftung des deutschen Volkes* for important financial support and encouraging events. I also thank the Network for the Advancement of Social and Political Studies for the scholarship for EITM Europe 2018 and the wonderful, inspiring time in Turin.

I am grateful to Ingrid for providing orientation and guidance at the start of the marathon, and to Simon for doing so at the end. I also thank Manuela Stetter for patiently answering

numerous questions on the formalities of this process.

I am grateful to several groups of people for their valuable comments, questions, and thoughts on my work: the participants of the colloquium of Prof. Dr. Paul W. Thurner, especially Paul himself, Oliver, Lukas, Jérôme, Marius, Simon, and Andreas; conference participants (particularly at DAGStat 2019); several anonymous reviewers; Christian, Peer, Matthias, Thomas, and Markus. I also thank Andreas for first letting me share an office and then continuing the exchange during the COVID lockdowns and thereafter.

I thank my friends, Elisa, Ines, Isi, Anna, Berit, Olivia, and Miss B. for being there, for listening, for asking thought-provoking questions, and for knowing and feeling what I need.

I thank my parents for supporting each marathon I decide to run, for their love and dedication, and for providing a refuge I can always come back to.

And I thank Moritz for being at my side every step of this marathon. I thank you for helping me to stand up after every time I fell down, for holding me tight on tough days, and for the patient, affectionate everyday-support. I can't imagine having come this far without you. You are an essential source of joy, happiness, calmness, strength, and love to me.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	Natural Language Processing
CBMM	*C*lassifier-Based *B*eta *M*ixed *M*odeling
AI	Artificial Intelligence
US	United States
UK	United Kingdom
EU	European Union
IRT	Item Response Theory
LDA	Latent Dirichlet Allocation
CTM	Correlated Topic Model
STM	Structural Topic Model
NTM	Neural Topic Model
POS	Part-of-Speech
NER	Named Entity Recognition
SVM	Support Vector Machine
RNN	Recurrent Neural Network
FNN	Feedforward Neural Network
ReLU	Rectified Linear Unit
GELU	Gaussian Error Linear Unit
CNN	Convolutional Neural Network
BPTT	Backpropagation Through Time
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
RecNNs	Recursive Neural Networks
NMT	Neural Machine Translation
GloVe	Global Vectors
CBOW	Continuous Bag-of-Words
GLM	Generalized Linear Model
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte-Pair Encoding
GPU	Graphics Processing Unit
NLI	Natural Language Inference

ELU	Exponential Linear Unit
CPU	Central Processing Unit
ZSL	Zero-Shot Learning
COPDAB	Conflict and Peace Data Bank
RBF	Radial Basis Function

Notation

If not specified otherwise, the following notation is used for core concepts in this dissertation.

y	scalar
$\boldsymbol{y}$	vector
$\boldsymbol{Y}$	matrix or higher-dimensional tensor
$\hat{y}$	estimate of y
$\boldsymbol{Y}^\top$	transpose of $\boldsymbol{Y}$
$\{a, b, c\}$	discrete, unordered set containing elements a, b, c
(a, b, c)	sequence, tuple (ordered list of elements a, b, c)
$[a, b, c]$	three-dimensional vector containing elements a, b, c
$\mathbb{R}$	the set of real numbers
$f(\boldsymbol{x}, \boldsymbol{\theta})$	function of $\boldsymbol{x}$ that is parameterized by the set of parameters $\boldsymbol{\theta}$. Note that the symbol f will be used to denote any function irrespective of whether the function takes as an input and produces as an output a scalar, vector, matrix, or tensor.
$\frac{\partial y}{\partial x}$	partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$	gradient of $f(\boldsymbol{x})$ with respect to $\boldsymbol{x}$

$D = (d_1, \ldots, d_i, \ldots, d_N)$	a collection of N observational units. In the context of NLP, d_i is typically called a document and D is called a corpus.
$\boldsymbol{y} = [y_1, \ldots, y_i, \ldots, y_N]^\top$	vector of output values for N training instances
$\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iu}, \ldots, x_{iU}]$	vector that represents instance d_i across U features
$\boldsymbol{X} = [\boldsymbol{x}_1 \vert \ldots \vert \boldsymbol{x}_i \vert \ldots \vert \boldsymbol{x}_N]^\top$	$N \times U$ matrix that represents each unit d_i as a row vector $\boldsymbol{x}_i$ across U features. If the d_i are documents, $\boldsymbol{X}$ is typically called a document-feature matrix.
$\mathsf{D}_{train} = (\boldsymbol{x}_i, y_i)_{i=1}^N$	training data set
$p(\boldsymbol{x}, y)$	unknown joint distribution over data representation input $\boldsymbol{x}$ and output y
f	true systematic relation between inputs and outputs for data points drawn from $p(\boldsymbol{x}, y)$
$\tilde{f}$	candidate function $\tilde{f}$. $\tilde{f}$ is an element of the hypothesis space $\mathcal{H}$ that a given machine learning algorithm can learn.
$\hat{f}$	model; result of the training process; function that a machine learning algorithm has learned on the basis of the training data with the aim to approximate f
$\mathcal{L}(y_i, \tilde{f}(\boldsymbol{x}_i))$	loss function that captures the discrepancy between the true value y_i and the value predicted by $\tilde{f}(\boldsymbol{x}_i)$
$\mathsf{D}_{test} = (\boldsymbol{x}_m^*, y_m^*)_{m=1}^M$	test data set
$\mathsf{f}(\boldsymbol{\theta})$	parametric machine learning method that is characterized by a set of parameters $\boldsymbol{\theta}$. The notation $\mathsf{f}(\boldsymbol{\theta})$ is used if it is to be particularly emphasized that the text refers to a parametric machine learning method.
$\tilde{\boldsymbol{\theta}}$; $\hat{\boldsymbol{\theta}}$	candidate parameter values; estimated parameter values

$d_i = (a_1, \ldots, a_t, \ldots, a_{T_i})$	ith document which is an ordered sequence of T_i tokens
$Z = \{z_1, \ldots, z_u, \ldots, z_U\}$	vocabulary Z consisting of U unique features. Each token a_t is an instance of one of the unique features in the vocabulary.
$\boldsymbol{z}_u \in \mathbb{R}^K$	K-dimensional real-valued vector representation of feature z_u
$\boldsymbol{z}_{[a_t]} \in \mathbb{R}^K$	K-dimensional real-valued vector representation of the feature corresponding to token a_t
$\boldsymbol{h}_l \in \mathbb{R}^{K_l}$	K_l-dimensional real-valued vector representation in the lth hidden layer
$\boldsymbol{W}_l \in \mathbb{R}^{K_l \times K_{l-1}}$	$K_l \times K_{l-1}$ weight matrix in the lth hidden layer
$\boldsymbol{b}_l \in \mathbb{R}^{K_l}$	K_l-dimensional bias vector in the lth hidden layer
σ	nonlinear activation function
$\mathcal{X}$	feature space
$\mathcal{Y}$	label space
$\mathcal{D}$	domain
$\mathcal{T}$	task

Summary

Text data emerge as natural (by-)products of political life. They allow political scientists to unobtrusively observe political actors and also provide rich information on political processes, institutions, occurrences, and concepts. Hence, text data are a highly valuable data source for political scientists.

Consequently, the analysis of text data is extensively practiced in political science. Yet the applied text analysis methods are not always the most suitable or the most effective for the given research objectives.

- Most prominently, fundamental methods and developments from the fields of machine learning and natural language processing (NLP) (especially deep neural networks and transfer learning) are not frequently applied and arrive only slowly in political science.
- When it comes to identifying relevant documents from larger corpora, political scientists tend to use keyword lists. Keyword lists, however, have a relatively high risk of producing (substantial) selection biases.
- For the text-based measurement of continuous concepts, researchers in political science—but also in NLP itself—use methods that require comprehensive, context-specific information or (probably unattainably) large amounts of resources.

This dissertation introduces, presents, compares, and evaluates methods that address these issues—and thereby contributes to advancing the quantitative study of text data in political science. The dissertation consists of an introductory chapter and three articles. The introductory chapter provides a review of core concepts as well as advanced methods from the fields of machine learning and NLP. It also motivates each of the articles and embeds each article in its research context. Furthermore, the introductory chapter discusses one core limitation of this dissertation's articles— which also is a limitation of NLP research in general. In the following, the contexts and contributions of the three articles are outlined. Moreover, the discussion on limitations is summarized.

Introduction to Neural Transfer Learning with Transformers for Social Science Text Analysis

When analyzing texts, political scientists frequently apply methods from machine learning and NLP. Political scientists employ supervised learning techniques on text data to measure concepts that are already clearly conceptualized and defined (see Sections 1.1.1.6 and 1.2.4). Unsupervised learning (especially topic models and ideal point estimation techniques) is frequently put to use in political science to detect latent structures from texts and to organize units along these detected structures (see Section 1.1.2).

In this dissertation's article *Introduction to neural transfer learning with Transformers for social science text analysis* (Chapter 2) the focus is on situations in which a researcher wishes to measure a concept from text, where the concept has already been conceptualized and there is a training data set that encodes the operationalization of the concept. Hence, the focus is on supervised learning.

If a supervised learning algorithm is applied for the measurement of concepts from texts, the learning algorithm can be regarded as a measurement instrument that assigns values to the units under study. One central quality criterion of a measurement instrument is its validity. Humans are regarded to be the best available tools for understanding and interpreting the content of text (Benoit 2020, p. 470). Thus, political scientists that employ supervised learning to measure concepts from texts should seek to have a model that as accurately as possible imitates human codings (Grimmer & Stewart, 2013, p. 270, 279).

In practice, political scientists typically combine bag-of-word-based representations with conventional supervised machine learning algorithms (see Section 1.2.4). This approach is not fundamentally wrong and can lead to acceptable prediction performances, but the field of NLP applies more complex and advanced methods that are likely to yield more accurate predictions—and thus more valid measures. Specifically, political science research practice so far has not yet widely taken up methods that have been developed or have been ubiquitously applied in the field of NLP during the last one and a half decades (most prominently deep neural networks, transfer learning, and the Transformer architecture).

In contrast to conventional algorithms, deep neural networks not only learn a mapping between data representations and outputs but also learn representations of data inputs (Goodfellow et al., 2016, p. 5; Sections 1.2.3.1 and 2.2.2). Especially as deep neural networks can learn multi-layered (i.e. deep) and contextualized representations of textual inputs, they are often likely to be more suitable for the processing of text. The Transformer is a deep neural network architecture developed by Vaswani et al. (2017). The Transformer is composed of (self-)attention mechanisms that enable the Transformer to learn contextualized token representations that can capture information from other tokens in a sequence regardless of the distance between the tokens (Vaswani et al., 2017, p. 5998; Sections 1.2.3.7 and 2.4). Transfer learning is a mode of learning in which information that has been acquired by training on a source task in a source domain is utilized for the train-

ing process on the target task in the target domain (Pan & Yang, 2010, p. 1347; Ruder, 2019, p. 42-44; Sections 1.2.3.8 and 2.3). Transfer learning tends to enhance efficiency and performance (Howard & Ruder, 2018; Ruder, 2019).

Pointing out the gap between political science research and modern NLP techniques, the article *Introduction to neural transfer learning with Transformers for social science text analysis* provides an introduction to deep learning for text data, explains the advantages of deep contextualized representations, presents the Transformer, and introduces transfer learning. The article then compares Transformer-based models for transfer learning with conventional machine learning algorithms on the basis of three supervised learning tasks. The results suggest that Transformer-based models that are used in a transfer learning setting are better than conventional methods in learning the operationalization encoded in the training data and hence yield more accurate measures of concepts from texts (see Section 2.6.6).

A Comparison of Approaches for Imbalanced Classification Problems in the Context of Retrieving Relevant Documents for an Analysis

The article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* (Chapter 3) focuses on one of the first steps of many text-based analyses: the retrieval of the population of documents that are relevant for an analysis from a larger corpus of otherwise irrelevant documents. One example in this regard would be a research project that seeks to estimate the attitudes that have been expressed toward a political candidate in tweets during a given time period. One of the first steps of such a study would be to identify those tweets that refer to the political candidate from the larger collection of tweets that have been posted during the given period.

The separation of relevant from irrelevant documents is often an imbalanced classification problem (Manning et al., 2008, p. 155). Any method that is applied to the identification of relevant documents can induce a selection bias, meaning that the question of whether a document is predicted to be relevant or not is correlated with the value of the document on a variable of interest.

So far, political scientists typically employ a list of usually human-created keywords to address the task of separating relevant from irrelevant documents (see Table 3.2). But keyword lists that are created by humans tend to be highly variable and incomplete (King et al., 2017, p. 973-975). The usage of human-generated keyword lists thus bears a relatively high risk of bringing forth a (large) selection bias.

Addressing this issue, the article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* describes, compares, and evaluates methods that have the potential to more accurately identify relevant documents. A higher prediction performance does not preclude the occurrence of a se-

lection bias. Yet the higher the share of relevant documents that a method can identify among the set of all relevant documents (i.e. the higher recall), the smaller the size that the induced selection bias can maximally have (see Figure 1.2). The evaluated methods are (1) query expansion techniques, (2) topic model-based classification rules, and (3 & 4) active as well as passive supervised learning. The results suggest that active learning techniques can separate relevant from irrelevant documents much more accurately than keyword lists and the other evaluated approaches (see Section 3.4.3). Therefore, active learning constitutes a learning strategy that political scientists can use to decrease the potential magnitude of the selection bias that can result from imperfectly identifying relevant documents.

How to Estimate Continuous Sentiments from Texts Using Binary Training Data

The starting point for this dissertation's article *How to estimate continuous sentiments from texts using binary training data* (Chapter 4) is the conceptualization of attitudes. In social psychology, an attitude is conceptualized as a latent continuous summarizing evaluation of an entity (Eagly & Chaiken, 1993, p. 1; Cacioppo et al., 1997, p. 10, 13). The psychological definition of an attitude very well encapsulates the descriptions in the NLP literature on the concept of sentiment (Liu, 2015, p. 2). Therefore, sentiments here are conceived of as attitudes.

In contrast to analyses that merely assess the tone expressed in a document, NLP tasks such as targeted sentiment analysis or stance detection capture important aspects that are implied by the definition of attitudes. However, the conceptualization of an attitude as a continuous concept is often ignored in practice: Sentiment analysis typically is conducted as a classification task in which documents are assigned to sentiment categories (e.g. Pang et al., 2002; Turney, 2002; Nakov et al., 2013; Socher et al., 2013; Pontiki et al., 2015; Patwa et al., 2020). Moreover, in those cases in which a correspondence between the conceptualization and the measurement of attitudes exists and attitudes are measured as continuous concepts, lexicon-based approaches or regression approaches are applied (Gatti et al., 2016; Mohammad et al., 2018). Lexicon-based and regression approaches, however, either rely on complete and accurate information about context-specific meanings and the compositionality of meaning in text, or they rely on highly detailed—and thus potentially prohibitively costly to create—training information.

Against this background, the article *How to estimate continuous sentiments from texts using binary training data* develops a method (named *c*lassifier-based *b*eta *m*ixed *m*odeling (CBMM)) that allows measuring attitudes from texts in a way that corresponds with the conceptualization of attitudes and merely requires binary labels for the training data.

CBMM is a three-step procedure. First, binary training data have to be created. For each training document, the assigned label should signify whether the document is closer to the positive or the negative side of the latent continuous attitude variable. In the second step,

an ensemble of classifiers is trained on the binary training data. Each classifier then is applied on test data documents to produce for each test document a predicted probability to belong to the positive category. Finally, a beta mixed model with document random intercepts and classifier random intercepts is applied to the set of predicted probabilities that the classifiers have generated for each document. The estimated document-specific intercepts are taken to be the continuous attitude estimates. An analysis based on four datasets (see Section 4.4.4) indicates that CBMM's continuous estimates are relatively close to true sentiment values and have a similar or only slightly lower performance than continuous sentiment estimates that result from regression approaches that can use fine-grained training data.

CBMM thus allows measuring attitudes based on text data in correspondence with the conceptualization of attitudes without requiring highly detailed information or highly fine-grained training annotations. CBMM's core contribution is that it generates continuous estimates based on binary training labels. The CBMM procedure can be applied beyond sentiment analysis whenever it is desirable to measure a continuous concept from text with mere binary training input.

Limitations and Further Research

Section 1.4 of the introductory chapter discusses limitations of this dissertation's articles and of NLP research in general. The central point made in this section is that *NLP as a science* seeks to make inferences about the performance effects that result from applying one method (compared to another method) in the processing of natural language. Yet NLP research in practice usually does not achieve this goal: In NLP research articles, typically only a few models are compared. Each model results from a specific procedural pipeline (here named processing system) that is composed of a specific collection of methods that are used in preprocessing, pretraining, hyperparameter tuning, and training on the target task. To make generalizing inferences about the performance effect that is caused by applying some method A vs. another method B, it is not sufficient to compare a few specific models that are produced by a few specific (probably incomparable) processing systems. Rather, the following procedure would allow drawing inferences about methods' performance effects:

- A population of processing systems that researchers seek to infer to has to be defined.
- A random sample of processing systems from this population is drawn. (The drawn processing systems in the sample will vary with regard to the methods they apply along their procedural pipelines and also will vary regarding the compositions of their training and test data sets used for training and evaluation.)
- Each processing system is applied once with method A and once with method B.
- Based on the sample of applied processing systems, the expected generalization errors

of method A and method B are approximated.

- The difference between the expected generalization errors of method A and method B is the estimated average treatment effect due to applying method A compared to method B in the population of processing systems.

The reason why this described procedure (or an equally adequate procedure) is not applied in practice is that research resources are finite. Even the implementation of a single processing system that then will produce a single model can absorb large amounts of resources. Due to these practical limitations, the articles in this dissertation implement research procedures that are closer to typical NLP practices rather than the ideal procedure outlined here. Nevertheless, the available amounts of resources in this dissertation could have been allocated better—for example by not implementing resampling techniques during hyperparameter tuning and evaluating each processing system on more train-test set compositions instead.

Section 1.4, furthermore, emphasizes that future research in political science is likely to benefit from (1) continuously monitoring and exploring developments in the field of NLP, as well as (2) devising and applying methods to handle prediction errors that arise when measuring concepts from texts.

Zusammenfassung

gemäß §9 Abs. 3 Satz 3 der Promotionsordnung der Sozialwissenschaftlichen Fakultät der Ludwig-Maximilians-Universität vom 18. März 2016

Textdaten entstehen als natürliche (Neben-)Produkte des politischen Lebens. Sie ermöglichen eine nicht-teilnehmende Beobachtung politischer Akteure und liefern reichhaltige Informationen zu politischen Prozessen, Institutionen, Ereignissen und Konzepten. Daher sind Textdaten für Politikwissenschaftler*innen eine äußerst wertvolle Datenquelle.

Folglich wird die Analyse von Textdaten in der Politikwissenschaft ausgiebig praktiziert. Allerdings sind die hierbei angewandten Textanalysemethoden nicht immer die am geeignetsten oder effektivsten für das jeweilige Forschungsziel.

- So werden in der Politikwissenschaft wichtige Entwicklungen und Methoden aus den Bereichen *machine learning* und *natural language processing* (NLP) (insbesondere *deep learning* und *transfer learning*) bisher kaum oder nur langsam übernommen.
- Wenn es um die Identifikation relevanter Dokumente aus größeren Korpora geht, tendieren Politikwissenschaftler*innen dazu, Suchwortlisten zu verwenden. Suchwortlisten sind einfache und kostengünstige Methoden, die jedoch ein relativ hohes Risiko haben, einen (substanziellen) *selection bias* zu generieren.
- Zur textbasierten Messung von kontinuierlichen Konzepten werden in der Politikwissenschaft – aber auch im Feld NLP selbst – Methoden eingesetzt, die umfassende, kontextspezifische Informationen oder eine (möglicherweise nicht aufbringbar) große Mengen an Ressourcen benötigen.

Die vorliegende Dissertation präsentiert, vergleicht, und evaluiert Methoden, die diese Aspekte adressieren, und leistet somit einen Beitrag zur Weiterentwicklung textbasierter politikwissenschaftlicher Forschungspraxis. Die Dissertation besteht aus einem einführenden Kapitel und drei Artikeln. Das Einführungskapitel gibt einen Überblick über zentrale Konzepte sowie fortgeschrittene Methoden aus den Bereichen *machine learning* und NLP. Das einführende Kapitel motiviert zudem jeden der Artikel und bettet jeden Artikel in seinen Forschungskontext ein. Darüber hinaus wird eine zentrale Limitation aller Artikel dieser Dissertation diskutiert (die eine Limitation der NLP-Forschung im Allgemeinen ist). Im Folgenden werden die Kontexte und Forschungsbeiträge der drei Artikel skizziert. Außerdem wird die Diskussion zu den Limitationen zusammengefasst.

Introduction to Neural Transfer Learning with Transformers for Social Science Text Analysis

Bei der Analyse von Texten wenden Politikwissenschaftler*innen häufig Methoden aus den Bereichen *machine learning* und NLP an. Techniken des *supervised learning* werden eingesetzt, um Konzepte zu messen, die bereits klar konzeptualisiert und definiert sind (siehe Abschnitte 1.1.1.6 und 1.2.4). *Unsupervised learning* (insbesondere *topic models* und *ideal point estimation*) wird genutzt, um latente Strukturen in Texten zu entdecken und Untersuchungseinheiten entlang dieser Strukturen zu organisieren (siehe Abschnitt 1.1.2).

Im Artikel *Introduction to neural transfer learning with Transformers for social science text analysis* (Kapitel 2) stehen Situationen im Fokus, in denen ein*e Forscher*in ein Konzept anhand von Texten messen möchte – wobei das Konzept bereits konzeptualisiert wurde und zudem auch ein Trainingsdatensatz vorliegt, der die Operationalisierung des Konzeptes kodiert. Der Fokus liegt also auf *supervised learning*.

Wird ein *supervised learning*-Algorithmus zur Messung von Konzepten aus Texten eingesetzt, so kann der Algorithmus als ein Messinstrument betrachtet werden, das den Untersuchungseinheiten Werte zuweist. Ein zentrales Qualitätskriterium eines jeden Messinstrumentes ist die Validität. Da Menschen als das 'beste verfügbare Werkzeug' gelten, um Textinhalte zu verstehen und zu interpretieren (Benoit, 2020, p. 470), sollten Politikwissenschaftler*innen, die *supervised learning* zur Messung von Konzepten aus Texten einsetzen, anstreben, ein Modell zu trainieren, das menschliche Kodierungen so genau wie möglich imitiert (Grimmer & Stewart, 2013, p. 270, 279).

In der Regel repräsentieren Politikwissenschaftler*innen Dokumente als *bag-of-words* und kombinieren diese Repräsentationsform mit konventionellen Lernalgorithmen (siehe Abschnitt 1.2.4). Dieser Ansatz ist nicht grundsätzlich falsch und kann zu einem akzeptablen Niveau an Vorhersagegenauigkeit führen. NLP-Forscher*innen wenden jedoch komplexere und fortschrittlichere Methoden an, die üblicherweise auch zu einer höheren Vorhersagegenauigkeit – und damit zu valideren Messungen – führen. Insbesondere werden in der politikwissenschaftlichen Forschungspraxis bisher noch nicht in größerem Umfang NLP-Methoden angewandt, die in den letzten eineinhalb Jahrzehnten entwickelt wurden und dort allgegenwärtig sind (vor allem: das Lernen mit tiefen neuronalen Netzen, *transfer learning* und die *Transformer*-Architektur).

Im Gegensatz zu konventionellen Algorithmen lernen tiefe neuronale Netze nicht nur eine Funktion, die den Zusammenhang zwischen Datenrepräsentationen (Inputs) und Outputs abbildet, sondern lernen auch Repräsentationen von Daten-Inputs (Goodfellow et al., 2016, S. 5; Abschnitte 1.2.3.1 und 2.2.2). Insbesondere da tiefe neuronale Netze mehrschichtige (d. h. tiefe) und kontextualisierte Repräsentationen von textuellen Inputs lernen können, sind sie häufig besser für die Verarbeitung von Textdaten geeignet als *bag-of-words*-Repräsentationen in Kombination mit konventionellen Lernalgorithmen. Der

Transformer ist eine tiefes neuronales Netz, das von Vaswani et al. (2017) entwickelt wurde. Der *Transformer* basiert auf *(self-)attention* Mechanismen, die es ermöglichen, kontextualisierte Repräsentationen für textuelle *token* zu lernen, die Informationen von anderen *token* in einer Textsequenz erfassen können – und zwar unabhängig von der Entfernung zwischen den *token* (Vaswani et al., 2017, p. 5998; Abschnitte 1.2.3.7 und 2.4). *Transfer learning* ist eine Lernform, bei der Informationen, die durch das Training auf einer Quellaufgabe in einer Quelldomäne erworben wurden, für den Trainingsprozess der eigentlich interessierenden Zielaufgabe in der Zieldomäne genutzt werden (Pan & Yang, 2010, p. 1347; Ruder, 2019, p. 42-44; Abschnitte 1.2.3.8 und 2.3). *Transfer learning* führt tendenziell zu einer Steigerung der Effizienz und Vorhersagegenauigkeit (Howard & Ruder, 2018; Ruder, 2019).

Der Artikel *Introduction to neural transfer learning with Transformers for social science text analysis* weist auf die Diskrepanz zwischen politikwissenschaftlicher Forschung und modernen NLP-Verfahren hin, bietet eine Einführung in das *deep learning*, erläutert die Vorteile tiefer kontextualisierter Repräsentationen, stellt den *Transformer* vor und führt in das *transfer learning* ein. Anschließend vergleicht der Artikel *Transformer*-basierte Modelle, die in einem *transfer learning*-Kontext trainiert werden, mit konventionellen Algorithmen anhand von drei Lernaufgaben. Die Ergebnisse deuten darauf hin, dass *Transformer*-basierte Modelle die in den Trainingsdaten kodierte Operationalisierung von Konzepten besser erlernen und somit genauere Messungen für Konzepte aus Texten liefern können als konventionelle Methoden (siehe Abschnitt 2.6.6).

A Comparison of Approaches for Imbalanced Classification Problems in the Context of Retrieving Relevant Documents for an Analysis

Der Artikel *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* (Kapitel 3) nimmt einen der ersten Schritte vieler textbasierter Analysen in den Blick: das Identifizieren der für eine Analyse relevanten Dokumentenpopulation aus einem größeren Korpus von ansonsten irrelevanten Dokumenten. Ein Beispiel hierfür ist ein Forschungsprojekt, in dem es darum geht, die während eines bestimmten Zeitraumes in Tweets zum Ausdruck gebrachten Einstellungen gegenüber einem politischen Kandidaten zu messen. Einer der ersten analytischen Schritte in einer solchen Studie besteht darin, die Tweets, die sich auf den politischen Kandidaten beziehen, aus der großen Anzahl an Tweets zu identifizieren, die in dem gegebenen Zeitraum gepostet wurden.

Die Trennung von relevanten und irrelevanten Dokumenten ist häufig ein *imbalanced classification problem* (Manning et al., 2008, p. 155). Jede Methode, die für die Identifikation relevanter Dokumente eingesetzt wird, kann potenziell einen *selection bias* hervorrufen. Dies bedeutet, dass die Frage, ob eine Methode ein Dokument als relevant oder nicht relevant ansieht, mit dem Wert des Dokumentes auf einer interessierenden Variablen kor-

reliert.

Bislang verwenden Politikwissenschaftler*innen in der Regel oftmals von Menschen erstellte Listen an Suchwörtern um relevante von irrelevanten Dokumenten zu trennen (siehe Tabelle 3.2). Von Menschen erstellte Suchwortlisten sind jedoch üblicherweise höchst variabel und unvollständig (King et al., 2017, p. 973-975). Die Verwendung menschlich erstellter Suchwortlisten birgt daher ein relativ hohes Risiko, einen (erheblichen) *selection bias* zu produzieren.

Der Artikel *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* beschreibt, vergleicht und bewertet Methoden, die das Potenzial haben, relevante Dokumente genauer zu identifizieren als Suchwortlisten. Eine höhere Vorhersagegenauigkeit schließt das Auftreten eines *selection bias* nicht aus, aber je höher der Anteil der identifizierten relevanten Dokumente an der Menge aller relevanten Dokumente ist (d. h. je höher der *recall*), desto geringer ist das Ausmaß, das der *selection bias* maximal annehmen kann (siehe Abbildung 1.2). Bei den evaluierten Methoden handelt es sich um (1) *query expansion*-Techniken aus dem Bereich des *information retrieval*, (2) auf *topic models* basierende Klassifikationsregeln, die in der Kommunikationswissenschaft entwickelt wurden, sowie (3 & 4) die *machine learning*-Verfahren des *active learning* und *passive learning*. Die Ergebnisse weisen darauf hin, dass *active learning*-Verfahren relevante von irrelevanten Dokumenten wesentlich genauer trennen können als Suchwortlisten und die anderen evaluierten Ansätze (siehe Abschnitt 3.4.3). *Active learning* stellt daher eine Lernstrategie dar, die Politikwissenschaftler*innen nutzen können, um das potenzielle Ausmaß des *selection bias* zu verringern.

How to Estimate Continuous Sentiments from Texts Using Binary Training Data

Der Ausgangspunkt für den Artikel *How to estimate continuous sentiments from texts using binary training data* (Kapitel 4) ist die Konzeptualisierung von Einstellungen (*attitudes*). In der Sozialpsychologie wird eine Einstellung als eine latente, kontinuierliche, zusammenfassende Bewertung einer Entität konzeptualisiert (Eagly & Chaiken, 1993, p. 1; Cacioppo et al., 1997, p. 10, 13). Die sozialpsychologische Definition einer Einstellung deckt sich sehr gut mit den Beschreibungen in der NLP-Literatur zum Konzept der Sentiments (Liu, 2015, p. 2). Daher werden Sentiments hier als Einstellungen aufgefasst.

Im Gegensatz zu Analysen, die lediglich den in einem Dokument ausgedrückten Tonfall bewerten, werden in der *targeted sentiment analysis* oder der *stance detection* wichtige Aspekte der Definition von Einstellungen erfasst. Die Konzeptualisierung einer Einstellung als kontinuierliches Konzept wird in der Forschungspraxis jedoch oft ignoriert: Eine Sentimentanalyse wird typischerweise als Klassifikationsaufgabe durchgeführt, bei der Dokumente in Sentimentkategorien eingeordnet werden (z. B. Pang et al., 2002; Turney, 2002; Nakov et al., 2013; Socher et al., 2013; Pontiki et al., 2015; Patwa et al., 2020). Darüber hin-

aus werden in den Fällen, in denen eine Korrespondenz zwischen der Konzeptualisierung und der Messung von Einstellungen besteht und Einstellungen als kontinuierliche Konzepte gemessen werden, lexikonbasierte Ansätze oder Regressionsansätze verwendet (Gatti et al., 2016; Mohammad et al., 2018). Diese Ansätze benötigen jedoch entweder umfassende Informationen über kontextspezifische Wortbedeutungen und die Entstehung von Bedeutungsinhalten in Texten, oder sie benötigen sehr detaillierte Trainingsinformationen, deren Erstellung äußerst kostenintensiv ist.

Vor diesem Hintergrund wird in dem Artikel *How to estimate continuous sentiments from texts using binary training data* eine Methode (genannt *c*lassifier-based *b*eta *m*ixed *m*odeling (CBMM)) entwickelt, die es ermöglicht, Einstellungen aus Texten in einer Weise zu messen, die der Konzeptualisierung von Einstellungen entspricht und gleichzeitig lediglich binäre Label für die Trainingsdaten benötigt. CBMM ist ein dreistufiges Verfahren. Der erste Schritt ist die Erstellung binärer Trainingsdaten. Für jedes Trainingsdokument soll das zugewiesene Label angeben, ob das Dokument eher auf der positiven oder der negativen Seite der latenten kontinuierlichen Einstellungsvariablen liegt. Im nächsten Schritt wird ein Ensemble an Klassifikationsalgorithmen auf den binären Trainingsdaten trainiert. Jeder Klassifikator wird dann auf die Testdaten angewandt, um für jedes Testdokument eine Vorhersage über die Wahrscheinlichkeit der Zugehörigkeit zur positiven Kategorie zu erstellen. Schließlich wird ein *beta mixed model* mit *document random intercepts* und *classifier random intercepts* auf den vorhergesagten Wahrscheinlichkeiten geschätzt, die von den Klassifikatoren für jedes Dokument erzeugt wurden. Die geschätzten *document random intercepts* werden als kontinuierliche Schätzer der Einstellungswerte betrachtet. Eine Analyse auf der Grundlage von vier Datensätzen (siehe Abschnitt 4.4.4) zeigt, dass die kontinuierlichen Schätzwerte von CBMM relativ nahe an den wahren Werten liegen und eine ähnliche oder nur geringfügig niedrigere Vorhersagegenauigkeit aufweisen als kontinuierliche Schätzwerte, die mit Regressionsansätzen, die höchst detaillierte Trainingsdaten verwenden, erzeugt wurden.

CBMM ermöglicht somit die Messung von Einstellungen auf der Grundlage von Textdaten in Übereinstimmung mit der Konzeptualisierung von Einstellungen ohne höchst detaillierte Informationen oder Trainingsdaten zu benötigen. Der zentrale Forschungsbeitrag von CBMM besteht darin, kontinuierliche Schätzwerte auf der Grundlage von binären Trainingsdaten zu erzeugen. Das CBMM-Verfahren kann über die Sentimentanalyse hinaus immer dann angewendet werden, wenn ein kontinuierliches Konzept anhand von Textdaten mit rein binären Trainingsdaten gemessen werden soll.

Limitations and Further Research

In Abschnitt 1.4 des einführenden Kapitels werden die Limitationen der Artikel dieser Dissertation und der NLP-Forschung im Allgemeinen diskutiert. Der zentrale Punkt in diesem Abschnitt ist, dass NLP als Wissenschaft das Ziel hat, Aussagen über den kausalen Effekt der Anwendung einer bestimmten Methode (im Vergleich zu einer anderen Meth-

ode) auf die Vorhersagegenauigkeit bei der Verarbeitung natürlicher Sprache zu treffen. Aufgrund der NLP-Forschungspraktiken wird dies jedoch in der Regel nicht erreicht: In NLP-Forschungsartikeln werden in der Regel nur einige wenige Modelle verglichen. Jedes Modell resultiert hierbei aus einer Sequenz an methodischen Verfahren. (Eine Sequenz solcher Verfahren wird hier als Verarbeitungssystem bezeichnet.) Um verallgemeinernde Schlüsse über Effekte auf die Vorhersagegenauigkeit zu ziehen, die durch die Anwendung einer Methode A im Vergleich zu einer anderen Methode B verursacht werden, reicht es jedoch nicht aus, einige spezifische Modelle zu vergleichen, die von einigen spezifischen (möglicherweise nicht vergleichbaren) Verarbeitungssystemen erzeugt wurden. Dies könnte vielmehr mit dem folgenden Verfahren erreicht werden:

- Zunächst muss eine Population an Verarbeitungssystemen definiert werden, auf die die Forscher*innen Rückschlüsse ziehen wollen.
- Aus dieser Population wird eine Zufallsstichprobe an Verarbeitungssystemen gezogen. (Die Verarbeitungssysteme in der Stichprobe werden sich sowohl in Bezug auf die Methoden unterscheiden, die sie entlang ihrer Verfahrenssequenz anwenden, als auch in Bezug auf die Aufteilung der Daten in Trainings- und Testdatensätze.)
- Jedes Verarbeitungssystem wird einmal mit Methode A und einmal mit Methode B implementiert.
- Auf der Grundlage der Stichprobe der implementierten Verarbeitungssysteme werden die erwarteten Generalisierungsfehler von Methode A und Methode B approximiert.
- Die Differenz zwischen den erwarteten Generalisierungsfehlern von Methode A und Methode B ist der geschätzte durchschnittliche kausale Effekt durch die Anwendung von Methode A im Vergleich zu Methode B in der Population der Verarbeitungssysteme.

Der Grund, warum dieses beschriebene Verfahren (oder ein ebenso adäquates Verfahren) in der Praxis nicht angewendet wird, ist, dass Forschungsressourcen nicht unendlich sind. Bereits die Implementierung eines einzelnen Verarbeitungssystems, das dann ein einzelnes Modell erzeugt, kann große Mengen an Ressourcen benötigen. Aufgrund dieser praktischen Ressourcenbeschränkungen werden auch in den Artikeln dieser Dissertation Verfahren implementiert, die eher der typischen NLP-Praxis als dem hier skizzierten Idealverfahren entsprechen. Dennoch hätten die verfügbaren Ressourcen in dieser Dissertation besser eingesetzt werden können – zum Beispiel indem auf *resampling*-Techniken im *hyperparameter tuning* verzichtet worden wäre und stattdessen einzelnen Verarbeitungssysteme anhand einer größeren Anzahl von unterschiedlichen Zusammensetzungen der Trainings- und Testdatensätze evaluiert worden wären.

In Abschnitt 1.4 wird außerdem betont, dass die künftige politikwissenschaftliche Forschung davon profitieren wird, (1) die Entwicklungen im Bereich NLP kontinuierlich zu beobachten sowie (2) Methoden zu entwickeln und anzuwenden, um Vorhersagefehler zu verarbeiten, die bei der Messung von Konzepten aus Texten entstehen.

Bibliography

Benoit, K. (2020). Text as data: An overview. In L. Curini & R. Franzese (Eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations* (pp. 461–497). SAGE Publications. https://www.doi.org/10.4135/9781526486387.n29

Cacioppo, J. T., Gardner, W. L., & Berntson, G. G. (1997). Beyond bipolar conceptualizations and measures: The case of attitudes and evaluative space. *Personality and Social Psychology Review*, *1*(1), 3–25. https://doi.org/10.1207/s15327957pspr0101_2

Eagly, A. H. & Chaiken, S. (1993). *The psychology of attitudes*. Harcourt Brace Jovanovich College Publishers.

Gatti, L., Guerini, M., & Turchi, M. (2016). SentiWords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, *7*(4), 409–421. https://doi.org/10.1109/TAFFC.2015.2476456

Grimmer, J. & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Howard, J. & Ruder, S. (2018). Universal language model fine-tuning for text classification. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (pp. 328–339). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1031

King, G., Lam, P., & Roberts, M. E. (2017). Computer-assisted keyword and document set discovery from unstructured text. *American Journal of Political Science*, *61*(4), 971–988. https://doi.org/10.1111/ajps.12291

Liu, B. (2015). *Sentiment analysis*. Cambridge University Press. https://doi.org/10.1017/CBO9781139084789

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Mohammad, S. M., Bravo-Marquez, F., Salameh, M., & Kiritchenko, S. (2018). SemEval-2018 task 1: Affect in tweets. In M. Apidianaki, S. M. Mohammad, J. May, E. Shutova, S. Bethard, & M. Carpuat (Eds.), *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*. Association for Computational Linguistics. https://doi.org/10.18653/v1/S18-1001

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., & Wilson, T. (2013). SemEval-2013 task 2: Sentiment analysis in Twitter. In S. Manandhar & D. Yuret (Eds.), *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (pp. 312–320). Association for Computational Linguistics. https://aclanthology.org/S13-2052

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 79–86). Association for Computational Linguistics. https://doi.org/10.3115/1118693.1118704

Patwa, P., Aguilar, G., Kar, S., Pandey, S., PYKL, S., Gambäck, B., Chakraborty, T., Solorio, T., & Das, A. (2020). SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, & E. Shutova (Eds.), *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 774–790). International Committee for Computational Linguistics. https://doi.org/10.18653/v1/2020.semeval-1.100

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015). SemEval-2015 task 12: Aspect based sentiment analysis. In P. Nakov, T. Zesch, D. Cer, & D. Jurgens (Eds.), *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 486–495). Association for Computational Linguistics. https://doi.org/10.18653/v1/S15-2082

Ruder, S. (2019). *Neural transfer learning for natural language processing.* PhD thesis, National University of Ireland, Galway. https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1631–1642). Association for Computational Linguistics. https://aclanthology.org/D13-1170

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In P. Isabelle (Ed.), *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417–424). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073153

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Chapter 1

Background and Motivation: A Review of Core Concepts and Advanced Methods in Machine Learning, Natural Language Processing, and Text-Based Political Science Research

Daily, constantly, and continuously, political processes and political behavior manifest themselves in textual form.[1] Politicians—whether competing in an election or already being elected—address political issues and state their policy positions via social media, press releases, on their websites, and in speeches (Druckman et al., 2009; Grimmer, 2010; Quinn et al., 2010; Lauderdale & Herzog, 2016; Barberá et al., 2019; Fowler et al., 2021). Parties set out their goals in election manifestos (Laver et al., 2003; Slapin & Proksch, 2008). Legislators debate policies and then finally pass laws (Schwarz et al., 2017; Anastasopoulos & Bertelli, 2020; Slapin & Kirkland, 2020). Meetings of councils and committees are recorded (Sanders et al., 2018; Baerg & Lowe, 2020). Interest groups and non-governmental organizations create reports and distribute statements informing about their policy positions (Klüver, 2009; Kim, 2017; Park et al., 2020). Courts provide judicial opinions (Clark & Lauderdale, 2010). Governments record their beliefs and actions in writing, for example by formulating coalition agreements (Klüver & Bäck, 2019), producing reports assessing political situations (Bagozzi & Berliner, 2018), and keeping track of diplomatic communication (Katagiri & Min, 2019). News articles report about and comment on daily political pro-

[1] The presentation of content in some parts of this chapter formed the basis for the submitted version of the article *Introduction to neural transfer learning with Transformers for social science text analysis*. The published version of the article is Wankmüller, S. (2022). Introduction to neural transfer learning with Transformers for social science text analysis. *Sociological Methods & Research*, (p. 1–77). https://doi.org/10.1177/00491241221134527

cesses and events (Amsalem et al., 2020; Alrababa'h & Blaydes, 2021; Sebők & Kacsuk, 2021). Wikipedia pages inform about political entities (Göbel & Munzert, 2018; Hermann & Döring, 2021). Citizens communicate their own political positions and express their views on politics and policies in posts on social media platforms (Ceron et al., 2014; Mitts, 2019; Jungherr et al., 2022). Individuals also use these channels to report about political incidents (Zhang & Pan, 2019; Muchlinski et al., 2021; Wu & Mebane, 2021) and to pass on information about political action and protest (King et al., 2013; Barberá et al., 2015b).

For political scientists, that seek to make inferences about politics, text data hence are a valuable source of information. Text data are one important avenue to the processes, occurrences, actors, systems, and concepts that political scientists seek to study. As texts are often generated innately in political processes, they allow political scientists to make unobtrusive observations. In contrast to other methods of inquiry, such as surveys and experiments, in which the subjects are aware of being studied, the act of analyzing texts that have been produced as natural (by-)products of political behavior cannot influence the texts themselves.[2] Moreover, texts can contain more information and more nuanced information than other empirical measures such as voting behavior (Clark & Lauderdale, 2010; Herzog & Benoit, 2015; Schwarz et al., 2017).

During the last two decades, digitization processes caused more text data to be produced and to be recorded in readily available electronic form. In addition, new forms of political interaction and communication have emerged (that constitute new phenomena to be studied). Simultaneously, developments in the fields of statistics, computer science, and natural language processing (NLP) allow for textual data to be analyzed with increasing levels of accuracy via increasingly complex models by utilizing increasing computational capacities (LeCun & Bengio, 1995; Hochreiter & Schmidhuber, 1997; Bengio et al., 2003; Mikolov et al., 2013a; Bahdanau et al., 2015; Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020).

Political scientists have leveraged the abundance of easily available text data and have made use of respective methods for the quantitative analysis of texts to study old and new research questions (Proksch & Slapin, 2010; Rheault et al., 2016; Barberá et al., 2019; Rodman, 2020). Researchers in political science even developed their own text analysis methods (Slapin & Proksch, 2008; Roberts et al., 2016).

Yet amongst all this progress there is still room for advancement. Especially: When working with text data, political scientists currently do not tend to use the most suitable or powerful tools from the available toolbox.[3] The most noticeable issue is that political

[2]This is not to say that texts would be sincere statements revealing the true positions of political actors. In addition to an assumed underlying true position, there is a large spectrum of factors that are likely to affect a text's content, for example social norms and strategic considerations (see Section 1.1.1.5 below).

[3]The statements made in this introductory chapter refer to political science. However, the statements usually also apply to other disciplines in the field of social science, e.g. communication science and sociology. The article *Introduction to neural transfer learning with Transformers for social science text analysis* and

scientists so far do not widely apply and only slowly adopt key developments and methods from the field of machine learning and NLP (especially: deep learning, transfer learning, and more recently the Transformer architecture). And so, a 2020 book chapter on text as data methods written for *The SAGE Handbook of Research Methods in Political Science and International Relations* by Kenneth Benoit only contains a few references to methods beyond what was already mentioned in a 2013 mapping of the field of quantitative text analysis in political science by Grimmer & Stewart (compare Grimmer & Stewart 2013 to Benoit 2020).

Moreover, for document retrieval tasks, political scientists still tend to implement keyword lists. The implementation of keyword lists, however, comes with a relatively high risk of drawing biased inferences. Furthermore, when it comes to the measurement of continuous concepts from texts, in political science as well as in NLP, methods are applied that a priori require highly detailed knowledge or (prohibitively) large amounts of resources.

This doctoral thesis presents, evaluates, and also develops methods that address these issues. In doing so, this thesis contributes to the advancement of text-based political science research. The overall aim is to improve and enlarge the set of tools that political scientists use to analyze texts.

The thesis comprises three articles:

- Chapter 2: Wankmüller, S. (2022). Introduction to neural transfer learning with Transformers for social science text analysis. *Sociological Methods & Research*, (p. 1–77). https://doi.org/10.1177/00491241221134527
- Chapter 3: Wankmüller, S. (2022). A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis. *Journal of Computational Social Science*, (p. 1–73). https://doi.org/10.1007/s42001-022-00191-7
- Chapter 4: Wankmüller, S. & Heumann, C. (2021). How to estimate continuous sentiments from texts using binary training data. In Evang, K., Kallmeyer, L., Osswald, R., Waszczuk, J., & Zesch, T. (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 182–192). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.16

Each of the three articles presents a set of methods that are likely to provide a concrete improvement over specific current text analytic practices in political science. In order to make this contribution, the doctoral thesis transfers and makes use of concepts and developments from statistics, machine learning, and especially the field of NLP.

Since this work heavily draws on research from fields other than political science, this first chapter of the thesis serves as an introduction in two respects. First, the introductory

the article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* in this dissertation thus are addressing a larger social science audience.

chapter presents basic concepts in machine learning that are applied throughout the three articles (Section 1.1). It also gives an overview of the field of NLP as a larger backdrop for the thesis (Section 1.2). An additional special focus is given to the measurement of attitudes from texts (Section 1.3). Second, this introduction embeds each of these three articles in a larger background in order to work out the contribution of each article. This is achieved by pointing out connections between the introduced concepts and issues on the one hand and the articles and their research contributions on the other hand. Throughout this introductory chapter (but especially at the ends of Sections 1.1, 1.2, and 1.3) it is shown how the articles contribute to the issues raised. Finally, Section 1.4 points out one central shortcoming of the articles presented here and elaborates on how more systematic, inference-focused research procedures within the field of NLP would improve conclusions drawn in NLP and thereby also benefit text-based political science research.

1.1 Machine Learning

Machine learning is a subfield of artificial intelligence (AI) (Chollet, 2021, ch. 1.1). In machine learning, algorithms learn based on data (Goodfellow et al., 2016, p. 97). What *learning based on data* means becomes clear when comparing machine learning with symbolic AI—another subfield of AI (Chollet, 2021, ch. 1.1.1). In symbolic AI, a computing system is provided with data, on the one hand, and with rules that are manually created by humans and specify how the data are to be processed, on the other hand (Chollet, 2021, ch. 1.1.2). The computing system then processes the data according to the specified rules and provides the answer to a problem via an automated process (Chollet, 2021, ch. 1.1.2). In machine learning, in contrast, computing systems are provided with data and—in supervised machine learning—also with the respective answers (Chollet, 2021, ch. 1.1.2). The machine learning algorithm then finds structure in data or learns the function that maps from the data to the answers (Chollet, 2021, ch. 1.1.2). Hence, in machine learning, the processing functions are not explicitly provided but are learned (Chollet, 2021, ch. 1.1.2).

Machine learning algorithms can be grouped into several categories. The largest and most important distinction is between supervised and unsupervised learning.

1.1.1 Supervised Learning

Assume that there is a population of observational units (e.g. a group of people, a collection of images, or a corpus of documents) on which some form of statistical learning process is to occur. Assume also that a researcher assembles a set of units from this population, denoted as $D = (d_1, \ldots, d_i, \ldots, d_N)$, to form a training data set. In supervised learning, the researcher knows the output values $\boldsymbol{y} = [y_1, \ldots, y_i, \ldots, y_N]^\top$ for the training data

(Bishop, 2006, p. 3).[4] Therefore, for each d_i, the corresponding response value y_i is given: $(d_i, y_i)_{i=1}^N$.

As a raw data unit (e.g. a raw text file) cannot be directly inputted into a machine learning algorithm, the raw units first have to be made compatible for data analysis (Benoit, 2020, p. 463-464). This is done by converting each instance d_i into a *representation*, which is an abstraction of d_i (Benoit, 2020, p. 463-464). A frequently used way to generate representations, is to transform each d_i into a feature vector $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iu}, \ldots, x_{iU}]$. Element x_{iu} of the vector gives the value of example d_i on the uth feature. The features are the variables on which the instances are measured. They are the variables via which the raw data examples d_i are represented. Collectively, the training data instances' representations are given by an $N \times U$ matrix $\boldsymbol{X} = [\boldsymbol{x}_1 | \ldots | \boldsymbol{x}_i | \ldots | \boldsymbol{x}_N]^\top$.

Given the units' representations $\boldsymbol{X}$ and the units' output values $\boldsymbol{y}$, the researcher applies a machine learning algorithm to search the space of possible mappings between $\boldsymbol{X}$ and $\boldsymbol{y}$ to find a function $\hat{f}$ such that the estimated function's predictions, $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{X})$, are maximally close to the true values $\boldsymbol{y}$ (Chollet, 2021, ch. 1.1.5).

This general description of machine learning needs further elaboration with regard to two aspects: First, note that each learning algorithm can (only) learn a certain space of possible functions, $\tilde{f} \in \mathcal{H}$ (Vapnik, 1991, p. 832). $\mathcal{H}$ is known as the hypothesis space (Chollet, 2021, ch. 1.1.3). Thus, when applying a machine learning algorithm, the search is over the space of possible mappings between $\boldsymbol{X}$ and $\boldsymbol{y}$ *that the given machine learning algorithm can learn* (Vapnik, 1991, p. 832-833).

Second, the discrepancy between the true value y of a given input $\boldsymbol{x}$ and the value predicted by the candidate model $\tilde{f}(\boldsymbol{x})$ is captured by a loss function $\mathcal{L}(y, \tilde{f}(\boldsymbol{x}))$ (Vapnik, 1991, p. 832). The expected value of the loss, also called risk, is (Bishop, 2006, p. 46):

$$\mathcal{R}(\tilde{f}) = \int \int \mathcal{L}(y, \tilde{f}(\boldsymbol{x})) p(\boldsymbol{x}, y) \, \mathsf{d}\boldsymbol{x} \, \mathsf{d}y \tag{1.1}$$

$p(\boldsymbol{x}, y)$ is the unknown underlying joint distribution over data representation $\boldsymbol{x}$ and output y (Goodfellow et al., 2016, p. 109). $p(\boldsymbol{x}, y) = p(y|\boldsymbol{x})p(\boldsymbol{x})$ describes the underlying data generating process that generates representation vectors, $p(\boldsymbol{x})$, and then maps representation vectors to output values, $p(y|\boldsymbol{x})$ (Vapnik, 1991, p. 832).

Machine learning methods seek to minimize the risk $\mathcal{R}(\tilde{f})$ over the space of functions that a given machine learning algorithm is able to implement: $\arg\min_{\tilde{f} \in \mathcal{H}} \mathcal{R}(\tilde{f})$ (Vapnik, 1991, p. 832). The fundamental problem, however, is that $p(\boldsymbol{x}, y)$ is unknown (Vapnik, 1991, p. 832). Yet information about $p(\boldsymbol{x}, y)$ is contained in the training set instances, $(\boldsymbol{x}_i, y_i)_{i=1}^N$, that are assumed to have been independently generated from $p(\boldsymbol{x}, y)$ (Vapnik, 1991, p. 832). Hence, the risk $\mathcal{R}(\tilde{f})$ typically is approximated by the empirical risk, $\mathcal{R}_{emp}(\tilde{f})$, which is computed as the mean of the observed losses over all training set instances (Vapnik, 1991,

[4] Here, the output y_i is treated as a scalar. Yet the output also could be a vector $\boldsymbol{y}_i$.

p. 832-833; Goodfellow et al., 2016, p. 272-273):

$$\mathcal{R}_{emp}(\tilde{f}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, \tilde{f}(\boldsymbol{x}_i)) \tag{1.2}$$

Thus, when training a machine learning algorithm, the algorithm's hypothesis space $\mathcal{H}$ is searched in order to find the function $\hat{f}$ that minimizes the empirical risk (Vapnik, 1991, p. 832-833; Goodfellow et al., 2016, p. 272-273; Chollet, 2021, ch. 1.1.3):

$$\hat{f} = \arg\min_{\tilde{f}\in\mathcal{H}} \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, \tilde{f}(\boldsymbol{x}_i)) \tag{1.3}$$

This procedure is known as empirical risk minimization (Vapnik, 1991, p. 832-833).

In machine learning terminology, the output of this training process, $\hat{f}$, is called a model (Molnar, 2022, ch. 2.3).[5] The trained model then can be applied to a set of new, yet unseen data $\boldsymbol{X}^*$ that have not been used in training (Bishop, 2006, p. 3; James et al., 2013, p. 30). The ultimate goal in supervised machine learning is to train a model that generalizes well (Bishop, 2006, p. 2; Goodfellow et al., 2016, p. 108). A well-generalizing model makes accurate predictions for unseen data $\boldsymbol{X}^*$ (Bishop, 2006, p. 2; Goodfellow et al., 2016, p. 108).

How well a model generalizes is assessed via the generalization error, which is also known as the test error $\mathcal{GE}$ (Goodfellow et al., 2016, p. 108). The generalization error is estimated as the mean over the losses of the instances in a test set (Grosse, 2020c, p. 1-2):

$$\widehat{\mathcal{GE}}(\hat{f}) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{L}(y_m^*, \hat{f}(\boldsymbol{x}_m^*)) \tag{1.4}$$

The test set tuples $(\boldsymbol{x}_m^*, y_m^*)_{m=1}^M$ have not been involved in the training process and the usual assumption is that the test set observations have been independently drawn from the same joint distribution $p(\boldsymbol{x}, y)$ as the training instances (Goodfellow et al., 2016, p. 109). Training and test data points are assumed to be i.i.d. (independent and identically distributed) (Goodfellow et al., 2016, p. 109).[6] Both the training and test data points are assumed to have been generated by the same data generating process that $p(\boldsymbol{x}, y)$ describes (Goodfellow et al., 2016, p. 109).

[5]The terminology in machine learning differs in various respects from the terminology in statistics. In machine learning terminology, a machine learning algorithm is a processing system that can be expressed in mathematical terms (Brownlee, 2020a). An algorithm can learn a specific range of functions and thus defines a hypothesis space (Vapnik, 1991, p. 832; Chollet, 2021, ch. 1.1.3). When a machine learning algorithm learns from data, it learns a machine learning model (Brownlee, 2020a; Molnar, 2022). Hence, an algorithm is a learning approach (e.g. logistic regression, deep neural network,...), whereas the term *model* refers to the specific function with specific parameter values that results from training (Brownlee, 2020a; Molnar, 2022). The trained model can be used to make predictions (Brownlee, 2020a; Molnar, 2022).

[6]Note that in practice there are situations in which this assumption is deliberately not met. For example,

Machine learning, therefore, can be understood as follows: Assume that for data points generated from $p(\boldsymbol{x}, y)$ the true but unknown relationship between $\boldsymbol{X}$ and $\boldsymbol{y}$ is

$$\boldsymbol{y} = f(\boldsymbol{X}) + \boldsymbol{\epsilon} \tag{1.5}$$

where function $f(\boldsymbol{X})$ describes the systematic relation between $\boldsymbol{X}$ and $\boldsymbol{y}$ (James et al., 2013, p. 16). $\boldsymbol{\epsilon}$ is the random error term that captures inherent non-systematic randomness and $\mathbb{E}(\boldsymbol{\epsilon}) = 0$ (James et al., 2013, p. 16, 18-19). Machine learning seeks to learn a function $\hat{f}$ that approximates the true systematic relation f in the sense that $\hat{f}$ provides an accurate mapping between inputs and outputs for data points that have been generated from $p(\boldsymbol{x}, y)$ (Bishop, 2006, p. 38; James et al., 2013, p. 16-18, 21). This is, the aim is to have $y_i \approx \hat{f}(\boldsymbol{x}_i)$ for any data unit from $p(\boldsymbol{x}, y)$ (James et al., 2013, p. 21). *Learning* in supervised machine learning thus essentially is function approximation (Vapnik, 1991, p. 832; James et al., 2013, p. 16-17).

As $p(\boldsymbol{x}, y)$ is unknown, researchers make use of observed training data points in order to approximate f (Goodfellow et al., 2016, p. 272-273). The empirical joint distribution over training data, $\hat{p}(\boldsymbol{x}, y)$, is used as a substitute for $p(\boldsymbol{x}, y)$ (Goodfellow et al., 2016, p. 272-273). Although training is conducted on training data, the aim nevertheless is to learn a model $\hat{f}$ that provides an approximation of the general systematic relation f. The actual goal in supervised machine learning is to have a well-generalizing model; a model that has learned the systematic relationship between inputs and outputs among instances with a shared data generating process; a model that—because it captures the general systematic relationship rather than idiosyncrasies or inherent non-systematic randomness in the training data—makes accurate predictions on previously unused data and therefore has a low generalization error. In practice, however, several issues arise when searching for such a model.

1.1.1.1 Overfitting and the Bias-Variance Trade-Off

A first issue that comes up when seeking to find a model that generalizes well to data points for which the output values are not known (or not disclosed) is due to the fact that approximating f only is possible on the training data $\boldsymbol{X}$ for which the target values $\boldsymbol{y}$ are revealed. This carries the risk of overfitting on the training data set. Overfitting means that an algorithm learns non-systematic, idiosyncratic patterns in the training data which harms its generalization performance (James et al., 2013, p. 22, 32; Goodfellow et al., 2016, p. 110).

---

in transfer learning (see Section 1.2.3.8) the data used in pretraining may come from a different data generating process as the data of the target task (Ruder, 2019a, p. 42, 86). Moreover, active learning and techniques for over- or undersampling are methods that intentionally alter the training data distribution to reduce the costs of annotating training examples or to change the cost-sensitivity of an algorithm (see this dissertation's article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* in Chapter 3).

Overfitting is more likely to occur for learning algorithms that have a high capacity (Goodfellow et al., 2016, p. 109-110). Learning algorithms with a high capacity have a large hypothesis space $\mathcal{H}$; they are flexible in the sense that they can approximate a large spectrum of various functional forms (James et al., 2013, p. 22, 32; Goodfellow et al., 2016, p. 110, 116). Overfitting is indicated by the test error (see Equation 1.4) being substantively higher than the training error (James et al., 2013, p. 32). (The training error is Equation 1.4 computed on the training data (Grosse, 2020c, p. 1-2).) More specifically, a learning algorithm is said to overfit on the training data if another learning algorithm with a lower capacity would have had a lower test error (James et al., 2013, p. 32).

Applying a flexible learning algorithm thus can mean risking overfitting. Yet an algorithm should also be flexible enough to adequately capture the systematic relationship between inputs and outputs in a more complex data structure. A learning algorithm whose capacity is too low given the complexity of the data structure is likely to underfit the training data as it is unable to approximate the function mapping from $\boldsymbol{X}$ to $\boldsymbol{y}$ (Goodfellow et al., 2016, p. 109-110). The tension between using a sufficiently flexible algorithm, on the one hand, and the risk of overfitting, on the other hand, is a constant theme in machine learning and is manifested in the bias-variance trade-off.

In general, the bias-variance trade-off refers to the trade-off between bias and variance—which are two error sources influencing the expected generalization error (James et al., 2013, p. 33-34). If one wanted to know the expected generalization error of one learning algorithm, then one would train one model on each possible training data set D_{train} of size N that can be drawn from the data generating distribution $p(\boldsymbol{x}, y)$ and one then would average over these models' predictions for an unused test data point. The bias is the difference between the test data point's true value and the average of the predicted values (Bishop, 2006, p. 149). The bias is a systematic error that results from approximating the true underlying function f with a learning algorithm whose encoded assumptions do not quite apply to the given learning problem (James et al., 2013, p. 35). For example, a learning algorithm that assumes the relationship between inputs and outputs to be linear is likely to have a high bias if the true underlying function f is highly nonlinear (James et al., 2013, p. 35). Variance, on the other hand, captures the amount by which the models' predictions vary around the average of their predictions (Bishop, 2006, p. 149). Variance thus indicates the variability in the learning algorithm's predictions that results from the learning algorithm being trained on differently composed training data sets (James et al., 2013, p. 34-35).

To have an as small as possible expected generalization error, one seeks to have a learning method with low bias and low variance (James et al., 2013, p. 36). Yet there is a trade-off: A learning method with high capacity is likely to be able to closely approximate the relationship between inputs and outputs (low bias) but, due to its flexibility, may too strongly adapt to the training data and may estimate a quite different functional form for another training data set (high variance) (James et al., 2013, p. 35-36). A method with low capacity, in contrast, due to its restricted flexibility, is likely to be less volatile to changes

Classification vs. Regression. Supervised learning can be used to model discrete and continuous output variables. A supervised learning task in which the response variable y is discrete, is called classification (James et al., 2013, p. 28). And—at odds with the terminology in statistics—a supervised learning task in which the response variable y is continuous, is called regression (James et al., 2013, p. 28). Note that many machine learning algorithms can be used for classification and regression tasks (e.g. K-nearest neighbors, tree-based algorithms, neural networks) (James et al., 2013, p. 29). If the classification task is to assign each data instance to one out of two mutually exclusive and exhaustive categories (where typically $y_i \in \{0, 1\}$), this is called binary classification (Bishop, 2006, p. 38; Grosse, 2020a, p. 1). A classification task in which there are more than two mutually exclusive and exhaustive categories, $y_i \in \{\mathcal{G}_1, \ldots, \mathcal{G}_c, \ldots, \mathcal{G}_c\}$, is called multi-class classification (Grosse, 2020b, p. 8). Multi-label classification, in contrast, refers to tasks in which the categories are not mutually exclusive and each data instance can be assigned to more than one category (Zhang & Zhou, 2014).

in the training data (low variance) but for the same reason may fail to produce an adequate mapping between $\boldsymbol{X}$ and $\boldsymbol{y}$ (which then causes high bias) (James et al., 2013, p. 35-36). Consequently, the goal is to have a learning algorithm that—given the complexity of the data structure—is flexible enough to achieve an adequate approximation (resulting in low bias) and at the same time is not so flexible that overfitting occurs (and thus variance stays low) (James et al., 2013, p. 36).

Underfitting [overfitting] can be remedied by choosing a more [less] flexible learning algorithm or by increasing [decreasing] the hypothesis space the given learning algorithm can learn (Goodfellow et al., 2016, p. 110, 116-117; James et al., 2013, p. 31-32). An important approach to address overfitting are regularization strategies which are presented in the context of deep neural networks in Section 1.2.3.4.

1.1.1.2 Loss Functions

Another issue that arises in machine learning is that the loss function that is used in the optimization process (i.e. the search process) is often not the same loss function that is used for evaluating the model's prediction performance on yet unseen test data. If a loss function that a researcher wishes to use cannot be optimized in an efficient way, the usage of a surrogate loss function is required to facilitate or enable the optimization process (Goodfellow et al., 2016, p. 273). The loss function applied in the optimization process (see Equation 1.3) then is a surrogate loss function for the loss function used for model evaluation (see Equation 1.4)—the latter being the actual loss function of interest (Goodfellow et al., 2016, p. 273).

To illustrate, in a binary classification task, one might like to assess the performance of a model via the 0–1 loss that assumes a value of 0 if the prediction is correct and assumes a value of 1 if the prediction is incorrect. One then would want to search for a model $\hat{f}$ that minimizes the empirical risk, where the loss is given by the 0–1 loss. Yet the 0–1 loss is a discontinuous step function whose derivative is 0 or undefined (Grosse, 2020b, p. 2). Therefore, the 0–1 loss is an unsuitable function for the application of effective optimization algorithms that make use of derivatives (e.g. gradient descent) (Grosse, 2020b, p. 2). Hence, optimization typically is conducted on a surrogate loss function (in the case of binary classification tasks often via the logistic loss) for which optimization is more efficient (Goodfellow et al., 2016, p. 273).

A reduction of the surrogate loss on the training data set in expectation will yield a reduction in the actual loss of interest (Goodfellow et al., 2016, p. 273). However, there can be differences in the functions' courses. For example, it can occur that the surrogate loss on the training data set still is decreasing whilst the 0-1 loss evaluated on the training set or the test set remains constant (Goodfellow et al., 2016, p. 274; Mosbach et al., 2021, p. 8).

The loss function that is used as a measure of discrepancy between predictions and true values in the process of training a machine learning algorithm (see Equation 1.3) fundamentally shapes how the training process can go about (e.g. whether the loss function is convex or whether effective optimization algorithms, such as gradient descent, can be applied) (Grosse, 2020b, p. 2, 9-10). The loss function that is used for assessing a model's prediction performance (see Equation 1.4) determines how the performance of a trained model is measured. It defines how the discrepancies between predictions and true values are finally evaluated.

In this dissertation, the application of deep neural networks to classification tasks plays an important role. Deep neural networks are parametric machine learning methods and their parameters are typically determined in a maximum likelihood framework (Bishop, 2006, p. 226; Goodfellow et al., 2016, p. 174).[7] Therefore, the training of parametric learning approaches in the maximum likelihood framework is outlined in the following. While doing so, common loss functions that are used for optimization on classification tasks are presented. Afterward, in Section 1.1.1.3, loss functions that are widely used when evaluating a trained model's prediction performance in classification settings are introduced.

But first, a note on parametric methods:[8] Parametric methods approximate the true un-

[7] But see, for example, Jospin et al. (2022) on Bayesian Neural Networks.

[8] Statistical learning procedures that aim at approximating a true underlying function f can be grouped into parametric vs. non-parametric methods (James et al., 2013, p. 21): Parametric methods are explained in the main text. A shortcoming of parametric methods is the possibility that the assumed functional form may poorly match the true underlying function f (James et al., 2013, p. 22). Non-parametric methods invoke few or no assumptions about the functional form of the mapping between inputs and outputs (James et al., 2013, p. 23, 104). An example of a non-parametric method is K-nearest-neighbors regression

derlying function f with a function $\mathsf{f}(\boldsymbol{\theta})$ whose precise functional form is defined by a finite set of parameters $\boldsymbol{\theta}$ (Bishop, 2006, p. 68, 120). This is, parametric methods start with a priori selecting a general functional form f that is to be used in approximating f (e.g. by choosing to apply a linear model with $y_i = \mathcal{N}(\mu_i, \sigma^2)$ and $\mu_i = \boldsymbol{x}_i\boldsymbol{\beta}$) (James et al., 2013, p. 21). Then, parametric approaches determine the precise mathematical relationship between inputs and outputs that f describes by estimating f's parameters $\boldsymbol{\theta}$ from the training data (James et al., 2013, p. 21). (In the example here $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \sigma^2\}$.) In doing so, parametric methods narrow down the problem of approximating the true underlying function f to the problem of estimating suitable values for a finite set of parameters $\boldsymbol{\theta}$ (James et al., 2013, p. 21-22). Parametric methods consequently aim at finding the set of parameter values in the parameter space, $\tilde{\boldsymbol{\theta}} \in \Theta$, that minimizes the empirical risk $\mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}})$ (Goodfellow et al., 2016, p. 272-273):[9]

$$\hat{\boldsymbol{\theta}} = \arg\min_{\tilde{\boldsymbol{\theta}} \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})) \tag{1.6}$$

If empirical risk minimization is conducted in a maximum likelihood framework, then the negative log-likelihood is used as the loss function (Goodfellow et al., 2016, p. 174): The likelihood function, $p(\boldsymbol{y}|\mathsf{f}(\boldsymbol{X}, \tilde{\boldsymbol{\theta}}))$, indicates how likely it is to observe data outputs $\boldsymbol{y}$ given the parameter values $\tilde{\boldsymbol{\theta}}$ of f (Bishop, 2006, p. 22, 28-29). (To comply with common notation, f is suppressed in the following and the likelihood function is expressed as $p(\boldsymbol{y}|\boldsymbol{X}, \tilde{\boldsymbol{\theta}})$ (King, 1998, p. 22; Bishop, 2006, p. 28).) In maximum likelihood estimation, the aim is to find the set of parameters $\hat{\boldsymbol{\theta}}$ that maximize the likelihood function (Bishop, 2006, p. 23). If individual units are assumed to be i.i.d., then this aim is (Goodfellow et al., 2016, p. 131)

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\tilde{\boldsymbol{\theta}} \in \Theta} \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}) \tag{1.7}$$

This maximization problem can be turned into a minimization problem over the negative log-likelihood (Goodfellow et al., 2016, p. 131, 174):

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\min_{\tilde{\boldsymbol{\theta}} \in \Theta} -\sum_{i=1}^{N} log\, p(y_i|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}) \tag{1.8}$$

in which the value predicted for test set unit $\boldsymbol{x}^*_m$ is the average of the values of the K training units closest to $\boldsymbol{x}^*_m$ (James et al., 2013, p. 105). Because they make few assumptions about the function's structure, non-parametric methods are more flexible in the shapes they can approximate (James et al., 2013, p. 23). In non-parametric methods, the number of parameters increases as a function of training data set size (Ghahramani, 2015). In parametric methods, in contrast,—because they narrow down the problem of approximating f to the problem of estimating a finite set of parameters—the number of parameters is fixed irrespective of the available amount of data (Ghahramani, 2015).

[9] In contrast to the general formulation of empirical risk minimization in Equation 1.3, the empirical risk now is a function of parameter values: $\mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}})$. As the general functional form is set a priori, the aim now is to find a finite set of parameters $\hat{\boldsymbol{\theta}}$ that minimizes the empirical risk rather than to search for an arbitrary functional form $\hat{f}$ that minimizes the empirical risk.

Machine learning methods that are optimized using maximum likelihood then simply take the negative log-likelihood as the loss function (Goodfellow et al., 2016, p. 174). This is, they define

$$\mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})) = -log\, p(y_i|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}) \tag{1.9}$$

Accordingly, the exact form of the loss function is determined by the function selected for $p(y_i|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})$ (Goodfellow et al., 2016, p. 177). For real-valued output values, the mean squared error is a commonly used loss function (Goodfellow et al., 2016, p. 177). In binary classification tasks with $y_i \in \{0, 1\}$, a typical choice is the Bernoulli distribution (Goodfellow et al., 2016, p. 178). For a single observational unit $-log\, p(y_i|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})$ then becomes

$$-\, log[\pi_i^{y_i}(1-\pi_i)^{1-y_i}] = -y_i\, log(\pi_i) - (1-y_i)\, log(1-\pi_i) \tag{1.10}$$

Here, $\pi_i = p(y_i = 1|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})$ is the probability that unit i falls into category 1. Equation 1.10 is known as the cross-entropy loss (Grosse, 2020b, p. 5). If π_i is modeled to be generated by the standard logistic function, also known as logistic sigmoid function, one obtains the logistic loss (also known as log loss) (Brownlee, 2019; Grosse, 2020b, p. 6):

$$-\, y_i\, log\left[\frac{exp(\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}))}{1 + exp(\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}))}\right] - (1-y_i)\, log\left[\frac{exp(-\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})}{1 + exp(-\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})}\right] \tag{1.11}$$

$$= y_i\, log[1 + exp(-\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}))] + (1-y_i)\, log[1 + exp(\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}))] \tag{1.12}$$

In multi-class classification tasks, in which y_i falls into one out of C categories, $y_i \in \{\mathcal{G}_1, \ldots, \mathcal{G}_c, \ldots, \mathcal{G}_C\}$, the negative log-likelihood for a single unit i is

$$-\sum_{c=1}^{C} 1_{[y_i=\mathcal{G}_c]}\; log\, \pi_{ic} \tag{1.13}$$

where $1_{[y_i=\mathcal{G}_c]}$ is 1 if $y_i = \mathcal{G}_c$ and 0 otherwise (Grosse, 2020b, p. 8-9). $\pi_{ic} = p(y_i = \mathcal{G}_c|\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})$ gives the probability predicted by the model that unit i falls into class $\mathcal{G}_c$. To predict π_{ic} for each class, the softmax function is used (Goodfellow et al., 2016, p. 181).

$$\pi_{ic} = \frac{exp(\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})_c)}{\sum_{j=1}^{C} exp(\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})_j)} \tag{1.14}$$

$\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})_c$ denotes the cth element in a model's C-dimensional output vector and can be described as the score for the membership of unit i in class $\mathcal{G}_c$. Accordingly, in a multi-class classification problem with C categories, the softmax function takes as an input a C-dimensional vector of scores (one for each class) and produces as an output a C-dimensional vector (Goodfellow et al., 2016, p. 181). The cth element of the resulting vector is π_{ic}, where $\pi_{ic} \in [0, 1]$ and $\sum_{c=1}^{C} \pi_{ic} = 1$ and π_{ic} is interpreted as the probability that $y_i = \mathcal{G}_c$ (Goodfellow et al., 2016, p. 181).

	truly positive $y = 1$	**truly negative** $y = 0$	
predicted positive $\hat{y} = 1$	True Positives (TP)	False Positives (FP) Type I Error	Precision, Positive Predictive Value: $\frac{TP}{TP+FP}$
predicted negative $\hat{y} = 0$	False Negatives (FN) Type II Error	True Negatives (TN)	Negative Predictive Value: $\frac{TN}{TN+FN}$
	Recall, Sensitivity, True Positive Rate: $\frac{TP}{TP+FN}$	Specificity, True Negative Rate: $\frac{TN}{TN+FP}$	Accuracy: $\frac{TP+TN}{N}$ Error Rate: $\frac{FP+FN}{N}$

Table 1.1: **Confusion Matrix.** The confusion matrix is a 2×2 contingency table that maps the true values $\boldsymbol{y}$ (columns) against predicted values $\hat{\boldsymbol{y}}$ (rows). In the cells of the table, the number of True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN) are given. Similar to the depiction of the confusion matrix in Wikipedia (https://en.wikipedia.org/wiki/Confusion_matrix), at the margins of the table, it is shown how common performance measures are computed.

1.1.1.3 Assessing Prediction Performance

Loss functions that are used for the evaluation of a trained model's predictions (see Equation 1.4) are commonly known as performance measures or performance metrics.[10] Many widely used performance measures for classification tasks can be derived from the confusion matrix which is a 2×2 contingency table that maps true values $\boldsymbol{y}$ against predicted values $\hat{\boldsymbol{y}}$ (Han et al., 2012, p. 366). In the case of a binary classification task with $y_i \in \{0, 1\}$, the confusion matrix has the form of Table 1.1. If $y_i = 1$ $[y_i = 0]$ then an instance is said to belong to the positive [negative] class. In the cells of the confusion matrix the following quantities are given (Han et al., 2012, p. 365-366):

- True Positives (TP): The number of instances that truly belong to the positive class and are correctly predicted to belong to the positive class.
- True Negatives (TN): The number of instances that truly belong to the negative class and are correctly predicted to belong to the negative class.

[10] So far, the loss function $\mathcal{L}(y_i, \hat{y}_i)$ has been introduced as a function that indicates the *discrepancy* between true and predicted values. The smaller the loss value, $L = \mathcal{L}(y_i, \hat{y}_i)$, the better. Yet, in the context of evaluation, $\mathcal{L}(y_i, \hat{y}_i)$ also can be a function that measures the *agreement* or *closeness* between true and predicted values. If this is the case, then the higher the value returned by function $L = \mathcal{L}(y_i, \hat{y}_i)$ the better.

- False Negatives (FN): The number of instances that truly belong to the positive class but are incorrectly predicted to belong to the negative class.
- False Positives (FP): The number of instances that truly belong to the negative class but are incorrectly predicted to belong to the positive class.

If N is the number of all instances, then (Han et al., 2012, p. 365-366):

- Accuracy: $\frac{TP+TN}{N}$
- Error Rate: $\frac{FP+FN}{N}$
- Recall (also known as Sensitivity, True Positive Rate): $\frac{TP}{TP+FN}$
- Specificity (also known as True Negative Rate): $\frac{TN}{TN+FP}$
- Precision (also known as Positive Predictive Value): $\frac{TP}{TP+FP}$
- Negative Predictive Value: $\frac{TN}{TN+FN}$

Accuracy and the error rate are global performance measures. As such they are not appropriate when the distribution of true class labels is imbalanced (meaning that the proportions of one class or several classes are substantively smaller than the proportion of other classes in the data) (Manning et al., 2008, p. 155; Kuhn & Johnson, 2013, p. 419). In a binary classification problem with imbalanced class labels a classifier that would assign all instances to the majority class, would have a high accuracy and a small error rate (Manning et al., 2008, p. 155). More adequate and widely used measures in such situations are recall and precision (Manning et al., 2008, p. 154-156). Recall solely focuses on the truly positive instances and gives the share of instances that have been correctly predicted to be positive among all truly positive instances. Precision only takes into account all instances that have been predicted to belong to the positive class and gives the share of truly positive instances that correctly have been classified into this positive class. The relationship between precision and recall is characterized by a trade-off (Manning et al., 2008, p. 156): A classification method that has a low [high] threshold for assigning instances to the positive class, is likely to have a high [low] recall but low [high] precision.

The F_ω-Score combines precision and recall into their weighted harmonic mean (Manning et al., 2008, p. 156):

$$F_\omega = \frac{(\omega^2 + 1) \cdot Precision \cdot Recall}{\omega^2 \cdot Precision + Recall} \tag{1.15}$$

In many applications ω is set to 1 (Manning et al., 2008, p. 156-157). The resulting F_1-Score is the harmonic mean of precision and recall (Manning et al., 2008, p. 156):

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{1.16}$$

In a binary classification setting, the positive class is usually the class of interest and

binary performance measures, such as precision, recall, and the F_1-Score, are reported for this positive class as presented here.

In the evaluation of multi-class classification tasks, the predictions for each class can be regarded as a binary classification problem for this class vs. the rest of the classes (scikit-learn Developers, 2021). Hence, the binary performance measures can be computed for each class independently (scikit-learn Developers, 2021). For the cth class (denoted by $\mathcal{G}_c$) recall then gives the share of instances that have been predicted to belong to the cth class among all instances that are truly in the cth class: $Recall_{\mathcal{G}_c} = \frac{TP_{\mathcal{G}_c}}{TP_{\mathcal{G}_c}+FN_{\mathcal{G}_c}}$. Precision then gives the share of instances that truly belong to the cth class among all instances that have been predicted to fall into the cth class: $Precision_{\mathcal{G}_c} = \frac{TP_{\mathcal{G}_c}}{TP_{\mathcal{G}_c}+FP_{\mathcal{G}_c}}$.

In a multi-class classification task with C categories and C separate performance scores for each metric, there is the question of how to average the separate measures (scikit-learn Developers, 2021). Three common ways of averaging the per-class scores are *micro*, *macro*, and *weighted* (scikit-learn Developers, 2021):

- *macro* assigns the same weight to each class score (scikit-learn Developers, 2021):
  - $\frac{1}{C}\sum_{c=1}^{C} Recall_{\mathcal{G}_c}$
  - $\frac{1}{C}\sum_{c=1}^{C} Precision_{\mathcal{G}_c}$
  - $\frac{1}{C}\sum_{c=1}^{C} F_1\text{-}Score_{\mathcal{G}_c}$
- *weighted* weights each class score according to the proportion of the class in the data (scikit-learn Developers, 2021):
  - $\sum_{c=1}^{C} \frac{|\mathcal{G}_c|}{\sum_{c=1}^{C}|\mathcal{G}_c|} Recall_{\mathcal{G}_c}$
  - $\sum_{c=1}^{C} \frac{|\mathcal{G}_c|}{\sum_{c=1}^{C}|\mathcal{G}_c|} Precision_{\mathcal{G}_c}$
  - $\sum_{c=1}^{C} \frac{|\mathcal{G}_c|}{\sum_{c=1}^{C}|\mathcal{G}_c|} F_1\text{-}Score_{\mathcal{G}_c}$
- *micro* sums up the separate $TP_{\mathcal{G}_c}$ values into one global TP value that captures the number of all true positive predictions across the classes and then does the same for $FN_{\mathcal{G}_c}$ and $FP_{\mathcal{G}_c}$. Afterward, recall, precision, and the F_1-Score are computed as described for the binary case above. Note that—because in multi-class classification all wrongly predicted instances are False Negatives (FN) and False Positives (FP)—the micro-averaged F_1-Score has the same value as micro-averaged precision, micro-averaged recall, and accuracy (Shmueli, 2019).

1.1.1.4 Resampling

When evaluating the performance of a learning method, one would actually like to know the expected generalization error, $\mathcal{EGE}$, which can be understood as the expectation of the loss

function under the data generating distribution $p(\boldsymbol{x}, y)$ (Bischl et al., 2012, p. 251).[11]

$$\mathcal{EGE}(\hat{f}) = \int \int \mathcal{L}(y, \hat{f}(\boldsymbol{x}))p(\boldsymbol{x}, y)\, \mathsf{d}\boldsymbol{x}\, \mathsf{d}y \tag{1.17}$$

As $p(\boldsymbol{x}, y)$ is unknown, one uses observed data to approximate the expected generalization error. An observed training data set, $\mathsf{D}_{train} = (\boldsymbol{x}_i, y_i)_{i=1}^{N}$, can be used to train a model and an observed test set, $\mathsf{D}_{test} = (\boldsymbol{x}_m^*, y_m^*)_{m=1}^{M}$, can be used for the evaluation of the trained model's performance.[12] This is, one can compute an estimate of the generalization error, that has been presented in Equation 1.4, and use it as an estimate of the expected generalization error.

Note, however, that the estimate of the generalization error in Equation 1.4 is a measure of the prediction error on a *particular* test set $\mathsf{D}_{test} = (\boldsymbol{x}_m^*, y_m^*)_{m=1}^{M}$ of a model that has been trained on a *particular* training data set, $\mathsf{D}_{train} = (\boldsymbol{x}_i, y_i)_{i=1}^{N}$ (Hastie et al., 2009, p. 220). Each time a learning algorithm is trained on another training set, a slightly different model with slightly different parameter values will be learned (Bishop, 2006, p. 148). Consequently, each time the trained model will produce a different generalization error on the test set (Bishop, 2006, p. 148). Thus, the value of the generalization error on test set D_{test} depends on the specific training data set an algorithm has been trained on (Hastie et al., 2009, p. 220). To emphasize this dependency, one can write Equation 1.4 as (Bischl & Molnar, 2018, Session 9, p. 16)

$$\widehat{\mathcal{GE}}_{\mathsf{D}_{test}}(\hat{f}_{\mathsf{D}_{train}}) = \frac{1}{|\mathsf{D}_{test}|} \sum_{(\boldsymbol{x}_m^*, y_m^*) \in \mathsf{D}_{test}} \mathcal{L}(y_m^*, \hat{f}_{\mathsf{D}_{train}}(\boldsymbol{x}_m^*)) \tag{1.18}$$

The problem when using Equation 1.18 as an estimate of the expected generalization error in Equation 1.17 is that the value computed in Equation 1.18 will depend on the composition of the training set and the composition of the test set used (James et al., 2013, p. 178). Another train-test set composition will yield another value (James et al., 2013, p. 178). The value computed in Equation 1.18 can exhibit considerable variability between different train-test set compositions (James et al., 2013, p. 178).

In order to achieve a better estimation of the expected generalization error, resampling techniques can be applied (Bischl et al., 2012, p. 253). Resampling techniques estimate the expected generalization error by (1) separating the entire set of available labeled data D into a training set, $\mathsf{D}_{train}^{\mathsf{k}}$, and a test set, $\mathsf{D}_{test}^{\mathsf{k}}$, (2) training an algorithm on the training set and estimating the trained model's generalization error on the test set, (3) repeating

[11] Note that, for reasons of readability, the notation here does not explicitly indicate the set of parameter values $\boldsymbol{\theta}$ that a parametric function f is characterized by. Note also that the expected generalization error here in Equation 1.17 is the same as the risk in Equation 1.1—except that the risk is the expectation of the loss function used in optimization whereas the expected generalization error is the expectation of the loss function used for evaluation.

[12] Note that here the assumption is always that all instances in the training set and the test set are i.i.d. observations from $p(\boldsymbol{x}, y)$.

this process K times, and then (4) estimating the expected generalization error as (Bischl et al., 2012, p. 253; Bischl & Molnar, 2018, Session 10, p. 3)

$$\widehat{\mathcal{EGE}} = \frac{1}{\mathsf{K}} \sum_{\mathsf{k}=1}^{\mathsf{K}} \widehat{\mathcal{GE}}_{\mathsf{D}^{\mathsf{k}}_{test}}(\hat{f}_{\mathsf{D}^{\mathsf{k}}_{train}}) = \frac{1}{\mathsf{K}} \sum_{\mathsf{k}=1}^{\mathsf{K}} \frac{1}{|\mathsf{D}^{\mathsf{k}}_{test}|} \sum_{(\boldsymbol{x}^*_m, y^*_m) \in \mathsf{D}^{\mathsf{k}}_{test}} \mathcal{L}(y^*_m, \hat{f}_{\mathsf{D}^{\mathsf{k}}_{train}}(\boldsymbol{x}^*_m)) \tag{1.19}$$

A common resampling technique is K-fold cross-validation. Here the entire set of available labeled data D are separated into K non-overlapping folds of equal size (Bischl et al., 2012, p. 254). Then, for each fold $\mathsf{k} \in \{1, \ldots, \mathsf{K}\}$, the kth fold is set aside as a test set and an algorithm is trained on all folds except the kth fold (James et al., 2013, p. 181). After training, the generalization error on the kth fold is estimated (James et al., 2013, p. 181). As soon as this process has been repeated for each of the K folds, an estimate of the expected generalization error is calculated as the mean over the K obtained generalization errors (just as in Equation 1.19 above) (Bischl et al., 2012, p. 253, 254).

Bootstrapping is another resampling strategy. Here, given an annotated data set D of size N, K training sets are created by drawing randomly with replacement from D (Bischl et al., 2012, p. 255). For each training set $\mathsf{D}^{\mathsf{k}}_{train}$, the test set is composed of those instances that have not been sampled into $\mathsf{D}^{\mathsf{k}}_{train}$. This is, $\mathsf{D}^{\mathsf{k}}_{test} = \mathsf{D} \setminus \mathsf{D}^{\mathsf{k}}_{train}$ (Bischl et al., 2012, p. 255). The expected generalization error then is estimated as in Equation 1.19 (Bischl et al., 2012, p. 253).

Now, however, it is the case that one usually does not only want to estimate the expected generalization error of a trained model, but that one also—in a previous step—wants to select a type of algorithm or a hyperparameter setting that on the given task at hand has the best generalization performance (Bischl et al., 2012, p. 250).[13] Consequently, one usually also wants to perform model selection and/or hyperparameter tuning besides model evaluation (Bischl et al., 2012, p. 250). If one wants to assess and compare the estimated expected generalization error of different learning algorithms and hyperparameter settings, and then wants to additionally estimate the expected generalization error of a finally selected model without re-using data for model evaluation that has already been used in model selection or hyperparameter tuning, a nested resampling approach is required (Bischl et al., 2012, p. 257-258). In nested resampling, there are inner and outer resampling loops. The inner loops are used for model selection/hyperparameter tuning and the outer loops are used for model evaluation (Bischl et al., 2012, p. 258-259). (For an illustration see Bischl et al. (2012, p. 259).)

Resampling techniques imply that each candidate model is trained and evaluated several times. Resampling thus can require substantial computational resources (James et al., 2013, p. 175). In the field of NLP, it can be computationally costly to train a model even once. Resampling techniques, let alone nested resampling approaches, are therefore not

[13] In the context of machine learning, a hyperparameter is a parameter that is set a priori by the researcher and is not estimated from data (Brownlee, 2017).

widely used in NLP. It is relatively common that a labeled data set is split into one training set for training, one validation set for model selection and hyperparameter tuning, and one test set for the evaluation of model performance (see e.g. Lai et al., 2017; Rajpurkar et al., 2018; Wang et al., 2019). Hence, the presented model performances in NLP are often estimates of the generalization error (and thus are based on a single train-test split as in Equation 1.18) and are not estimates of the expected generalization error as in Equation 1.19.

1.1.1.5 Supervised Learning, Text Data, and Scientific Research

Supervised machine learning primarily focuses on making predictions $\hat{\boldsymbol{y}}$ for data inputs $\boldsymbol{X}$. In doing so, the aim is to approximate the true underlying function f (only) to the extent that the mapping from inputs to outputs is as accurate as possible (Molnar, 2014, p. 7).[14] Supervised machine learning thus is not concerned with adequately modeling the underlying data generating process or identifying causal relationships (Breiman, 2001b, p. 199; Molnar, 2014, p. 7). The goal (merely) is to imitate f in order to have a well-generalizing model that produces accurate predictions (Breiman, 2001b, p. 199). In unsupervised learning, the aim is to learn structures in the data. And again, this is usually not done in order to understand the mechanisms of the data generating process. The aim (merely) is to uncover interesting, useful patterns in data (Grimmer & Stewart, 2013, p. 269, 270).

The machine learning mindset thus contrasts with the mode of inquiry in other scientific fields—such as political science—in which identifying and estimating causal effects is a major goal (King et al., 1994, p. 8). Nevertheless, machine learning can make valuable contributions to these scientific fields. One significant contribution is that machine learning can improve the conceptualization, operationalization, and measurement of concepts: To empirically test hypotheses between theoretical concepts, the theoretical concepts must be precisely defined and translated into empirically measurable variables (also named indicators) (Kellstedt & Whitten, 2009, p. 9-10; Schnell et al., 2018, p. 111, 114). In political science, this is often conducted in a deductive process (Ahlquist & Breunig, 2012, p. 94). First, the theoretical concept—that often initially exists as a vague notion in mind—is described, defined, and given a name (Ahlquist & Breunig, 2012, p. 94; Schnell et al., 2018, p. 112-113). In this process of conceptualization, the dimensionality of the concept also has to be determined (Schnell et al., 2018, p. 112-113). Second, after a concept and its dimensionality have been defined, the operationalization follows. In the operationalization, measurement instructions are set up that specify how the concept (and its dimensions) are transformed into measurable indicators (Schnell et al., 2018, p. 113-114). Operationalization specifies which variables are used to measure a theoretical concept and how the units of inquiry are to be mapped to the variables' values (Diekmann, 2007, p. 239; Schnell

[14] Note that the term *accuracy* in this section is used in its general meaning of being close (however defined) to the true value. *Accuracy* here does not refer to the specific performance metric presented in Section 1.1.1.3.

et al., 2018, p. 113-114). Measurement is the actual process of quantification in which the observational units are assigned to a variable's values (Diekmann, 2007, p. 239; Schnell et al., 2018, p. 113-114).[15]

In a situation in which a researcher follows this deductive process and has a clear conceptualization and definition of a concept, supervised learning can be used in the measurement process. For example: As has been established at the beginning of this introductory chapter, text data are a highly important data source for political scientists because they are habitually produced in a large range of political processes and thus offer the opportunity to unobtrusively observe these processes. However, a manual evaluation of text data is very resource intensive and therefore not applicable in many studies—unless a part of the measurement process can be automated. Supervised machine learning exactly does this. Starting from a set of labeled training data that defines how textual units represented by $\boldsymbol{x}$ are to be assigned to the values of y, a supervised learning algorithm can be trained that then assigns unlabeled textual units as accurately as possible to the values of the variable. In general, the training data thus are observational units for which the assignments to values of a variable, that serves as an indicator of the concept under study, are known. The training data set encodes the measurement instructions specified in the operationalization. The learning algorithm then learns these measurement instructions from the training data to then apply the learned mapping to yet unlabeled data.

Thus, whilst supervised learning itself is an inductive procedure (the function $\hat{f}$ is learned from training examples), supervised learning can play a role in the deductive top-down process that moves from concept specification to operationalization to measurement. Therefore, supervised learning techniques can be applied as measurement instruments for the measurement of already conceptualized concepts.

In every measurement process, political scientists aim to have a reliable and valid measurement instrument. Reliability can be understood as the degree to which the measurement instrument—when repeatedly applied to the same unchanged object—will yield the same result (Schnell et al., 2018, p. 132).[16] Validity is the degree to which the measurement instrument provides a measure of the concept that it is designed to measure (Schnell et al., 2018, p. 135).

When a supervised learning algorithm serves as a measurement instrument, reliability increases as the algorithm's variance decreases.[17] A learning approach whose predictions for

[15] A more precise definition of measurement arises "if one understands measurement to be a structure-conformable mapping of a set of objects to a set of numbers. Structure-conformable means that the relations existing between the objects are 'reflected' in [the relations between] the set of numbers" (Diekmann, 2007, p. 279, translated from German). Thus, when here it is said that observational units are assigned to a variable's values, it is assumed that the variable allows for a structure-conformable mapping.

[16] More precisely, reliability is defined as the variance of the true values divided by the variance of the measured values (Schnell et al., 2018, p. 132-133). It is that part of the variance of the measured values that is due to the variance of the true values (Jackman, 2008, p. 123)

[17] Note that here the supervised learning approach and not the already trained model is considered to be the measurement instrument.

test data points are highly consistent in the sense that they do not vary much over different training set compositions (low variance) constitutes a reliable measurement instrument. For a supervised algorithm to be a valid measure, its bias has to be low: The algorithm's predictions for test data points have to be close to their true values. Moreover, for a measurement instrument to be valid, it also has to be reliable: A learning approach that produces very different predictions for the same test data point not only is unreliable but also cannot be valid (Schnell et al., 2018, p. 136).[18]

When working with text data, humans are typically regarded as the ultimate provider of validity (Benoit, 2020, p. 470; Song et al., 2020, p. 551). Humans are seen best equipped for understanding and decoding the meaning of text and thus are seen as the best tools for making conceptual judgments (e.g. deciding whether the sentiment expressed in a text toward a specific entity is positive or negative, or deciding whether a text does or does not fall into a specific conceptual category) (Krippendorff & Craggs, 2016, p. 181; Song et al., 2020, p. 551). These human conceptual judgments can encode the meaning of text by assigning textual units d_i to the values of the output variable y, such that one has $(d_i, y_i)_{i=1}^{N}$ (Krippendorff & Craggs, 2016, p. 181). Consequently, the validity of a text-based supervised learning approach is usually evaluated by assessing how accurately the trained model's predictions can replicate human coding (Grimmer & Stewart, 2013, p. 279).

As several studies show that humans are not necessarily reliable when coding text data (e.g. Mikhaylov et al., 2012; Ennser-Jedenastik & Meyer, 2018), the question arises to what extent human assessments really can be considered valid (Song et al., 2020, p. 553). Research on how the reliability of text-based human coding—and thus the potential for validity—can be increased or established (Hayes & Krippendorff, 2007; Krippendorff, 2013; Pilny et al., 2019) is important but not the subject of this dissertation. In this dissertation, it is assumed—in consistency with the literature at large (see e.g. Benoit, 2020; Nelson et al., 2021)—that the best available way to assess the validity of a text-based measurement instrument is to compare the results of the measurement instrument with human judgments. Hence, a trained model's prediction performance on a yet unused set of test data points for which human codings are available is taken as an indicator of a supervised model's validity. The lower a model's expected generalization error, the higher its validity.

The important point to understand here is this: The political scientist who uses a supervised learning method for the purpose of obtaining a valid measure of a concept she is interested in and the machine learning researcher who (just) wants a well-generalizing model that makes accurate predictions for a given task both want the same thing: a low expected generalization error. Political scientists are actually and ultimately interested in explanation and causal inference. Yet making causal inferences implies translating concepts into measurable indicators. And if political scientists make use of supervised learning techniques in the measurement process, then—in this very instant, for this very purpose—the goals of political scientists coincide with the goals of machine learning researchers: the aim is to reach an as high as possible prediction performance.

[18] For a similar statement related to the variance and bias of an estimator see Jackman (2008, p. 121-123).

Hence, in the context of applying a supervised learning method to get a valid measure for an a priori-defined concept, the goal is *not* causal inference but prediction. This constitutes a major deviation from the mindset political scientists are trained to have. Following their trained ways of thinking, some political scientists advise researchers to "prefer model specification based on theory and isolating the effect of specific explanatory factors" (Benoit, 2020, p. 490) when working with text data and call for the preference of less complex, interpretable models over more complex models even if this implies that prediction accuracy is reduced (Benoit, 2020, p. 490). Yet when using supervised learning on text data for the purpose of measurement, the aim is not to understand or identify the causal links that map from textual elements to the values of the output variable. (For example, the aim is not to make causal inferences about whether and in how far the occurrence of the term *'good'* causes a text to express positive sentiment.[19]) The goal merely is to learn and to imitate this mapping as closely as possible to make as accurate as possible predictions on new data points (Grimmer & Stewart, 2013, p. 270, 279).

This dissertation—and especially the article *Introduction to neural transfer learning with Transformers for social science text analysis*—seeks to contribute to this goal. It seeks to contribute to the research endeavor of increasing the accuracy with which already defined concepts are measured via supervised learning techniques from text data in political science. The more accurately human codings for yet unused text data units can be emulated by a supervised model, the higher the indicated validity of the model that one employs as a measurement instrument. Thus, one could also say that the goal of this dissertation is to increase the validity of text-based measurements in political science studies.

The articles in this dissertation apply supervised learning techniques on text data with a focus to obtain as accurate as possible measures for a priori-defined concepts. When political scientists use supervised learning on text data for the purpose of measurement as part of a scientific research project, there are numerous issues regarding the data generating process, inference, and analysis that need to be considered. These issues will be outlined in the following.

Observing Textual Expressions. Typically, the concepts that political scientists seek to measure from texts are latent characteristics of the texts (Egami et al., 2018, p. 4). And frequently, the concept of interest is a latent characteristic of the (political) actor that has generated the text (e.g. the topics political parties focus on in their election manifestos, the sentiment a citizen expresses in a tweet, or the ideological position a legislator takes in a set of speeches) (Egami et al., 2018, p. 4; Benoit, 2020, p. 465-466).

The first important point to note here is that textual data only allow making inferences on the basis of what has been expressed in text, and what is expressed in textual form

[19]Note that there are studies that aim at making such inferences about language use (e.g. Slapin & Kirkland, 2020). In such cases, it is indeed important to have a model that allows one to identify the effects of single linguistic features. Studies that seek to make inferences about language are a separate category from the studies that are discussed here that use supervised machine learning on text data for the purpose of measurement.

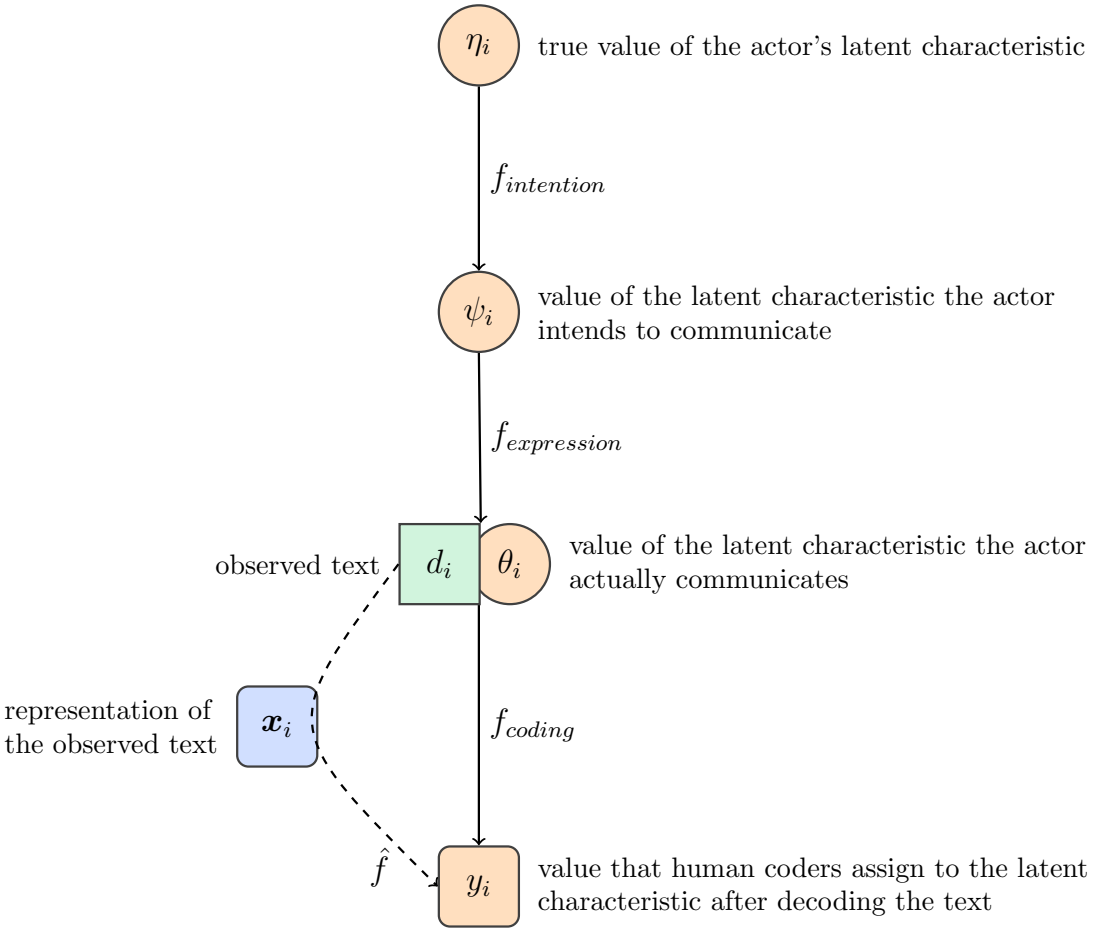


Figure 1.1: **Process of Text Generation and Supervised Learning.** This figure describes the stochastic process of text generation and supervised learning. η_i is the true value of the ith actor's latent characteristic. By some process (here indicated by function $f_{intention}$), the true value is mapped to the value of the latent characteristic that the actor intends to communicate, denoted by ψ_i. By a further process (indicated by function $f_{expression}$), the intended value ψ_i is mapped to the value of the latent characteristic the actor actually communicates (which is θ_i) by means of producing the sequence of words and symbols d_i. The observed text d_i is read by human coders that decode the text and assign value y_i to the latent characteristic (f_{coding}). Given a data set, $(d_i, y_i)_{i=1}^N$, a supervised machine learning algorithm then can be applied to learn the mapping from observed text data $(d_i)_{i=1}^N$ to values $(y_i)_{i=1}^N$. Here, each d_i is converted to some representational form (denoted with $\boldsymbol{x}_i$).

need not necessarily correspond with the true value of the latent characteristic under study (Benoit et al., 2009, p. 497-499): Assume that the true attitude of a legislator toward a policy issue is η_i. The legislator then may make strategic considerations or may take into account social norms to then form an attitude position ψ_i that he *intends* to communicate (Benoit et al., 2009, p. 497) (see Figure 1.1 that illustrates this process). ψ_i is likely to be based on η_i. Yet the degree to which the intended ψ_i is informed by the true η_i can vary from very strong to weak. Having formed an attitude position he intends to express, the legislator then can go about expressing it. But how far ψ_i corresponds with the latent attitude position θ_i that he *actually expresses* by means of using natural language also depends on various factors (Benoit et al., 2009, p. 497). First, how closely θ_i matches ψ_i will depend on the legislator's general linguistic ability to express his intended position. The legislator seeks to communicate ψ_i, but the words the legislator uses may convey a slightly different position θ_i. Here, technical factors (such as length restrictions on tweets or time limits on parliamentary speeches) may also come into play. Contextual factors can also have an effect. For example, in an attempt to respond to what other actors in the political discourse have stated so far, the legislator may put more emphasis on some aspects and less on others, thereby possibly distorting what is expressed from the actually intended ψ_i. And then, even if the same legislator with the same level of linguistic ability tried to express the same intended position ψ_i via the same communication channel in the same context a second time, a third time, and a fourth time, it is likely that the statement he would make using natural language slightly differed each time because the usage of natural language can be regarded as an inherently random process (Manning & Schütze, 1999, p. 15; Benoit et al., 2009, p. 497).

Let $d_i = (a_1, \ldots, a_t, \ldots a_{T_i})$ be the sequence of words and symbols the legislator uses.[20] η_i is mapped to d_i by some stochastic text generation process and the latent position that the text d_i actually communicates is θ_i. Whereas η_i, ψ_i, and θ_i are latent quantities, d_i constitutes manifest data. Researchers thus only observe d_i and d_i only conveys the latent position θ_i. Hence, researchers—without making additional assumptions—can only infer θ_i from d_i.

This is true beyond the example given. Observing d_i provides information on the latent characteristic θ_i as expressed in text. The text's latent property θ_i, however, may or may not correspond with the political actor's latent characteristics, ψ_i and η_i, that researchers are commonly interested in (Benoit et al., 2009, p. 499). For example, the topic proportions a political actor expresses in text do not necessarily have to closely resemble the political actor's true issue emphasis. Estimating θ_i from text thus does not allow researchers to make inferences about ψ_i or η_i—unless additional assumptions are imposed (Benoit et al., 2009, p. 499), for example, that the actor intends to communicate his true characteristic and is perfectly able to do so such that $\eta_i = \psi_i = \theta_i$.

[20] d_i either already is text or, if the legislator made a verbal statement, the verbal expression is assumed to be mapped without error into textual form.

Selection Mechanisms. A second point that political scientists should be aware of when using text data is that the text data generating process and the text data collection process are both subject to numerous selection mechanisms.[21] Selection mechanisms that operate during the text generation process and selection mechanisms that play a role during text data collection can cause selection biases. Selection biases occur (1) if the mechanism that selects observational units for an analysis from a larger population of units for which inferences are to be made is correlated with the units' values on the dependent variable, or (2) if the assignment of units to the values of the explanatory variables is correlated with the units' values on the dependent variable (King et al., 1994, p. 115-116, 124-125).[22] Here, the focus is on the first mentioned type of selection bias. Thus, the following discussion focuses on biases that arise when the question of whether or not a text is selected into the sample of units to be analyzed correlates with a property of the text (or the text's creator) that serves as the outcome variable.[23] First, it will be illustrated how biases can arise from the text data generating process itself, and then it will be explicated how the researchers' chosen data collection strategy can cause selection biases.

Selection biases caused by the text data generating process. Assume that an individual has a true attitude position η_i toward a policy issue, but she does not form the intention to communicate her position. The attitude may be toward a policy issue that is of little importance to the individual, or the individual may decide that it would be strategically best not to convey anything about η_i. In another scenario, assume that the individual has a true attitude position η_i and there is also an intended position ψ_i she might communicate, but she does not come across a situation in which she does communicate something about her attitude toward the policy. Finally, assume that the individual has a true position η_i, intends to reveal ψ_i and indeed does express some latent position θ_i, but she only expresses θ_i in private conversations with friends. In the first two scenarios, no data would be generated that researchers could observe. In the third setting, no data would be generated that is recorded and available to researchers.

The thus far described selection mechanisms of text generation have the effect that only a subset of existing latent positions is expressed in accessible textual format. If researchers seek to make descriptive inferences about the distribution of true attitude positions η_i toward a policy issue in a population by observing attitude expressions in a corpus of text data, then the inferences drawn can be biased if the selection mechanisms of text

[21] Parts of the content on pages 60 to 67 has first been published by Springer Nature in Wankmüller, S. (2022). A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis. *Journal of Computational Social Science*, (p. 1–73). https://doi.org/10.1007/s42001-022-00191-7 and here is reproduced with permission from Springer Nature.

[22] Selection biases also occur if the selection rule or the assignment rule are correlated with the size of the causal effect that units will experience (King et al., 1994, p. 138-139).

[23] If a study seeks to make descriptive rather than causal inferences, a selection bias is produced if the rule for selecting observational units for analysis is correlated with *the* variable of interest (King et al., 1994, p. 135). The following discussion points out sources for selection bias using illustrative studies whose goal is descriptive rather than causal inference. Accordingly, when referring to the definition of selection bias, here the expression *'the outcome variable'* rather than *'the dependent variable'* will be used.

generation are such that the value of true attitude position η_i correlates with the question of whether a text is produced at all or is accessible at all (King et al., 1994, p. 132). Such systematic forms of (self-)censorship may occur in autocratic regimes. Such a situation, however, also may occur if individuals that have a supportive position toward a given policy are more likely to publicly state their position toward the policy, whereas individuals with an opposing position toward the policy are more likely to remain silent (possibly because their stakes are less high). Thus, when working with text data, researchers should be aware that bias can arise if the availability of text data is correlated with the outcome variable of interest (King et al., 1994, p. 132, 135).

Selection biases caused by data collection strategies. Besides such selection biases that are induced by the text data generating process itself, selection biases also can be caused by researchers' data collection strategies (King et al., 1994, p. 132, 135). There are two sources of bias to be mentioned here. First, the type of text data, that a researcher chooses to use for analysis, causes the selection mechanisms of the text data generating process to come into effect. For example: Assume that a political scientist seeks to estimate the distribution of positions toward a policy issue in parliament and chooses to use parliamentary speeches debating the issue as her text data source. By deciding to use speech data, the speech generation-related selection mechanisms come into effect. These selection mechanisms influence (1) which legislators are allowed to speak on the issue at all (and hence for which legislators a text d_i can be observed at all) and (2) what the content of observed speeches is (this is, which intended position ψ_i each legislator tries to convey) (Proksch & Slapin, 2012, p. 520). If the speeches are given in an institutional setting in which party unity plays an important role, legislators whose position is more distant from their party leadership are less likely to be given the opportunity to speak (Proksch & Slapin, 2012, p. 526-527, 533-534). Moreover, the position ψ_i that a legislator intends to communicate when giving a speech, is more likely to be adjusted such that it is closer to the position of the party leadership (Proksch & Slapin, 2012, p. 523, 526-527). In this case, the political scientist that uses the observable speech data would estimate an incomplete distribution of policy positions and would underestimate the variance in policy positions.

A second prominent example is the usage of text data from social media platforms in research projects that aim at making inferences about a population that stretches beyond social media users. If a team of researchers aims at estimating the distribution of attitudes toward a political candidate among the voters in a country and they decide to do so by observing attitude expressions in tweets, then the researchers cannot observe text data for individuals in the population that do not use Twitter. Thus, by selecting tweets as a data source, the selection mechanisms for this very data source (Who is using Twitter? Who actively posts a tweet that expresses an attitude toward the candidate on Twitter?) are kicking in. And if across the population an individual's attitude toward the political candidate correlates with the question of whether the individual expresses the attitude toward the candidate on Twitter, then estimates on the observed Twitter data are biased.

Social Media. For political scientists, social media have several benefits as a data source: Data from social media platforms allow political scientists to observe ordinary individuals in an unobtrusive manner (Barberá & Steinert-Threlkeld, 2020, p. 405). There are two aspects here. First, social media data enable political scientists to not only study political elites (on which data is usually available in some form or another) but also allow studying voters and citizens. Second, voters or citizens can be studied without having to conduct an expensive, time-consuming survey, which has the added disadvantage of not being unobtrusive.

As soon as a researcher has acquired the skills of web data collection and has been granted access to the data by the platform operators, data collection is fast and easy (Munzert et al., 2015). Moreover, data collection can be conducted in real-time. Phenomena of interest (for example protest movements) can be observed whilst they emerge, go on, and perhaps finally cease (Barberá & Steinert-Threlkeld, 2020, p. 406). This is advantageous compared to a situation in which a researcher has to collect data afterward and has to deal with, for example, recall biases that can occur if survey participants try to recollect past events.

A related advantage of social media data is that they allow for a temporally and spatially more differentiated measurement (Barberá & Steinert-Threlkeld, 2020, p. 406). For example, at the beginning of an election campaign, parties produce election manifestos to lay out their political agenda and to state their positions on policy issues. Election manifestos, however, provide no information on how this agenda changes over the course of the campaign and later over the course of a legislative period. An election manifesto also provides no information regarding how issue emphasis and ideological positions vary among candidates that belong to the same party. Statements that individual political candidates make in posts on social media platforms constitute data that allow for such temporally more fine-grained analyses and also allow for analyses of spatial within-party variation.[a] Similarly, geolocated social media posts of citizens enable researchers to conduct a temporally and spatially more differentiated study of public opinion than can be achieved with survey research alone (Beauchamp, 2017).

Social media, moreover, not only provide more detailed information, but they also are likely to provide more immediate information (Ceron et al., 2014, p. 344). As social media posts are public, social norms and the social desirability of beliefs, attitudes, and behaviors are likely to affect citizens and politicians when communicating their views via social media (Barberá & Steinert-Threlkeld, 2020, p. 405).

[a] I am grateful to Paul W. Thurner for pointing this out to me.

Social Media (cont.). Yet, because expressions of beliefs and attitudes on social media are likely to be more spontaneous and less constrained by formal institutional rules, there might be a chance that on social media the position θ_i that an individual actually expresses in text on average is a bit closer to the individual's true position η_i. Politicians, for example, can use social media to state their views directly and spontaneously (instead of presenting their position in a deliberate, polished speech in a probably formal setting). The low-threshold, low-cost nature of social media furthermore produces immediate information on the importance of topics: Politicians and citizens can freely choose the topic they want to address in a social media post. Legislators that want to emphasize a topic are not dependent on whether they get a chance to speak in parliament or a media outlet on the issue. And an individual that cares about a specific topic can express her concern instantly without having to engage in time-consuming (non-)electoral forms of political participation—and without having to wait until a group of researchers asks her to participate in a survey in which (because in the meantime the researchers have realized that this could possibly be an important political topic) a question on the topic is included.

Finally, the spread and usage of social media led to the emergence of new channels for political information and political communication. It led to the creation of new forms of political interaction. Hence, social media constitute a new field for political research in their own right. Social media thus are not only a valuable data source for political scientists, but they are *the* data source for the observation of new, social media-related political phenomena. Moreover, because social media also transform political processes, political scientists started to study the effects of social media on the political sphere (Barberá & Steinert-Threlkeld, 2020, p. 411).

A major problem of social media as a data source for making inferences about populations of ordinary citizens is selection.[a] Both the question of who uses social media at all and the question of who makes political statements on social media are subject to selection processes. Instead of the ideal setting in which researchers select individuals from the population of interest through a randomized process, individuals at their own behest choose to use or not to use social media. Then again, among all those individuals that use a particular social media platform, selection processes operate that cause some individuals to express their political position on a particular topic and cause others not to do so.

[a] If the aim is to make inferences about political elites, selection mechanisms are less likely to be problematic, as a large share of political elites has social media profiles (Barberá & Steinert-Threlkeld, 2020, p. 406).

Social Media (cont.). Hence, inferences beyond the set of social media users whose posts have been analyzed in a study thus should be drawn with utmost care. If the selection mechanisms and a study's research design are such that the outcome variable of interest correlates with the question of whether an individual uses or not uses a specific social media platform and expresses or not expresses a position on a particular topic during a given time period and thus is selected into the study or not, then a selection bias occurs (Barberá & Steinert-Threlkeld, 2020, p. 406-407).

The second source for selection biases that is related to the data collection strategy is particularly common when working with text data. The origin of this potential source of bias is that the population of observational units on the basis of which inferences are to be made cannot be clearly determined (King et al., 1994, p. 125). Assume that a political scientist is interested in the attitudes that are expressed toward Hillary Clinton during a given time period on Twitter. (No inferences beyond Twitter users are to be made in this case. The unit of analysis here is a single tweet.) The problem that the political scientist faces is that there is no list that comprehensively registers all tweets that communicated an attitude toward Hillary Clinton in the given time period. Thus, it is initially unclear which tweets from the entire stream of tweets that have been produced during the time period belong to the population of tweets that is of interest to the study. Therefore, the scientist in a first analytical step has to identify the tweets that belong to this population. If in this first analytical step the question of whether the political scientist considers a tweet to belong vs. not belong to the population is related to the value on the outcome variable of interest (here the expressed attitude toward Hillary Clinton), then a bias will be induced.

Assume that in order to address this task of separating the relevant tweets, that are part of the population, from the irrelevant tweets, that are not, the political scientist uses a set of keywords. A tweet that contains any of the keywords is considered relevant and then is used as an observational unit in subsequent analytical steps for the estimation of attitudes. A tweet that does not contain any of the keywords is not considered relevant and is not used for the analysis of attitudes. Suppose that the set of used keywords is: *'Hillary'*, *'Hillary Clinton'*, and *'@HillaryClinton'*. The problem with this list of keywords is that it is incomplete. The list, for instance, fails to identify tweets that contain derogative terms that are sometimes used to refer to Hillary Clinton, e.g. *'Nasty woman'* or *'Hilliary'*. The incomplete keyword list will consequently miss a set of tweets that are likely to express a negative attitude toward Hillary Clinton. Hence, in the given example, there is likely to be a correlation between the attitude that a tweet expresses and the question of whether a tweet is considered to be part or not part of the population of tweets that will be analyzed. Therefore, the estimation of the distribution of expressed attitudes on the basis of the set of tweets that are considered to be relevant by the keyword list is likely to be biased.

To keep these biasing effects as minimal as possible, it is very important that researchers identify as accurately as possible the population of documents that they are interested in.

It is essential that they separate as accurately as possible relevant documents that belong to the population from all other documents that do not. Separating as accurately as possible is no guarantee that selection bias will not occur. Yet the smaller the share of documents that belong to the population of interest but are not identified as being relevant, the smaller the capacity for strongly biasing effects due to the missing out of relevant documents. Or to put it another way: The higher recall, the smaller the size the bias due to false negatives can maximally assume. (For a more detailed description of this relationship between recall and the potential size of bias see Figure 1.2.) Moreover, the smaller the share of documents that are considered to be relevant although they do not belong to the population of interest (this is, the higher precision), the more an analysis is based on documents for which the analysis actually wants to make inferences. Consequently, the higher precision, the less the degree to which estimated values can be biased by documents that are not actually of interest.

This dissertation's article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* seeks to contribute to addressing this task of as accurately as possible identifying the population of documents that are of interest in an analysis. When conducting this first analytical step of trying to retrieve the share of documents that are relevant for the analysis at hand from a corpus of otherwise irrelevant documents, then the share of relevant documents is typically very small compared to the share of irrelevant documents that make up the rest of a usually very large corpus. Thus, researchers often face an imbalanced classification problem when seeking to identify the population of relevant documents. So far, political scientists usually approach this imbalanced classification task by applying a list of keywords. This is a fast, easy, and inexpensive procedure. Yet research indicates that the human creation of keyword lists is unreliable and incomplete (King et al., 2017, p. 973-975). Hence, as has been illustrated above, there is the risk of selection bias. Addressing this issue, the article presents and evaluates methods for the retrieval of relevant documents that are more complex and more expensive than keyword lists but have the potential to yield higher retrieval performances and thus might reduce the potential size of selection bias. The evaluated methods are (1) query expansion techniques from the field of information retrieval, (2) topic model-based

[a]The effect of precision on the size and direction of bias likely depends on the (distribution of) characteristics of the truly irrelevant documents that are erroneously predicted to be relevant vis-à-vis the (distribution of) characteristics of the truly relevant documents as well as the processing of these characteristics by the learning algorithm.

[b]Thus, the simulation can also be understood as a sampling from the set of 1,000 truly relevant documents, where the proportions sampled from positive vs. negative attitude expressing documents can differ from the proportions of positive vs. negative documents in the population of 1,000 truly relevant documents. Many thanks to Christian Heumann for pointing this out to me.

[c]Note that the relationship between recall and the size of this form of bias also holds if the true proportion of positive vs. negative documents is different from 1:1. If among the truly relevant documents there are substantively more positive than negative documents (or the other way around), the precise functional form of this relationship between recall of positive documents, recall of negative documents, and bias is different than the function in the presented plot, but the general relationship between recall and bias remains the same.

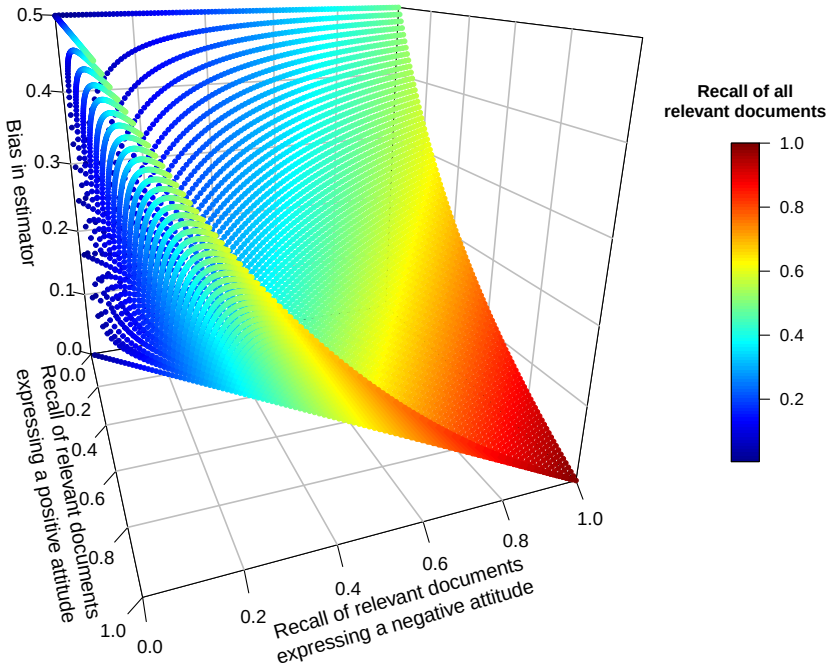


Figure 1.2: **Recall and the Maximum Size of Bias.** This plot is generated from a simulation that assumes the following scenario: Among a large corpus of documents, 1,000 documents are relevant for an analysis. Among these 1,000 relevant documents, 500 documents express a positive attitude toward a political candidate and 500 documents express a negative attitude toward the candidate. The true attitude value of all positive attitude expressing documents is 1 and the true value of all negative documents is 0. Hence, the true mean attitude value in this population of documents is 0.5. Now it is assumed that researchers in a study first apply a selection rule via which they try to identify the relevant documents from the corpus. In a second step, the researchers then compute an estimate for the mean attitude value based on those documents that the selection rule identified as being relevant. These two steps are repeated several times, each time applying a different selection rule. The question addressed in this simulation is the effect that recall (i.e. the share of the 1,000 truly relevant documents that a selection rule correctly predicts to be relevant) has on the size of bias in the estimation of quantities (here the mean attitude value) from documents that are predicted to be relevant. In order to examine the effect of recall on bias in isolation from other possible biasing effects, that can arise if truly irrelevant documents are erroneously predicted to be relevant, the assumption here is that for all selection rules precision is 1 (such that actually irrelevant documents are not selected into the study).[a] [b] Furthermore, it is assumed that the researchers are perfectly able to determine the true attitude value of a document. For example, if a selection rule identifies 50 positive and 200 negative attitude expressing documents, then the researchers will conclude that the 50 positive documents have an attitude value of 1 and the 200 negative documents have an attitude value of 0 and hence they estimate the mean attitude value to be 0.2. The difference between such an estimated value and the true mean value of 0.5 here is called bias. The plot shows how this bias depends upon the recall of relevant documents that express a positive attitude (x-axis), the recall of relevant documents that express a negative attitude (y-axis), and the overall recall of relevant documents (indicated by the color of the dots). The plot demonstrates that an increase in overall recall does not necessarily imply that the bias in the estimator decreases. An increase in the overall recall rate can even mean that the bias increases. Note that selection bias arises if the recall of positive relevant documents is higher or lower than the recall rate of negative relevant documents. If an overall increase in recall implies that this imbalance in the recall rates increases further, then an increase in recall causes an increase in bias. Yet the size that this bias can maximally assume decreases with an increasing overall recall: As the color of the dots moves from blue (low overall recall) to red (high overall recall), the maximum magnitude that this bias can reach decreases. If a selection strategy yields high overall recall, selection bias still can occur if the recall rates vary with the value of the outcome variable of interest. But the higher recall, the less strong this biasing effect can be.[c] The footnotes a to c are given on the previous page. This figure first has been published by Springer Nature in Wankmüller, S. (2022). A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis. *Journal of Computational Social Science*, (p. 1–73). https://doi.org/10.1007/s42001-022-00191-7

classification rules that have been developed in communication science, and (3) the machine learning techniques of active and passive supervised learning. The results show that whereas query expansion techniques and topic model-based classification rules do not tend to improve upon lists of empirically predictive keywords, active learning—if applied with a not too small labeling budget—yields considerably higher retrieval performances than keyword lists. Hence, the article points out that active learning is a method that political scientists could use to approach retrieval tasks better than they do so far. This is important because a more accurate retrieval method reduces the maximal magnitude of the selection bias that can be caused by the very process of separating documents that belong to the population of interest from those that do not.[24]

Interpretability. Although the utilization of supervised learning for prediction purposes discussed here does not aim at causal inference, it nevertheless aims at "making inferences that go beyond the particular observations collected" (King et al., 1994, p. 8). The aim in supervised learning is to approximate the systematic relationship that f describes (see again Equation 1.5) in order to make generalizing inferences beyond the training data points for yet unused data points. The aim, however, is not to learn about the causal effects underlying the data generating process that has produced the outputs $\boldsymbol{y}$ from $\boldsymbol{X}$. The aim is to *imitate* the systematic relationship f for the purpose of generalization without necessarily *understanding* f.

This is not to say that it might not be important to use tools for interpreting a supervised machine learning method. To investigate why a model made the predictions it has made, to examine what a model has learned, or to assess the importance of specific features for the model's predictions are important issues that can increase a researcher's trust in the applied model (Doshi-Velez & Kim, 2017, p. 2; Molnar, 2022, ch. 3.1). Interpretability tools, for example, can increase a researcher's confidence that the model picks up the systematic relationship between inputs $\boldsymbol{X}$ and outputs $\boldsymbol{y}$ and consequently does react to meaningful perturbations in the data but does not react strongly to non-meaningful perturbations (Doshi-Velez & Kim, 2017, p. 2; Ribeiro et al., 2020; Molnar, 2022, ch. 3.1). A general introduction to interpretability in machine learning is provided by Molnar (2022). An overview of approaches for the interpretation of deep neural networks is given by Belinkov & Glass (2019).

Traceability and Replicability. If supervised learning is used in the context of scientific research, then all the conducted processes involved in training and applying a supervised learning technique have to be explicitly and publicly reported such that the processes can be fully traced and the analysis is, as far as possible, reproducible (King et al., 1994, p. 26).

[24]Note that even if a method were perfectly able to separate documents that belong to the population of interest from those that do not and thus one would have a complete census of observational units, selection biases could still arise from selection mechanisms of the data generating process as described above. Therefore, the article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* makes an important contribution, but its focus is only on one out of many sources for selection bias.

This might concern, for example, the procedures used in collecting the data; rules for human coding; the separation into training, validation, and test sets; the type of learning technique used; the procedure used for hyperparameter tuning; and the settings in the optimization process. If research procedures and results cannot be intersubjectively traced, they cannot be considered scientific (King et al., 1994, p. 8, 26-27). Here, for supervised learning, the same rules and standards apply as in studies that aim at estimating causal effects.

With regard to the field of NLP there is one point worth emphasizing: In NLP, it is common that large models comprising many parameters are pretrained on massive amounts of data utilizing considerable amounts of computational resources. The aim in pretraining is to get a highly general language representation model that then can be used as an input for further training on the actual target task of interest (Ruder, 2019a, p. 64) (see Section 1.2.3.8 on sequential transfer learning). To the extent that the data and procedures used in the pretraining process are public and thus pretraining is retraceable and (given certain amounts of resources and an identical computing environment) also potentially replicable, there is no reason why political scientists should not use these pretrained models as an input for the training processes on their supervised learning tasks.

In fact, the articles of this dissertation make ample use of pretrained language models whose pretraining corpora are (not entirely but in large parts) accessible (Aßenmacher & Heumann, 2020, p. 4), whose architecture and pretraining procedures are reasonably satisfactorily described (Devlin et al., 2019; Liu et al., 2019b; Beltagy et al., 2020), whose source code is available[25], and for which the resulting pretrained models are freely accessible from an open source library (Wolf et al., 2020). Yet the larger the share of pretraining data that is not accessible, the higher the extent to which pretraining procedures are not explicitly reported and the source code is not publicly disclosed, the less the criterion of traceability is met. Political scientists should refrain from applying models for which levels of traceability are low (for example the GPT-3 model by Brown et al., 2020). The same applies to models from private providers, where it is unclear which learning approaches they apply and which data they have been trained on.

1.1.1.6 Text-Based Supervised Learning in Political Science

Supervised machine learning can be applied in the process of measuring (dimensions of) an a priori-defined concept. This section is designed to provide an overview of the spectrum of concepts that are relevant to political science and that researchers have measured by means of applying supervised learning on text data. A systematic overview that focuses on the supervised learning methods used in text-based political science research articles is given in Section 1.2.4.

[25]The source code for the pretrained language representation model BERT by Devlin et al. (2019) is available at https://github.com/google-research/bert. For RoBERTa (Liu et al., 2019b) the source code is released at https://github.com/pytorch/fairseq, and for the Longformer (Beltagy et al., 2020) at https://github.com/allenai/longformer.

Attitudes. While attitudes are traditionally measured via surveys, supervised machine learning (in combination with the digitization of communication processes) has made it possible for political scientists to use new indicators for the measurement of attitudes toward political actors or issues: There are several studies that employ supervised learning tools to measure attitudes based on social media data. Ceron et al. (2014), for example, use supervised learning on tweets to predict public opinions toward leading political figures in Italy. In a later study, they use the same method to predict voting intentions during the 2012 US presidential election campaign (Ceron et al., 2015). Amador et al. (2017) apply supervised learning techniques to measure support for the Leave vs. Remain campaigns prior to the 2016 UK Brexit referendum on the basis of tweets. King et al. (2013) use a supervised learning technique to estimate the share of posts that are supportive vs. critical of the Chinese state among censored and uncensored Chinese social media posts. And Mitts (2019) trains supervised classifiers to detect support for the Islamic State on several dimensions from tweets.

Events. In further applications, the concept under study is an event, and supervised learning algorithms are trained to detect the occurrences of events from large pools of data. D'Orazio et al. (2014), for example, use supervised machine learning to detect militarized interstate disputes from newspaper articles. Zhang & Pan (2019) apply supervised learning to identify collective action events in China from social media data. Muchlinski et al. (2021) utilize supervised learning techniques to recognize reports about occurrences of electoral violence in tweets. Wu & Mebane (2021) train a supervised learning method on text and image data to detect reports on several types of election incidents in the context of the 2016 US national elections.

Concepts in Category Systems. Supervised learning algorithms can also play an important role in data collection projects. Here, trained supervised models can serve as substitutes for coders filling the entries in a database and thereby can make data collection processes more efficient or allow them to be conducted on a larger, more comprehensive scale (D'Orazio et al., 2014). Olsson et al. (2020), for example, train different supervised learning algorithms to assign news articles into the category system of the Uppsala Conflict Data Program database—a task usually done by human coders. Koh & Boey (2021) use supervised learning to classify quasi-sentences from English election manifestos into the seven major policy domain categories as well as the 57 fine-grained categories of the Manifesto Project.[26] Glavaš et al. (2017) make a similar attempt to supplant human coding in the Manifesto Project: They seek to predict the seven major policy domains for manifestos written in four different languages. Moreover, Meidinger & Aßenmacher (2021) apply supervised learning to emulate human coding on a carefully designed coding scheme for open-ended survey questions from the 2008 American National Elections Studies.

[26]The Manifesto Project (Volkens et al., 2021a) is a highly important but also a very resource-intensive political science data collection project in which coders hand-code quasi-sentences from election manifestos into a fine-grained category system. The project has been going on for decades (its origins dating back to the 1970s) and so far 4739 manifestos from 56 countries are covered (Volkens et al., 2021b, p. 2).

Taken together, these research projects indicate that supervised machine learning can be used for extensive data collection purposes, but that the accuracy of predicted assignments can vary considerably—for example as a function of the number of training data that is available for a given coding category or coding task (Koh & Boey, 2021, p. 7; Meidinger & Aßenmacher, 2021, p. 870-871).

Other. Supervised learning furthermore is applied for the measurement of a large spectrum of various other concepts: Ramey et al. (2019) make use of supervised learning to obtain measures of the personalities of US Congress members based on speech data. Barberá et al. (2021) apply supervised learning to assess the tone in newspaper articles about the US economy. Rudkowsky et al. (2018) capture the level of negativity expressed in speeches held in the Austrian parliament. Haim & Hoven (2022) measure different forms of hate speech, for example, in a data set of tweets that were written by or addressed to members of German state parliaments. And seeking to study the nature of delegation in EU legislation, Anastasopoulos & Bertelli (2020) train a supervised learner to predict for each article in EU law between 1958 and 2017 whether it grants authority to an agency or specifies constraints for the agency.

1.1.2 Unsupervised Learning

Whereas in supervised learning a researcher has to have a clear conceptualization of the output variable that is operationalized in the form of the output values $\boldsymbol{y}$, no such a priori conceptualizations and operationalizations exist in unsupervised learning. In unsupervised learning, no target variable $\boldsymbol{y}$ and only the data $\boldsymbol{X}$ are given (Bishop, 2006, p. 3). The aim is to discover, describe, and extract patterns or structures among the data (Hastie et al., 2009, p. xi).

A very common goal in unsupervised learning is to group the data instances into homogeneous clusters (clustering) (Bishop, 2006, p. 3). Clustering procedures vary widely regarding the techniques they use to detect clusters. The clusters that a technique discovers tend to mirror the characteristics of the applied clustering technique. For example, distance-based hierarchical clustering techniques and partitioning methods (e.g. K-means) tend to separate the data into clusters with spherical convex shapes (Han et al., 2012, p. 448-449). Density-based clustering methods, in contrast, operate on the density of data points within a certain region and are able to find clusters of arbitrary shape (Han et al., 2012, p. 450). Probabilistic model-based clustering procedures model the observed data points as a mixture of underlying distributions (Ahlquist & Breunig, 2012, p. 96-97; Han et al., 2012, p. 501-503). The distributions represent latent clusters that are assumed to have generated the observed data (Han et al., 2012, p. 502).

Unsupervised learning also comprises techniques to estimate the underlying distribution of the data (density estimation) and methods that project the data into a lower-dimensional continuous space (e.g. principal component analysis, factor analysis) (Bishop, 2006, p. 3,

		mode of learning	
		supervised	**unsupervised**
predicted or learned output	**discrete**	classification	clustering
	continuous	regression	projections into lower-dimensional continuous space

Table 1.2: **Categorization of Machine Learning Approaches.** This table categorizes machine learning approaches according to the mode of learning and the nature of the predicted or learned output using terminology from machine learning.

Self-Supervised Learning. An important further branch of machine learning, that plays a central role in NLP, is self-supervised learning. In self-supervised learning, the supervising signal comes from the data itself (Chollet, 2021, ch. 4.1.3). For example, in language modeling, a learning algorithm learns to predict the next word given the sequence of preceding words (Bengio et al., 2003, p. 1138). An algorithm is trained to do so by applying it to existing text data (Bengio et al., 2003, p. 1141-1142). At each prediction step, the supervising information comes from the true next word in the given text (Bengio et al., 2003, p. 1141-1142).

559-560). Thus, just as in supervised learning where predictions can be made for discrete outputs (classification) and continuous outputs (regression), unsupervised learning approaches can learn structures of a discrete (clustering) and a continuous nature (see Table 1.2).

Unsupervised Learning and Scientific Research. When applying supervised learning, a clear conceptualization of the concept under study and the encoding of the operationalization in a training data set are a requirement. Supervised learning thus is part of a deductive process in which a concept and the way it is measured are defined a priori and then objects are assigned to a variable's values accordingly (Ahlquist & Breunig, 2012, p. 94-95).

Unsupervised learning, in contrast, can be a tool in an inductive process (Ahlquist & Breunig, 2012, p. 94-95). In unsupervised learning, a set of objects is measured on a set of variables and then a structure discovering algorithm is applied to find structures in the data—and the detected structures depend on the set of objects, the set of variables, and the specific algorithm used (Ahlquist & Breunig, 2012, p. 94-95).

Unsupervised learning thus is useful in settings in which the aim is to discover a latent structure that maps and organizes the units under study, but the precise content of the latent structure is yet to be explored. As such, unsupervised learning approaches can help

to develop, refine, or even test theory-based conceptualizations: Cserekyei et al. (2017), for example, apply model-based clustering to country-year data on the energy mix of the member states of the EU. Cserekyei et al. (2017) then inspect how the detected clustering relates to the energy ladder hypothesis. Ahlquist & Breunig (2012) reveal that theoretical expectations of how countries would group into types of a well-known typology do not consistently coincide with the country clusters produced via model-based clustering. (For a similar study see Magyar, 2022.)

Among political science research articles that use unsupervised learning for latent structure estimation there are two large groups: One group of studies aims at ideal point estimation, the other at topic modeling.

Ideal Point Estimation. The first group of studies aims at the estimation of ideological positions of political actors (parties, legislators) or entire populations on latent continuous policy dimensions. In political science, this is also sometimes referred to as scaling (Grimmer & Stewart, 2013, p. 269). Ideal point estimation has a long history in political science since spatial models of politics are central to many theoretical frameworks and empirical analyses.

While there are also approaches for ideal point estimation in which the contents of the policy dimensions are defined a priori (e.g. the Wordscores method developed by Laver et al., 2003), many approaches and applications follow an inductive approach (e.g. Gabel & Huber, 2000; Slapin & Proksch, 2008; Barberá, 2015; Däubler & Benoit, 2021; Rheault & Cochrane, 2020; Herrmann & Döring, 2021). Inductive approaches start by simply assuming that there is a small finite number of latent continuous dimensions along which the policy space can be structured (Däubler & Benoit, 2017, p. 2). Then, a learning technique is applied on provided data to find "the best-fitting empirical representation of the policy space under investigation, [...] to infer latent policy dimensions" (Benoit & Laver, 2006, p. 59). What the contents of the inferred latent dimensions are, is determined a posteriori (Benoit & Laver, 2006, p. 59). Proponents of this inductive a posteriori approach argue that the substantive meaning of the latent policy dimensions can vary over space and time and thus the dimensions' content cannot, and should not, be defined a priori (Gabel & Huber, 2000, p. 96; Däubler & Benoit, 2017, p. 5-6). According to this line of argument, the application of unsupervised learning approaches is required.

The methods developed for unsupervised ideal point estimation vary regarding the types of data they use and the kinds of actors for which they seek to estimate ideological positions. Notwithstanding this variation, the methods all have in common that they describe the policy dimensions as a low-dimensional reconstruction of the input space. Most often the dimensions are latent variables that generate the observed data (see e.g. Clinton et al., 2004; Slapin & Proksch, 2008; Barberá, 2015; Däubler & Benoit, 2021).

At first, inductive approaches estimated ideal positions based on roll call votes (Poole & Rosenthal, 1985, 1991; Clinton et al., 2004; Hix et al., 2006; Hare & Poole, 2014; Bräuninger et al., 2016). Then, over time, techniques for the estimation based on other data types such

as text data (Slapin & Proksch, 2008; Proksch & Slapin, 2010; Lauderdale & Herzog, 2016; Lo et al., 2016; Rheault & Cochrane, 2020), social network data (Barberá, 2015; Barberá et al., 2015a), annotations of election manifestos (Däubler & Benoit, 2021), and parties' Wikipedia pages (Herrmann & Döring, 2021) have been added.

A central model for estimating positions on a continuous latent dimension based on text is the Wordfish model (Slapin & Proksch, 2008). If x_{iu} is the observed frequency with which unique vocabulary term z_u occurs in document d_i, the Wordfish model assumes that (Slapin & Proksch, 2008, p. 709; Grimmer & Stewart, 2013, p. 293)

$$x_{iu} \sim Poisson(\lambda_{iu}) \tag{1.20}$$

$$log(\lambda_{iu}) = \alpha_i + \xi_u + \beta_u \theta_i \tag{1.21}$$

This is, each x_{iu} is assumed to be drawn independently from a Poisson distribution with parameter λ_{iu} (Grimmer & Stewart, 2013, p. 292-293; Lowe & Benoit, 2013, p. 301). λ_{iu}, in turn, is a function of a document fixed effect α_i, term fixed effect ξ_u, term discrimination parameter β_u, and ideological position θ_i (Slapin & Proksch, 2008, p. 709). α_i controls for document length, ξ_u accounts for the fact that independent of a document's ideological position some terms are used much more frequently than other terms, and β_u is the term weight that indicates how strongly term z_u discriminates between documents on the underlying dimension (Slapin & Proksch, 2008, p. 709). The quantity of interest is θ_i that gives the position of document d_i on the continuous latent dimension (Slapin & Proksch, 2008, p. 709).[27]

Wordfish is a widely known model in political science and has been used in various applications (Klüver, 2009; Proksch & Slapin, 2009, 2010; Lauderdale & Herzog, 2016; Nanni et al., 2016; Schwarz et al., 2017). The significance of the Wordfish model, however, also is due to the fact that the ideas it encodes are reflected in many other text-based scaling models. This is especially true for its rooting in item response theory (IRT).[28] Core notions from IRT not only form the basis for Wordfish but also for many later developed scaling models (e.g. Elff, 2013; Barberá, 2015; Däubler & Benoit, 2021; Herrmann & Döring, 2021).

IRT models are measurement models that have been intensely employed and developed in the field of educational and psychological testing (Hambleton et al., 1991, p. ix). In the typical IRT setting, a set of question items that are assumed to measure a single latent variable (often: an ability) are applied to a set of subjects whose value on the latent variable

[27]The parameters α_i, ξ_u, β_u, θ_i are all unobserved and have to be estimated. In the original paper this is done via expectation maximization (Slapin & Proksch, 2008, p. 709-710). The model here is identified by setting $\alpha_1 = 0$, $mean(\theta) = 0$, and $Var(\theta) = 1$ (Slapin & Proksch, 2008, p. 710). Wordfish assumes that there is a single latent dimension, the observed term counts are conditionally independent given λ_{iu}, and that the observed word counts follow a Poisson process (Lowe & Benoit, 2013, p. 301-302). The latter of these strong, simplifying assumptions has been addressed in Lo et al. (2016). The model furthermore could be easily extended to more than one dimension (see Däubler & Benoit, 2021, p. 7).

[28]Wordfish effectively is an IRT model with a Poisson link function (Grimmer & Stewart, 2013, p. 292-293; Däubler & Benoit, 2017, p. 11).

is to be determined (Hambleton et al., 1991, p. 7-11). For each subject, a response to each item is obtained. The aim then is to infer the subjects' values on the latent variable from their observed response patterns to the question items (Moosbrugger, 2012, p. 228-229). In doing so, each individual's manifest response is modeled as a function of the individual's latent ability and item characteristics (Hambleton et al., 1991, p. 9).

When political scientists apply IRT models to text data, the documents are treated as the subjects whose latent continuous positions are to be estimated (see e.g. Däubler & Benoit, 2017, p. 8). What is considered an item and which observed responses are made use of depends on the specific scaling method. Thus far, word counts (Slapin & Proksch, 2008), coding scheme category counts (Elff, 2013; Däubler & Benoit, 2021), the (not) following of Twitter users (Barberá, 2015), and tag assignments on parties' Wikipedia pages (Herrmann & Döring, 2021) served as manifest responses.

IRT also influences the method of CBMM that is introduced in this dissertation's article *How to estimate continuous sentiments from texts using binary training data*. Here, the aim is to obtain estimates for documents' positions on a latent continuous variable using binary training data. In the chosen setting, the latent variable of interest is sentiment rather than political ideology or a policy dimension, but the CBMM method can be applied on any latent variable. In contrast to the political science methods presented so far, the primary goal of CBMM is to generate continuous estimates *based on binary training data*. Thus, the approach starts with supervised learning and makes use of a training data set that—by means of defining for each training document whether it is positioned on the positive or the negative side of the latent sentiment variable—encodes a definition of the latent dimension. The training data set in fact provides anchor points for the latent dimension. Then, a set of classifiers is trained on the training documents to learn the mapping from text data to the positive and negative sentiment anchor points. The trained classifiers now can be viewed as items that have been designed to measure a particular ability (here: sentiment orientation). Subsequently, the trained set of classifiers is applied to a set of test set documents (just as a set of items is applied to a set of subjects). Then, each classifier predicts for each test set document the probability to belong to the positive class. These predictions are the utilized manifest information on the basis of which the test set documents' latent continuous positions are estimated. In accordance with the modeling procedures in IRT, the observed predicted probability to belong to the positive class is a function of a document's latent continuous sentiment position as well as classifier characteristics.

Topic Modeling. Topic models are probabilistic model-based clustering procedures for text data. They are applied to detect a latent topic structure in a corpus of text documents. The set of topic models that are applied in political science are hierarchical mixed membership models (Blei & Lafferty, 2007, p. 18; Roberts et al., 2016, p. 988). Here, each word in each document is modeled as a mixture of topics, where each topic is a probability mass function over the terms in the vocabulary (Blei et al., 2003, p. 997-998; Blei & Lafferty, 2007, p. 17). The Latent Dirichlet Allocation (LDA) is the most basic and best-known

topic model (Blei et al., 2003; Blei, 2012).

Assume that there is a corpus of N documents in which each document d_i is observed as a sequence of T_i words, $d_i = (a_{i1}, \ldots, a_{it}, \ldots, a_{iT_i})$. Note that words are instances of unique vocabulary terms and that a vocabulary composed of U unique terms is denoted as $Z = \{z_1, \ldots, z_u, \ldots, z_U\}$. The LDA operates on a bag-of-words-based representation of documents in which each document is represented as a vector of dimensionality U in which the uth element counts the number of times that term z_u occurs in document d_i (Blei et al., 2003, p. 994, 998; Zhao et al., 2021, p. 4714). Given these observed frequencies of terms in documents, LDA estimates a latent topic structure with K topics by postulating the following data generating process (Blei et al., 2003, p. 996-997; Roberts et al., 2016, p. 988-989):

- A topic, $\boldsymbol{\beta}_k = [\beta_{k1}, \ldots, \beta_{ku}, \ldots, \beta_{kU}]$, is a probability mass function over the U terms in the vocabulary. For each term in the vocabulary, $z_u \in \{z_1, \ldots, z_U\}$, $\boldsymbol{\beta}_k$ specifies the probability of term z_u occurring in the kth topic.[29]
- Each document d_i is considered to be characterized by a distribution over topics. For each document d_i, a vector of topic proportions $\boldsymbol{\theta}_i = [\theta_{i1}, \ldots, \theta_{ik}, \ldots, \theta_{iK}]$ is drawn from a Dirichlet distribution which is defined by K-dimensional parameter vector $\boldsymbol{\alpha}$:

$$\boldsymbol{\theta}_i \sim Dir_K(\boldsymbol{\alpha}) \tag{1.22}$$

  The element θ_{ik} in vector $\boldsymbol{\theta}_i$ gives the expected proportion assigned to topic $\boldsymbol{\beta}_k$ in document d_i. Then—given a document's topic proportion vector $\boldsymbol{\theta}_i$—for each word a_{it} in document d_i, a specific topic assignment $\boldsymbol{g}_{it}$ is sampled by drawing once from a Multinomial distribution with parameter $\boldsymbol{\theta}_i$:

$$\boldsymbol{g}_{it} \sim Multinomial_K(\boldsymbol{\theta}_i) \tag{1.23}$$

  Given $\boldsymbol{g}_{it}$ (which is a K-dimensional vector indicating the topic of word a_{it} in document d_i) a specific term for word a_{it} is chosen by drawing once from a Multinomial with $\boldsymbol{\beta}_{[\boldsymbol{g}_{it}]}$:

$$\boldsymbol{w}_{it} \sim Multinomial_U(\boldsymbol{\beta}_{[\boldsymbol{g}_{it}]}) \tag{1.24}$$

  where $\boldsymbol{w}_{it}$ indicates which of the U terms in the vocabulary, word a_{it} has materialized into.

The Correlated Topic Model (CTM) (Blei & Lafferty, 2007), a well-known variant of the LDA, assumes that the documents' topic proportion vectors $\boldsymbol{\theta}_i$ are not drawn from a Dirichlet as in Equation 1.22 but are generated from a logistic normal (Blei & Lafferty, 2007, p. 20; Roberts et al., 2016, p. 991):

$$\boldsymbol{\theta}_i \sim LogisticNormal_{K-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{1.25}$$

[29] Griffiths & Steyvers (2004, p. 5229) assume that each topic's term distribution $\boldsymbol{\beta}_k$ is sampled from a Dirichlet distribution with parameter $\boldsymbol{\delta}$ of order U: $\boldsymbol{\beta}_k \sim Dir_U(\boldsymbol{\delta})$.

The logistic normal implies that

$$\boldsymbol{\eta}_i \sim Normal_{K-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{1.26}$$

and

$$\theta_{ik} = \frac{exp(\eta_{ik})}{\sum_{j=1}^{K} exp(\eta_{ij})} \quad \text{where} \quad \eta_{iK} = 0 \tag{1.27}$$

Whereas the Dirichlet distribution used in the LDA assumes near independence between topic proportions, the CTM allows for correlations between topics by means of inducing covariances via $\boldsymbol{\Sigma}$ (Blei & Lafferty, 2007, p. 19-21).[30]

The widely used Structural Topic Model (STM) (Roberts et al., 2016), in turn, extends the CTM by allowing observed document-level variables to affect the topic proportions within a document (topic prevalence) or to affect the probabilities of terms within a topic (topical content) (Roberts et al., 2016, p. 988-990). In the topic prevalence model, the vector of topic proportions $\boldsymbol{\theta}_i$ is generated from a logistic normal with parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}$ (similar to Equation 1.25) (Roberts et al., 2016, p. 991). Yet, in contrast to the CTM, the document-specific vector $\boldsymbol{\mu}_i$ is a linear function of Q-dimensional document-level vector $\boldsymbol{x}_i$ that gives the values observed for document d_i on Q variables (Roberts et al., 2016, p. 991):

$$\boldsymbol{\theta}_i \sim LogisticNormal_{K-1}(\boldsymbol{\mu}_i = \boldsymbol{\Gamma}^\top \boldsymbol{x}_i^\top, \boldsymbol{\Sigma}) \tag{1.28}$$

$\boldsymbol{\Gamma}$ is a $Q \times K-1$ matrix containing coefficients.

Note that all presented topic models estimate a latent topic structure characterized by $N \times K$ document-topic matrix $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1|\ldots|\boldsymbol{\theta}_i|\ldots|\boldsymbol{\theta}_N]^\top$ and $K \times U$ topic-term matrix $\boldsymbol{B} = [\boldsymbol{\beta}_1|\ldots|\boldsymbol{\beta}_k|\ldots|\boldsymbol{\beta}_K]^\top$. Studies that apply topic models typically use these matrices in their analyses.

Besides the LDA, the CTM, and the STM, there exists a wide spectrum of topic models that are specified as Bayesian hierarchical mixed membership models. In some cases, these models were even developed by political scientists. There is, for example, the Expressed Agenda Model (Grimmer, 2010), the Dynamic Multitopic Model (Quinn et al., 2010), the Geographic Topic Model (Eisenstein et al., 2010), and the Keyword Assisted Topic Model (Eshima et al., 2021). Furthermore, Schulze et al. (2021) present two improvements over the standard STM procedure of using an OLS regression for modeling the relationship between document-level covariates and the estimated latent topic structure: a beta regression in a frequentist framework and a Bayesian beta regression.

Neural Topic Models (NTMs) constitute another branch of topic models that make use of deep neural networks (often variational autoencoders (Kingma & Welling, 2014)) in the

[30] As the topic proportions θ_{ik} have to sum up to 1 such that $\sum_{k=1}^{K} \theta_{ik} = 1$, there is some dependence between the proportions θ_{ik} (Schulze & Wiegrebe, 2020, p. 6). However, vector $\boldsymbol{\theta}_i$ that arises from a Dirichlet with $\boldsymbol{\theta}_i \sim Dir_K(\boldsymbol{\alpha})$ is a completely neutral vector as each θ_{ik} has no influence on the *relative* sizes of the remaining proportions $(\theta_{ij})_{j=k+1}^{K}$ (Connor & Mosimann, 1969, p. 195-196, 200).

estimation of topics in order to facilitate estimation and enhance the performance and flexibility of topic models (Miao et al., 2016; Zhao et al., 2021, p. 4713). (For an overview see Zhao et al., 2021.) So far, NTMs have hardly found their way into political science. However, there are applications with social science data. For example, Card et al. (2018) apply NTMs to a corpus of newspaper articles on immigration.

Topic models can be used for exploring and describing the content and proportions of topics within corpora (Blei, 2012). In political science, topic models furthermore are often applied as measurement tools in empirical analyses (e.g. Barberá et al., 2019; Dietrich et al., 2019; Martin & McCrain, 2019; Baerg & Lowe, 2020). In such studies, the documents' estimated topic shares for one specific topic (or a small set of specific topics), serve as a measure for the dependent or an independent variable. For example, in a study that seeks to identify the causal effect of changes in the ownership of local TV stations on the coverage of local vs. national politics, Martin & McCrain (2019) apply an LDA on transcripts of the local TV stations' news broadcasts in order to create their dependent variable: the proportions assigned to local vs. national topics. Baerg & Lowe (2020) apply a topic model on meeting transcripts of the US Federal Open Market Committee to then obtain a measure for each of the committee's members' preferences based on the relative emphasis each member puts on the topics inflation vs. economic output and unemployment. Barberá et al. (2019) employ an LDA for the identification of political topics expressed in tweets by US Congress members and further Twitter users to then study the relationship between the issue attention of legislators and the issue attention of (subpopulations of) the general public over time.

The STM is frequently employed to explore the effects of document-level variables on the estimated latent topic structure (e.g. Roberts et al., 2014; Lucas et al., 2015; Kim, 2017; Bagozzi & Berliner, 2018; Blaydes et al., 2018). Lucas et al. (2015), for instance, apply an STM on texts written in Arabic by Muslim clerics to inspect the topical prevalences in texts that were written by Jihadist vs. non-Jihadist clerics. Roberts et al. (2014) examine in how far exposure to different treatment conditions affects the topical prevalence and the topical content expressed in open-ended survey questions. And Bagozzi & Berliner (2018) apply an STM on human rights reports from the US State Department in order to analyze the effects of domestic political factors as well as characteristics of bilateral relations on the topics covered in these reports.

Furthermore, topic models are also used to retrieve documents that are about topics that are relevant for an analysis from large corpora of documents that otherwise primarily cover topics that are irrelevant for the analysis at hand (Baden et al., 2020).

When inspecting these various applications of topic models closely, one finds that (political) scientists apply topic models even if they are a priori interested in a specific set of topics (e.g. Dietrich et al., 2019; Martin & McCrain, 2019; Baden et al., 2020; Baerg & Lowe, 2020) and therefore employing a supervised classifier would actually be appropriate. One reason why topic models are nevertheless applied is of a practical nature: Barberá et al. (2019, p. 889), for example, explain their use of a topic model by stating that "Despite the

existence of well-known categories of political issues, training an accurate classifier would be an incredibly arduous task, given the large number of categories, making unsupervised models a preferable option." In other cases, there is an imbalanced classification problem in which the corpus under study is immensely large whilst only a small proportion of documents fall into the relevant topic category (e.g. Baden et al., 2020). In a supervised learning setting, in both cases, it is important to have enough training data such that the supervised algorithm can learn to accurately recognize documents also from small categories. The costs of generating an adequate amount of training data thus are one reason why researchers resort to unsupervised learning approaches although they conduct a deductive process in which the definition of a concept is followed by operationalization and measurement, and hence supervised learning would be the adequate learning approach to use.

The costs of creating training data play a role in all three articles of this dissertation. Especially the article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* takes up the issues raised here by evaluating an unsupervised topic model-based approach as well as supervised approaches in the context of identifying the small share of relevant documents from larger collections of documents.

1.2 Natural Language Processing

Natural language processing (NLP) refers to the computational processing of natural language (Smith, 2011, p. xiv). Natural languages are languages that have developed naturally and are used as a means of communication by human beings (Kumar, 2011, p. 1). Natural language can be expressed via speaking, writing, or signing. In this dissertation, the focus is on the analysis of natural language in the form of text.

In the following, an overview of the varied tasks that NLP research addresses will be given (Section 1.2.1). Then, the historic development of the field will be outlined (Section 1.2.2). Afterward, an introduction to deep learning, transfer learning, and language representation learning is provided (Section 1.2.3). Finally, against the background of the presented historically informed overview of NLP methods, the techniques used in text-based political science research are mapped (Section 1.2.4).

As a start, concepts (along with notation) are introduced that are useful when treating natural language in a computational and statistical context: Texts are discrete sequential data (Smith, 2011, p. 2). They are sequences of symbols (characters, punctuation, spaces, etc.) (Smith, 2011, p. 2). These symbols—either on their own or when glued together into meaningful sequences (e.g. words)—form textual *tokens* (Manning et al., 2008, p. 22). A sequence of tokens that is an observational unit in an analysis here is called a *document*. A document thus could be, for example, a phrase, a sentence, the context of a word, a

paragraph, a text in its entirety, or even a concatenation of texts. A collection of documents is called a *corpus* (Manning et al., 2008, p. 4). A corpus with N documents is denoted as $D = (d_1, \ldots, d_i, \ldots, d_N)$. The ith document in this corpus is a sequence of tokens: $d_i = (a_1, \ldots, a_t, \ldots, a_{T_i})$. The textual token a_t is the token at the tth position within document d_i.[31] A token is an instance of a class of identical tokens, called *type* (Manning et al., 2008, p. 22). A (possibly normalized or annotated version of a) type that enters the analysis is named *term* or *feature* (Manning et al., 2008, p. 22). The set of unique features is the *vocabulary* (Manning et al., 2008, p. 6, 22). A vocabulary with U unique features is denoted as $Z = \{z_1, \ldots, z_u, \ldots, z_U\}$.

1.2.1 Natural Language Processing Tasks

NLP comprises a large spectrum of tasks and thus the types of outputs produced by NLP systems vary widely. One set of NLP tasks focuses on analyzing texts at the linguistic level. Tasks in this group, for example, are

- *sequence segmentation*: the separation of texts into segments (e.g. *tokenization*) (Smith, 2011, p. 4)
- *language modeling*: the prediction of the next textual token given a sequence of preceding tokens (Bengio et al., 2003)
- *lemmatization*: the mapping of inflectional word forms to their canonical base form, their *lemma*; *stemming*: the transformation of inflectional and derived word forms into simpler forms via heuristic algorithms (Manning et al., 2008, p. 32-33).
- *part-of-speech (POS) tagging*: the identification of each word's POS.[32]
- *syntactic parsing*: the syntactic analysis of sentences, e.g. by analyzing syntactic dependencies between tokens (*dependency parsing*) (Smith, 2011, p. 10-11)

Another set of NLP tasks tries to capture the meaning of (elements of) texts (Smith, 2011, p. 11, 13). These tasks involve an analysis at the semantic level and are considered to be *natural language understanding* tasks (MacCartney, 2014). Examples of tasks in this group are

- *word sense disambiguation*: the identification of the context-dependent meaning of a word, e.g. identifying the sense the word *'party'* has in a given text (Raganato et al., 2017)
- *coreference resolution*: the identification of which textual expressions refer to which entities, e.g. identifying which entity *'it'* refers to in the sentence *'The party group*

[31] Note that document lengths may differ as indicated by the subscript i at the last token position index T: a_{T_i}.

[32] The POS is a category of words that play similar syntactic roles, e.g. noun, verb,... (Smith, 2011, p. 5)).

submitted a bill and it was not adopted.' (Pradhan et al., 2012)

- *information extraction*: "[...] the automatic extraction of structured information [...] from unstructured sources" (Sarawagi, 2007, p. 263). Information extraction, for example, comprises the identification of named entities (*named entity recognition*), the extraction of relationships between entities (*relationship extraction*), or *event extraction* (Walker et al., 2006; Sarawagi, 2007, p. 269-271).
- *semantic parsing*: the mapping of text to a formal machine-readable representation of meaning, e.g. mapping a sentence to a graph in the *Abstract Meaning Representation* format (Knight et al., 2020)
- *text summarization*: the extensive or abstractive summarization of one or several documents (Rush et al., 2015; Nallapati et al., 2016)
- *topic modeling*: the identification of underlying themes within a corpus (Chang et al., 2009)
- *sentiment analysis*: the identification of sentiment that is expressed in text. More precisely, sentiment analysis seeks to identify which attitude holder expresses which sentiment toward which (aspect of which) entity at what point in time (Liu, 2015, p. 22-23).
- *natural language inference*: inferring whether a statement (the hypothesis) is entailed in, contradicts, or is neutral toward a provided premise (Bowman et al., 2015)
- conversing in a *dialogue*. This implies, amongst others, keeping track of the user's wishes (Henderson et al., 2014) and selecting adequate responses (Henderson et al., 2019).
- *question answering*. Question answering can take various forms. A common task—typically referred to as *reading comprehension*—is to pose a question on a provided text passage. The task for the NLP model then is to provide the text span from the passage that answers the question (Rajpurkar et al., 2016), to answer a multiple choice question relating to the passage (Lai et al., 2017), or to provide a free-form answer (Kočiský et al., 2018). Other types of question answering tasks may require the NLP system to use not provided general knowledge (Clark et al., 2018), and may require reasoning (Levesque et al., 2012), common sense (Zellers et al., 2018, 2019), and the ability to apply learned knowledge (Hendrycks et al., 2020). Moreover, question answering tasks can require considerable conversational ability: Providing the answer to a question, for example, may involve having to generate follow-up questions (Saeidi et al., 2018) or keeping track of the conversation history (Reddy et al., 2019).

The complement to natural language understanding is *natural language generation*, which means producing as an output understandable natural language text (Gatt & Krahmer, 2018, p. 68). Two groups of natural language generation tasks can be distinguished:

- In *text-to-text generation* tasks, natural language output is generated based on linguistic input (Gatt & Krahmer, 2018, p. 65). Examples are *text summarization* and *machine translation* (Gatt & Krahmer, 2018, p. 66). In machine translation, a sequence of tokens in one language is translated to another language (Smith, 2011, p. 18).
- In *data-to-text generation* tasks, natural language output is generated based on non-linguistic input (Gatt & Krahmer, 2018, p. 66). Examples are the generation of news reports from data tables (Wiseman et al., 2017) as well as *image captioning* and *video captioning* (Gatt & Krahmer, 2018, p. 66-68). The task in image or video captioning is to produce text that describes the content of images or videos (Zhou et al., 2017; Agrawal et al., 2019)

Image and video captioning also are instances of *multimodal tasks* that combine the processing of textual data with the processing of audio, video, or images. Other multimodal tasks, for example, are the answering of natural language questions referring to images (*visual question answering*) (Goyal et al., 2017) and the classification of multimodal inputs into predefined categories (as e.g. in *multimodal sentiment analysis* or *multimodal emotion recognition* (Busso et al., 2008)).

Machine learning techniques and NLP methods are also used for *information retrieval* tasks (Manning et al., 2008). Information retrieval is a field closely related to NLP. The starting point in information retrieval is a large collection of unstructured items (often a collection of documents, but it could also be a collection of images or videos) and an information need that is typically explicitly expressed in the form of a user query (Manning et al., 2008, p. xxxiv, 1, 5). The standard information retrieval task then is to rank the items according to how relevant they are to the query (Manning et al., 2008, p. 1, 16). When addressing this task, information retrieval systems usually make use of machine learning and NLP tools such as clustering, or representation learning with various neural network architectures (Manning et al., 2008; Huang et al., 2020a).

A final aspect to be mentioned here is that in the field of NLP the to be predicted output often is not a single value (i.e. a class label as in classification tasks or a real-valued scalar as in regression) but rather consists of several elements that are related to each other (Goldberg, 2016, p. 381). This is called *structured output prediction*. In dependency parsing, for example, the task is to predict an entire parse tree over the input sequence (Smith, 2011, p. 11). Another example is sequence tagging. In sequence tagging tasks, for each token in a sequence a tag is to be predicted. POS tagging is a standard example in this regard (Goldberg, 2016, p. 381). Named entity recognition (NER) can also be formulated as a sequence tagging task (Tjong Kim Sang & De Meulder, 2003). In NER, the task is to identify sub-sequences of tokens that refer to named entities and then to assign appropriate labels to the identified sub-sequences of tokens (e.g. to predict whether an identified named entity is an organization, person, or location) (Tjong Kim Sang & De Meulder, 2003; Smith, 2011, p. 11). For each token, a tag can be predicted independently as in a multi-class classification task (Lample et al., 2016, p. 261). But tags may not be independent of

each other. There are several factors (e.g. logical constraints of the task) that can cause dependencies between tags (Lample et al., 2016, p. 261). For example, predicting one token to be part of a named entity of type `<PERSON>` and then predicting the next token to belong to the same named entity but this time predicting the named entity to be of type `<LOCATION>`, would be a logical inconsistency (Lample et al., 2016, p. 261).[33]

This overview of NLP tasks is not exhaustive and many of the mentioned tasks comprise several subtasks or variants. Furthermore, time and again new tasks are created. A recently developed group of tasks, for example, seeks to identify statements and beliefs in texts that are implied rather than stated explicitly (e.g. Habernal et al., 2018; Sap et al., 2020).

1.2.2 A History of Natural Language Processing

The current state of NLP research is the product and the continuation of preceding decades of research. This section gives a historical overview of the field:

In the early phase of NLP (from the 1950s up until and including the 1980s), a majority of researchers sought to automate NLP tasks via hard-coded, hand-crafted rules and thus mirrored the symbolic approach that was taken in the larger field of AI at the time (Manning & Schütze, 1999, p. 4-5; Chollet, 2021, ch. 1.1.1).[34] In this symbolic approach to NLP researchers aimed at making machines intelligent by a priori determining the rules and mechanisms via which the machines were to process the provided inputs (Manning & Schütze, 1999, p. 4-5; Chollet, 2021, ch. 1.1.1). For example, in the famous 1954 Georgetown-IBM experiment, a team of researchers managed to automatically translate more than 60 sentences from Russian to English via a dictionary and six prespecified rules (Hutchins, 2004). The experiment stirred hopes that high-quality, automated machine translation was soon in reach (Hutchins, 2004, p. 113). These hopes, however, were not met as the system that had been tailored for the occasion did not generalize well (Hutchins, 2004, p. 113). Another strong influence in this early phase of NLP was Noam Chomsky (1957, 1965) who theorized that already at birth humans possess structural linguistic knowledge, meaning that rules for processing language exist in the human brain prior to sensory input (Manning & Schütze, 1999, p. 4-5; Chomsky, 2017). In line with this theory, research from this time can be viewed as an attempt to equip machines with these rules in order to give machines the same starting position as the human brain (Manning & Schütze, 1999, p. 5).

Then, starting at the end of the 1980s, the field of NLP underwent a substantive change with the adoption of statistical machine learning methods (Louis, 2020). Statistical NLP became the dominant approach (Manning & Schütze, 1999). The shift from symbolic to statistical NLP implied a shift from the focus on innate linguistic knowledge to a treatment

[33] On procedures for handling structured output prediction tasks see Goldberg (2016, p. 381-385) and Deshwal et al. (2019). For recent research see for example the Workshop on Structured Prediction for NLP (Kozareva et al., 2021).

[34] Manning & Schütze (1999, p. 4-5) see this phase lasting from the 1960s to the end of the 1990s.

of natural language as an empirical phenomenon whose generation underlies a probabilistic process (Manning & Schütze, 1999, p. 6). Instead of providing inputs and hand-crafted processing rules and letting the outputs be generated automatically, now supervised machine learning algorithms were fed with inputs and corresponding outputs and then learned the processing rules (Chollet, 2021). Common learning methods that were applied for supervised text classification tasks were support vector machines (SVMs), naive Bayes, logistic regression, and tree-based algorithms (Sebastiani, 2002; Manning et al., 2008). For unsupervised text clustering tasks, flat partitioning algorithms (e.g. K-means) and hierarchical clustering algorithms were used (Manning & Schütze, 1999; Liu & Croft, 2004). Moreover, the best known probabilistic topic model, the LDA (Blei et al., 2003), and its variants and extensions (e.g. Blei & Lafferty, 2007; Roberts et al., 2016) were developed. For NLP tasks that were framed as sequence tagging tasks hidden Markov models and conditional random fields (Lafferty et al., 2001) were applied (Manning & Schütze, 1999; McCallum & Li, 2003; Jin & Ho, 2009; Mitchell et al., 2013).

Many conventional machine learning methods (such as SVMs, naive Bayes, logistic regression, tree-based algorithms, or LDA) require as an input an $N \times U$ matrix $\boldsymbol{X}$ that represents each of the N instances by a single feature vector of dimensionality U. Each of the U features defines one dimension of the feature space, and the feature vector of an instance positions the instance within this space. During the training process, the parameters of a model that operates within this space are learned. The elements of the feature vectors themselves, however, are not updated during training. This implies that when applying a conventional machine learning method, researchers have to provide it with prefabricated representations.

For NLP researchers to use these conventional machine learning methods, vector space models (Salton, 1971) were developed that represent entities (e.g. documents) as vectors in feature space (Turney & Pantel, 2010, p. 141). The traditional way to create such a feature space in which documents can be represented as vectors is to represent each textual feature by a one-hot encoded vector. For example, if there were 1,000 unique features, then each feature would be represented by a 1,000-dimensional vector in which 999 elements assumed a value of 0 and one element (indicating the index of the feature) assumed a value of 1 (Pilehvar & Camacho-Collados, 2020, p. 4). The utilization of one-hot representations implies that each feature is represented as a unique dimension that is independent of the other textual features (Goldberg, 2016, p. 349). A document then is represented by a feature vector whose dimensionality corresponds to the number of textual features. For document d_i, the feature vector is denoted as $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iu}, \ldots, x_{iU}]$, where element x_{iu} is some function of the (weighted) number of times the uth textual feature appears in document d_i (Turney & Pantel, 2010, p. 143, 147). The document-feature matrix $\boldsymbol{X} = [\boldsymbol{x}_1 | \ldots | \boldsymbol{x}_i | \ldots | \boldsymbol{x}_N]^\top$, that comprises all document feature vectors, then is the input to the machine learning method.

Thus, what all these conventional machine learning methods had in common, was that these methods were provided with high-dimensional, one-hot encoded representations of

linguistic features that had to be created in a manually controlled process (Goldberg, 2016, p. 345, 349-354). Whilst processing rules were no longer explicitly coded but learned, representations of linguistic features and documents still had to be crafted.

There is a large number of diverse operations that can be used when defining textual features and when transforming a set of text documents into a set of feature vectors. Typically, several operations are implemented sequentially in a whole preprocessing pipeline. Such a preprocessing pipeline can involve (Sebastiani, 2002, p. 10-18; Turney & Pantel, 2010, p. 153 ff.; Sarkar, 2016, ch. 3; Denny & Spirling, 2018, p. 170-172)

- tokenization operations (i.e. the separation of a textual sequence into segments that build the basic textual features, e.g. characters, words, n-grams, collocations)[35]
- feature exclusion operations (e.g. the removal of punctuation, symbols, numbers, emoticons, stopwords, or the removal of terms with a very low or a very high document frequency),
- normalization operations (e.g. lowercasing, stemming, lemmatization, spelling correction, clustering of features),
- the annotation of textual features with additional information (e.g. adding POS tags, adding NER tags, including the features' sentiment values, including information from dependency parsing),
- mathematical operations on the elements x_{iu} of the document-feature matrix. Originally x_{iu} denotes the absolute frequency with which the uth textual feature is present within document d_i. Instead of recording the absolute frequency, x_{iu} could, for example, be transformed to a relative frequency by adjusting for the length of document d_i, it could merely indicate the presence (1) vs. absence (0) of a feature, or it could give the term frequency-inverse document frequency (tf-idf) (Salton & Buckley, 1988) of the uth feature in document d_i.

Which preprocessing operations are conducted in which order and are combined in which way then defines the documents' feature vector representations (Denny & Spirling, 2018). The set of features in the document-feature matrix, in turn, defines the feature space the machine learning method operates in—and thus, the set of features that are selected or created in preprocessing affect how well a supervised machine learning method performs and what an unsupervised model can learn (Goodfellow et al., 2016, p. 3). Therefore, when applying conventional algorithms, the question of which features are used and how they are created is an important and often extensive part of a research article (see e.g. Koo et al., 2008, p. 597-598).

Due to the highly varied nature of language, the document-feature matrix typically is highly sparse (Grimmer & Stewart, 2013, p. 273). This is especially the case when no major feature

[35] n-grams are features that comprise n tokens that occur in a sequence. Often, bigrams or trigrams are used.

exclusion or normalization operations are applied. Thus, during preprocessing, researchers would seek to reduce the dimensionality of the feature space (via feature exclusion and normalization) but at the same time also ensure that features that are informative for the machine learning task at hand are kept—or even additionally added, e.g by including n-grams, collocations or conducting annotation operations.

Some of the preprocessing operations can be automated (e.g. the clustering of features), but the entire process of deciding which operations are performed (and which are not), how the operations are performed, and in which order is manually guided by the researcher. In general, it can be highly difficult to predict in advance which sets of features are informative for the task at hand and which are not (Goodfellow et al., 2016, p. 3-4). Even after years of research, a community of experts, who know the texts, the task, and the machine learning algorithms they apply, may still struggle to find a set of well-performing features (see e.g. Chen & Manning, 2014, p. 740).

Besides the requirement to create prefabricated representations, the usage of a document-feature matrix in which each selected or generated feature constitutes its own separate dimension of the feature space has three further consequences: The first implication is that each feature is independent of each other feature and hence word features as *'happy'* and *'joyful'* are considered as distant from one another as *'happy'* and *'angry'* (Goldberg, 2016, p. 350-351).

The second consequence is that the dimensionality of the feature space equals the usually very large number of distinct textual features, which gives rise to generalization problems due to the curse of dimensionality (Goldberg, 2016, p. 349): The curse of dimensionality is a common problem in machine learning (Goodfellow et al., 2016, p. 152). As the number of features increases, the number of possible combinations of the features' values grows exponentially—such that in high-dimensional feature spaces there are usually many more possible combinations than training instances (Goodfellow et al., 2016, p. 152-153). In such settings in which the observed training data points are highly sparse, there is the question of how to generalize to parts of the feature space not covered by the training data (Goodfellow et al., 2016, p. 153).

The curse of dimensionality is particularly prominent in NLP. Given a vocabulary of $10,000$ unique words, a document containing 10 words (e.g. a sentence) is just one materialization out of more than 2.76×10^{33} possible 10-word combinations (for a similar example see Bengio et al., 2003, p. 1137).[36] If each term in the vocabulary is represented as a one-hot encoded feature, each document is a point in a high-dimensional feature space. Especially as the vocabulary size (i.e. the dimensionality of the space) increases, each document is likely to have a high distance from each other document (Bengio et al., 2003, p. 1137-1138). Moreover, it is likely that most of the yet unseen test documents are word sequences that differ from the word sequences included in the training data. In this case, it is very difficult

[36]Note that the order of words here is not taken into account as document-feature matrices do not capture word order information.

to generalize from what is known about the training set documents to new, yet unseen, and likely very distant documents (Bengio et al., 2003, p. 1137-1138). For example, the sentences: *'The Democratic candidate delivered a thrilling speech.'*, *'One Republican senator gave an exciting talk.'*, and *'This new governor's inaugural address was dull.'* do not share a single term. Hence, if features are treated as independent dimensions of a feature space, knowing that the first sentence expresses a positive sentiment toward a political speech and the last sentence is labeled as expressing a negative sentiment toward a political speech, provides no information on the label of the second sentence.

The third implication of document-feature matrix-based representations is that documents are represented as a multiset of textual features, meaning that a document is represented as an unordered collection of textual features, where the same textual feature can occur multiple times (Turney & Pantel, 2010, p. 147). As a multiset is also named a bag, documents are said to be represented as a bag-of-words (Turney & Pantel, 2010, p. 147). A bag-of-words representation contains information on the frequency with which textual features occur in a document but it does not encode sequential information on the order of textual tokens within a document (Turney & Pantel, 2010, p. 147).[37] By representing a document as a bag-of-words, one makes the assumption of the exchangeability of the textual features within a document (Blei et al., 2003, p. 994). In some applications (for example if the aim is to get an approximation of the topics in a document), a bag-of-words representation may be sufficient (Turney & Pantel, 2010, p. 147). Yet if the NLP task requires capturing the meaning of text as it emanates from syntactic dependency structures and context-specific semantics, the representation of a document as an unordered bag-of-words does not tend to be particularly well suited (Nakagawa et al., 2010; Socher et al., 2013).

These difficulties and inadequacies were alleviated with the introduction of (deep) neural networks to the field of NLP. This process started in the 2000s (see Bengio et al., 2003; Collobert & Weston, 2007, 2008), but the development really took off in the 2010s (major milestones and notable early applications being Mikolov et al., 2013a,b; Socher et al., 2013; Kalchbrenner et al., 2014; Kim, 2014). The move from the application of conventional machine learning methods in the era of statistical NLP to the application of deep neural networks in the era of neural NLP implies that

- features are represented as real-valued vectors embedded in a Euclidean space rather than as unique dimensions defining the feature space (Goldberg, 2016, p. 349). More formally, the uth unique textual feature z_u does not constitute a dimension of the feature space but rather is represented as a real-valued vector in a K-dimensional Euclidean space: $\boldsymbol{z}_u \in \mathbb{R}^K$ (see Figure 1.3). (As the representation vector elements are real-valued and thus are continuous, the Euclidean space in the following is referred to with the term *continuous space*.) $\boldsymbol{z}_u$ is called an embedding. For each feature z_u, the values of the elements of its embedding vector $\boldsymbol{z}_u$ are learned during the training process just like any other parameter of the model (Goldberg, 2016, p. 349).

[37] An exception is if n-grams are used as features. n-grams contain information on short token sequences.

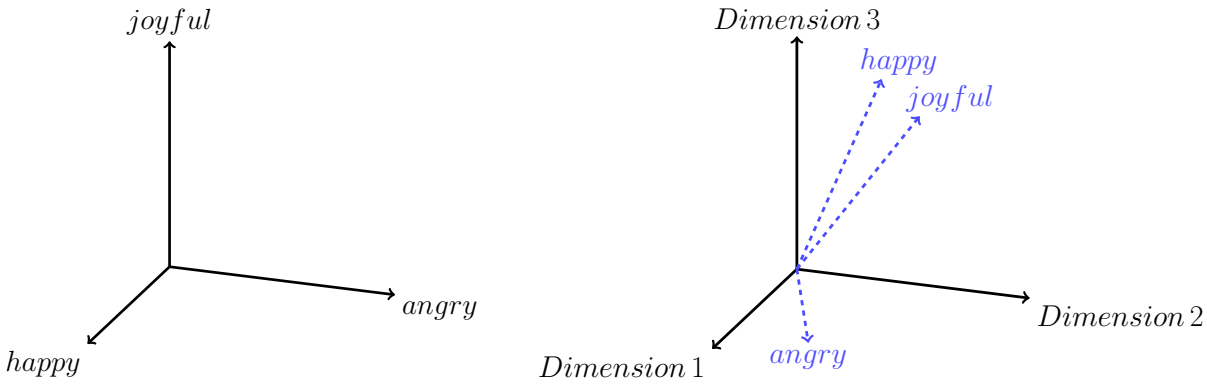


Figure 1.3: **Feature Representations.** (**3*a***) In a document-feature matrix, features are dimensions that define the feature space. In this illustrative, exemplary plot, the word features *'happy'*, *'joyful'*, and *'angry'* define a three-dimensional feature space. (**3*b***) The features *'happy'*, *'joyful'*, and *'angry'* now are represented as real-valued vectors (named embeddings) that are positioned in a three-dimensional continuous space. The embedding vectors are such that semantically similar features are close in space.

Semantically or syntactically similar textual features now are likely to have similar embeddings and thus are likely to be close in space (Mikolov et al., 2013c, p. 746). For example, the terms *'happy'* and *'joyful'* are likely to have similar embedding vectors and thus are likely to be positioned close in space. Moreover, their embedding vector representations are likely to be distant from the embedding for *'angry'*.

- the dimensionality of the feature space is defined by the dimensionality of the embedding vectors K rather than the typically much larger number U of distinct textual features that have been selected and created during preprocessing (Goldberg, 2016, p. 350-351). As usually $K << U$ and because the real-valued elements of embedding vectors are learned, representations of textual features now are low-dimensional and dense rather than high-dimensional and sparse (Goldberg, 2016, p. 350-351). The generalization to yet unseen sequences of textual features via local smoothness assumptions thus is greatly facilitated (Bengio et al., 2003, p. 1137-1140): For example, the sentences *'The Democratic candidate delivered a thrilling speech.'* and *'One Republican senator gave an exciting talk.'* do not share a single term, but they are composed of syntactically and/or semantically similar terms: *('One', 'The'); ('Democratic', 'Republican'); ('delivered', 'gave'); ('an', 'a'); ('thrilling', 'exciting'); ('speech', 'talk')* (for a similar example see Bengio et al., 2003, p. 1139-1140). As the real-valued vector representation for similar terms are likely to be close and the sentences share a similar syntactic structure, the representations of the entire sentences also are likely to be close. The function that maps from continuous sentence representations to the probability of expressing a positive vs. negative sentiment will be a smooth function over the representational space and thus a small difference in a sentence representation will cause only a small difference in the predicted probability

(Bengio et al., 2003, p. 1140). Hence, knowing that the first sentence expresses a positive sentiment toward a political speech allows generalization to yet unseen sentences that (such as the second sentence) have similar constituent features and a similar syntactic structure and thus are likely to have a similar representation (Bengio et al., 2003, p. 1137, 1140).

- representations for textual features and documents no longer have to be prefabricated in a manually guided process but are automatically learned during the training process. As explicated above, in a bag-of-words framework, the representation for a document is usually generated by the researcher in a process that involves selecting and defining core features (via tokenization, feature exclusion, and normalization operations), combining core features into more complex feature combinations (for example via adding POS tags to word features), and mathematically preprocessing the document-term matrix (for example via weighting the matrix' elements) (see Turney & Pantel 2010 p. 153-158 and see again page 84). In a deep learning framework, in contrast, a researcher only has to define core features (Goldberg, 2016, p. 353). The core features' embedding representation vectors and parameters of functions, that define how feature representations are transformed into updated versions of these representations in deeper layers and define how core feature representations combine and interact to generate document representations, are learned during training. Thus, when applying deep neural networks, much less preprocessing is involved and representations are learned rather than being defined a priori. A researcher may, for example, merely tokenize each lowercased training text document into subwords and then feed each document (along with a corresponding output label) as an input to the neural network (see e.g. Devlin et al., 2019, p. 4174-4175).
- there are neural network architectures that can process texts as sequential data, that can account for dependencies between tokens, and that can construct contextualized representations of textual entities (see Section 1.2.3.6 on recurrent neural networks (RNNs), Section 1.2.3.7 on attention and the Transformer, and Section 1.2.3.9 on representations).
- representation learning models can be transferred across thematic domains, languages, or tasks, thereby leveraging potentially useful knowledge across learning processes (Ruder, 2019a) (see Section 1.2.3.8 on transfer learning).

1.2.3 Introduction to Deep Learning in the Context of Natural Language Processing

In the following, deep learning is introduced in an NLP context. In doing so, concepts and methods are explicated that are applied throughout the dissertation. Whilst subsections 1.2.3.1 to 1.2.3.8 introduce concepts and methods, subsection 1.2.3.9 assembles various of the presented aspects into an overall picture. Thereby, a mapping of the foundations of

Deep Learning. Deep learning is a subfield of machine learning in which deep neural networks, that learn layered representations of data, are applied (Goodfellow et al., 2016, p. 1-10; Chollet, 2021, ch. 1.1). Only if a neural network is deep (meaning that it comprises several hidden layers), it is considered a deep learning model (Kavlakoglu, 2020).

current methods in advanced NLP is provided. Against this background, Section 1.2.4 finally provides an overview of the supervised learning methods that are used in text-based political science research.

1.2.3.1 Deep Learning

The aim in supervised machine learning is to learn the true underlying function f that maps from data inputs (here: a corpus of documents $D = (d_1, \ldots, d_i, \ldots, d_N)$) to respective outputs $\boldsymbol{y} = [y_1, \ldots, y_i, \ldots, y_N]^\top$. In conventional machine learning approaches, the process of approximating f can be regarded as a two-step process (Goodfellow et al., 2016, p. 10): In the first step, representations $\boldsymbol{X}$ of raw textual inputs D are created, and then, in the second step, a supervised learning algorithm is applied to learn the mapping between representations $\boldsymbol{X}$ and outputs $\boldsymbol{y}$.

Because conventional supervised learning algorithms require prefabricated representations as an input that remain fixed during the training process, the mapping from the corpus of raw documents D to document representations $\boldsymbol{X}$ (which typically take the form of a document-feature matrix) is done manually by the researcher. When depicting this manual process in mathematical terms (though, of course, no parameters of a function are estimated in the usual sense), then for a single document d_i, this process can be represented as

$$\boldsymbol{x}_i = \mathsf{f}_l(d_i, \hat{\boldsymbol{\theta}}_l) \tag{1.29}$$

Subsequently, the mapping from input representations $\boldsymbol{X}$ to the respective outputs $\boldsymbol{y}$ is learned by a machine learning algorithm. After training the algorithm, the whole mapping for a single document can be described as

$$\hat{y}_i = \mathsf{f}(d_i, \hat{\boldsymbol{\theta}}) = \mathsf{f}_o(\mathsf{f}_l(d_i, \hat{\boldsymbol{\theta}}_l), \hat{\boldsymbol{\theta}}_o) \tag{1.30}$$

In contrast to conventional machine learning methods, deep neural networks not only learn a function that relates the data representations to the outputs but also learn representations of the textual inputs. They thereby alleviate the problems related to the manually guided process of representation creation.

On an abstract level, many deep learning structures can be described as a set of stacked (most often nonlinear) functions (Goodfellow et al., 2016, p. 164-165):

$$\mathsf{f}(d_i, \hat{\boldsymbol{\theta}}) = \mathsf{f}_o(\ldots \mathsf{f}_{l_3}(\mathsf{f}_{l_2}(\mathsf{f}_{l_1}(d_i, \hat{\boldsymbol{\theta}}_{l_1}), \hat{\boldsymbol{\theta}}_{l_2}), \hat{\boldsymbol{\theta}}_{l_3}) \ldots, \hat{\boldsymbol{\theta}}_o) \tag{1.31}$$

Deep neural networks require the inputs, that they take in and then process, to take the form of vectors (Goldberg, 2016, p. 360). Then, deep neural networks apply a chain of functions $(\mathsf{f}_{l_1}, \mathsf{f}_{l_2}, \mathsf{f}_{l_3}, \dots)$ to the input vectors and in that way produce layers of representations (in the form of sets of vectors). Each function maps from the representation in one layer to the representation in the next layer (Goodfellow et al., 2016, p. 164-165). The model as a whole thereby generates a complex, layered representation of the inputs (Goodfellow et al., 2016, p. 5). The elements of the representation vectors in the input layer are learned like any other parameter during the optimization process (Goldberg, 2016, p. 349). Moreover, the model parameters, whose values are learned in training, determine how representations from one layer are mapped to representations in the next layer. Hence, representations are learned rather than being preconstructed and fixed.

As they are composed of nested layers of functions, deep learning models are called *deep* (Goodfellow et al., 2016, p. 164-165). The representation layers are named hidden layers (Goodfellow et al., 2016, p. 165). The dimensionality of the hidden layer vectors is a model's width and the number of hidden layers is the depth of a model (Goodfellow et al., 2016, p. 8, 165). There, however, is no agreement on how many hidden layers there have to be for a model to be considered *deep* (Goodfellow et al., 2016, p. 8).

Note that it is not that deep neural networks directly would take a raw document d_i as an input. An initial transformation from d_i to a data format that the deep neural network can operate on is still required. In NLP, a document $d_i = (a_1, \dots, a_t, \dots, a_T)$ is commonly represented as a sequence of embedding vectors $(\boldsymbol{z}_{[a_1]}, \dots, \boldsymbol{z}_{[a_t]}, \dots, \boldsymbol{z}_{[a_T]})$. Yet, in contrast to the elaborate text preprocessing procedures that are usually conducted when applying conventional machine learning methods, here preprocessing is limited to the definition of core input features (Goldberg, 2016, p. 349-350). This usually involves a few simple steps: primarily tokenization, sometimes lowercasing, and then the initialization of each feature's embedding vector (Goldberg, 2016, p. 349). Moreover, the central difference is that whereas in conventional machine learning representations are typically not learned on the basis of data (though they sometimes may result from unsupervised dimension reduction techniques), in deep learning the model parameters that create representations are learned during the optimization process on the training data (Goodfellow et al., 2016, p. 10; Goldberg, 2016, p. 349).

1.2.3.2 Feedforward Neural Networks

A feedforward neural network (FNN)—also known as multilayer perceptron—is the most basic deep learning model (Goodfellow et al., 2016, p. 164). In an FNN, the input is processed forward through the functions of the network without loops (Goodfellow et al., 2016, p. 164). Figure 1.4 and Equations 1.32 to 1.35 describe an FNN with L hidden layers,

vector input $\boldsymbol{x}$ and vector output $\boldsymbol{y}$:[38]

$$\boldsymbol{h}_1 = \sigma_l(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \tag{1.32}$$

$$\boldsymbol{h}_2 = \sigma_l(\boldsymbol{W}_2\boldsymbol{h}_1 + \boldsymbol{b}_2) \tag{1.33}$$

$$\dots$$

$$\boldsymbol{h}_l = \sigma_l(\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{b}_l) \tag{1.34}$$

$$\dots$$

$$\boldsymbol{y} = \sigma_o(\boldsymbol{W}_o\boldsymbol{h}_L + \boldsymbol{b}_o) \tag{1.35}$$

$\boldsymbol{h}_1 \in \mathbb{R}^{K_1}, \boldsymbol{h}_2 \in \mathbb{R}^{K_2}, \dots, \boldsymbol{h}_l \in \mathbb{R}^{K_l}, \dots, \boldsymbol{h}_L \in \mathbb{R}^{K_L}$ are the data representations in the hidden layers. Equation 1.35 describes the operations in the output layer. $\boldsymbol{W}_1$ to $\boldsymbol{W}_o$ are weight matrices, $\boldsymbol{b}_1$ to $\boldsymbol{b}_o$ are bias vectors, σ_l is the nonlinear activation function in the hidden layers and σ_o is the activation function in the output layer.

FNNs can be understood as a stack of nonlinear, vector-valued functions that are combined with linear transformations (Goodfellow et al., 2016, p. 164-165; Ruder, 2019a, p. 31): Each layer takes as an input the representation from the previous hidden layer, $\boldsymbol{h}_{l-1}$, then an affine function followed by a nonlinear activation function is applied to produce a new representation $\boldsymbol{h}_l$. The network as a whole thereby generates a layered representation of the data, $(\boldsymbol{h}_1, \dots, \boldsymbol{h}_l, \dots, \boldsymbol{h}_L)$ (Goodfellow et al., 2016, p. 164-165).

As a stack of linear functions can be represented by another linear function, the fact that FNNs (and neural networks in general) are a stack of nonlinear activation functions (alternated with linear transformations) is the central building block that gives neural networks their high capacity to approximate a large spectrum of complex relationships between inputs and outputs (Goodfellow et al., 2016, p. 168; Ruder, 2019a, p. 31).

Hidden layer activation functions are usually applied per element (Goodfellow et al., 2016, p. 171). Accordingly, if the affine function in the lth hidden layer is $\boldsymbol{q} = \boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{b}_l$ and if $\boldsymbol{q}$ is a K-dimensional vector, $\boldsymbol{q} = [q_1, \dots, q_k, \dots, q_K]^\top$, then an activation function is applied on each element q_k. A widely known activation function is the Rectified Linear Unit (ReLU) (Nair & Hinton, 2010). ReLU is defined as (Goodfellow et al., 2016, p. 171)

$$\sigma_l(\boldsymbol{q})_k = max\{0, q_k\} \tag{1.36}$$

$\sigma_l(\boldsymbol{q})_k$ then is the kth element of hidden representation vector $\boldsymbol{h}_l$. (For a visualization see Figure 1.5a.) Another activation function that, for example, is used in Transformer-based language representation models that are applied in this dissertation, is the Gaussian Error Linear Unit (GELU) (Hendrycks & Gimpel, 2016). In GELU, each input q_k is weighted

[38] In correspondence with the notation common in the deep learning literature (Goldberg, 2016, p. 346), vectors in the following are treated as column vectors.

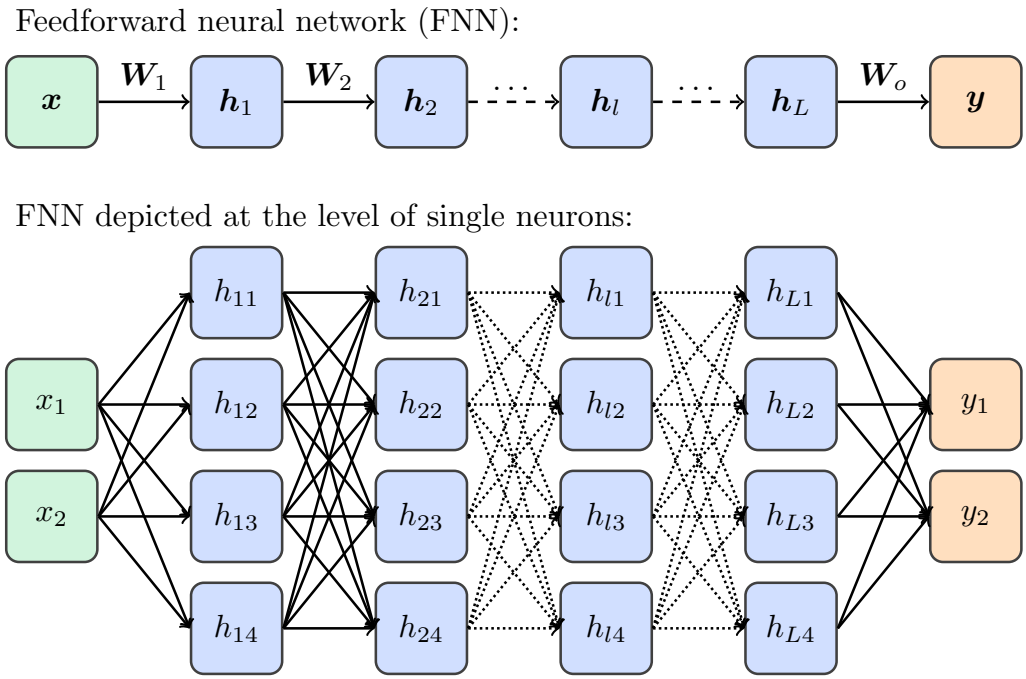
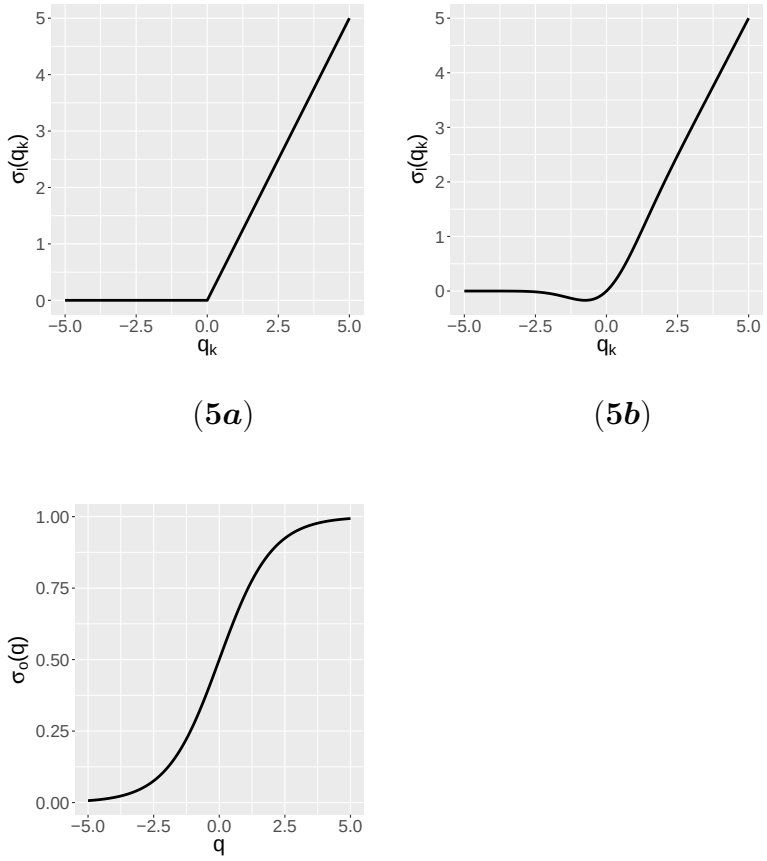


Figure 1.4: **Feedforward Neural Network.** This figure depicts an FNN with L hidden layers and output vector $\boldsymbol{y}$. Inputs are given in green, hidden representations in blue, and outputs in orange. The connecting lines indicate the transformations from one representation to the next. The illustration at the top depicts each vector as a single box. The illustration below depicts each element of each vector as a single box. A single vector element, e.g. h_{21}, is the output from a single operational unit in the network, that is called neuron (or simply: unit) (Goodfellow et al., 2016, p. 170; Goldberg, 2016, p. 354-355). h_{21} is the output from the neuron that conducts the following operation: $h_{21} = \sigma_l(\boldsymbol{w}^\top \boldsymbol{h}_1 + b_1)$ (Goldberg, 2016, p. 354-355). The lines connecting the representation vector elements indicate the transformations that are encoded in each weight matrix. A weight matrix linearly maps the representation from the previous layer into a new representation (Goodfellow et al., 2016, p. 168). (Not explicitly shown here are the bias terms and the nonlinear activation functions.)



(**5*a***) (**5*b***)

(**5*c***)

Figure 1.5: **Activation Functions.** (**5*a***) ReLU activation function. (**5*b***) GELU activation function. (**5*c***) Logistic sigmoid function.

by its value on the cumulative distribution function of the standard normal distribution (Hendrycks & Gimpel, 2016, p. 2) (see Figure 1.5b):

$$\sigma_l(\boldsymbol{q})_k = q_k \, \Phi(q_k) \tag{1.37}$$

In general, the functional form of an activation function affects the gradient-based learning process. For example, if an activation function saturates, gradients may vanish (Li et al., 2020a). A vanishing gradient impedes the flow of signaling gradients through the network and implies that hardly any training takes place on affected units (Hansen, 2019). (For detailed information on how activation functions affect optimization via gradient descent with backpropagation see Section 1.2.3.3 below)

The output layer activation function σ_o has to map from the last hidden layer representation $\boldsymbol{h}_L$ to the output $\boldsymbol{y}$ whose dimensionality is task-specific. In binary classification tasks with $y \in \{0, 1\}$, the logistic sigmoid function is typically employed (Goodfellow et al., 2016, p. 179-180). In this case, the affine function in the output layer in Equation 1.35 would be made to produce a scalar output: $q = \boldsymbol{w}_o \boldsymbol{h}_L + b_o$. Then, the logistic sigmoid function

$$\sigma_o(q) = \frac{exp(q)}{1 + exp(q)} \tag{1.38}$$

maps q to a value in the range $(0, 1)$. This value is interpreted as the probability that $y = 1$. (The logistic sigmoid function is depicted in Figure 1.5c.)[39]

In multi-cass classification tasks with $y \in \{\mathcal{G}_1, \ldots, \mathcal{G}_c, \ldots, \mathcal{G}_C\}$, a common choice for σ_o is the softmax function (Goodfellow et al., 2016, p. 180-181). In this case, the affine transformation in the output layer in Equation 1.35 would be designed to produce a C-dimensional vector: $\boldsymbol{q} = \boldsymbol{W}_o \boldsymbol{h}_L + \boldsymbol{b}_o$, where $\boldsymbol{q} = (q_1, \ldots, q_c, \ldots, q_C)$. Then, the softmax function is

$$\sigma_o(\boldsymbol{q})_c = \frac{exp(q_c)}{\sum_{j=1}^{C} exp(q_j)} \tag{1.39}$$

$\sigma_o(\boldsymbol{q})_c$ is the cth element of the C-dimensional softmax function output and gives the probability that $y = \mathcal{G}_c$.[40]

[39] Note here the connection to the logistic loss outlined in Equation 1.11. In the more general formulation in Equation 1.11, the input to the logistic sigmoid function is the output from a supervised learning algorithm f that is parameterized by parameter set $\tilde{\boldsymbol{\theta}}$. In the more specific case here, q is the output produced by a neural network which is parameterized by parameter set $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_l, \ldots, \boldsymbol{W}_L, \boldsymbol{w}_o, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_l, \ldots, \boldsymbol{b}_L, b_o\}$.

[40] Note here the connection to Equation 1.14. In the more general formulation in Equation 1.14, the input to the softmax function is the output from a supervised learning algorithm f that is parameterized by parameter set $\tilde{\boldsymbol{\theta}}$. In the more specific case here, q_c is the output produced by a neural network which is parameterized by parameter set $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_l, \ldots, \boldsymbol{W}_L, \boldsymbol{W}_o, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_l, \ldots, \boldsymbol{b}_L, \boldsymbol{b}_o\}$.

1.2.3.3 Gradient Descent and Backpropagation

In supervised learning, a researcher passes the training data tuples $(\boldsymbol{x}_i, y_i)_{i=1}^N$ to the neural network. Hence, the input $\boldsymbol{x}_i$ and the corresponding output label y_i are being given and fixed for each training instance. All other elements, the weights and biases, are parameters whose values are learned in the optimization process (Goodfellow et al., 2016, p. 165). For the FNN described in Equations 1.32 to 1.35 the parameter set hence is: $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_l, \ldots, \boldsymbol{W}_L, \boldsymbol{W}_o, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_l, \ldots, \boldsymbol{b}_L, \boldsymbol{b}_o\}$. As each $\boldsymbol{W}_l$ is a $K_l \times K_{l-1}$ matrix of to be estimated parameters and each $\boldsymbol{b}_l$ is a K_l-dimensional vector of parameters, deep neural networks usually have a very large number of parameters (typically significantly more parameters than conventional machine learning methods). In order to effectively train deep neural networks one therefore needs substantive computational resources and a large number of training data.

Gradient Descent. When training neural networks, variants of the gradient descent algorithm are typically implemented (Goodfellow et al., 2016, p. 173). Gradient descent is an optimization algorithm designed to find the local minimum of a function g (Goodfellow et al., 2016, p. 80-81). The algorithm makes use of the gradient, which is a vector of partial derivatives: If a function g maps the values of H variables into a single value, e.g. $g : \mathbb{R}^H \to \mathbb{R}$, then the derivative of g at point $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_h, \ldots, \theta_H]^\top$ is an H-dimensional vector, called the gradient (Johnson, 2017, p. 2). The gradient is commonly denoted as $\nabla_\theta g(\boldsymbol{\theta})$. If $y = g(\boldsymbol{\theta})$, the gradient is also denoted as $\frac{\partial y}{\partial \boldsymbol{\theta}}$ (Johnson, 2017, p. 2).

$$\frac{\partial y}{\partial \boldsymbol{\theta}} = \left[\frac{\partial y}{\partial \theta_1}, \ldots, \frac{\partial y}{\partial \theta_h}, \ldots, \frac{\partial y}{\partial \theta_H}\right]^\top \tag{1.40}$$

The hth component of the gradient, $\frac{\partial y}{\partial \theta_h}$, is the partial derivative of g with respect to θ_h (Johnson, 2017, p. 2). It tells how y changes given an infinitesimally small change in θ_h (Moore & Siegel, 2013, p. 359).[41] The gradient of function g at point $\boldsymbol{\theta}$ indicates the direction in which function g is increasing fastest (Moore & Siegel, 2013, p. 362) and the direction of the negative gradient gives the direction of the steepest descent (Goodfellow et al., 2016, p. 83).

Gradient descent is an iterative algorithm that makes use of this directional information contained in the gradient. At each iteration, the gradient descent algorithm calculates the gradient of g at current point $\boldsymbol{\theta}_j$ and then updates $\boldsymbol{\theta}_j$ by moving into the direction given

[41] Just as the gradient is the generalization of a partial derivative in the sense that it is a vector of partial derivatives, there is a generalization of the gradient to matrix form: If function g maps the values taken by H variables into a vector of size K, $g : \mathbb{R}^H \to \mathbb{R}^K$, then the derivative of g at $\boldsymbol{\theta}$ is a $K \times H$ matrix of partial derivatives known as the Jacobian (Johnson, 2017, p. 3). The Jacobian gives the partial derivative of each of the K outputs with respect to each of the H inputs. The Jacobian can be generalized to a tensor (Johnson, 2017, p. 4). In the following, as is often done in the literature, the term *gradient* will be used irrespective of the dimensionality of the partial derivatives in question and thus may denote a scalar, vector, matrix, or tensor of partial derivatives.

by the negative gradient (Goodfellow et al., 2016, p. 83-84):

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \eta \nabla_{\boldsymbol{\theta}_j} g(\boldsymbol{\theta}_j) \tag{1.41}$$

$\eta \in \mathbb{R}_+$ is the learning rate. The hope is that by iteratively computing the gradient and then moving into the direction of the negative gradient with an adequately small learning rate η, the algorithm will approach a local minimum (Goldberg, 2016, p. 369-371; Goodfellow et al., 2016, p. 173).[42]

In the context of training neural networks, function g is the empirical risk $\mathcal{R}_{emp}$. The empirical risk is defined as

$$\mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})) \tag{1.42}$$

(see again Section 1.1.1 as well as Equations 1.2 and 1.6) (Vapnik, 1991, p. 832-833; Goodfellow et al., 2016, p. 272-273). This is, when training neural networks the gradient descent algorithm is employed to find the set of parameter values that minimizes the mean of the training set instances' losses (Goldberg, 2016, p. 369). To compute the gradients, the backpropagation algorithm (Rumelhart et al., 1986) is used (Goldberg, 2016, p. 371).

Gradient descent with backpropagtion starts with initializing the set of parameters $\tilde{\boldsymbol{\theta}}$ of a neural network f (Goodfellow et al., 2016, p. 291). Then, the following steps are repeated as long as the stopping criteria are not met (Goldberg, 2016, p. 370):

1. For each of N training set instances $(\boldsymbol{x}_i, y_i)$,
   (a) the input $\boldsymbol{x}_i$ is propagated forward through the layers of the network to obtain the network's prediction $\hat{y}_i = \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}_j)$ (Han et al., 2012, p. 401)
   (b) the loss $L = \mathcal{L}(y_i, \hat{y}_i)$ is determined and the chain rule is applied in a backward manner to compute the gradient on all parameters (Li et al., 2020c). (Details on this backward propagation step are given below.)
2. The parameters are updated via gradient descent

$$\tilde{\boldsymbol{\theta}}_{j+1} = \tilde{\boldsymbol{\theta}}_j - \eta \nabla_{\tilde{\boldsymbol{\theta}}_j} \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_j) \tag{1.43}$$

   where the gradient is the mean of the gradients over all N training instances (Goodfellow et al., 2016, p. 275, 291):

$$\nabla_{\tilde{\boldsymbol{\theta}}_j} \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_j) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\tilde{\boldsymbol{\theta}}_j} \mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}_j)) \tag{1.44}$$

[42] If applied on a convex function, all local minima are global minima and thus gradient descent with an adequately small learning rate η will converge to a global minimum (Goldberg, 2016, p. 371; Thomas, 2018, p. 32). If the function is nonconvex, there is no guarantee that gradient descent will converge to the global minimum (Goldberg, 2016, p. 371). The non-linearity of neural networks implies that most loss functions are nonconvex (Goodfellow et al., 2016, p. 173). Hence, there is no global convergence guarantee and gradient-based learning will move toward a low value (Goldberg, 2016, p. 371; Goodfellow et al., 2016, p. 173).

This outlined procedure, known as batch gradient descent, in which for every single update of parameter values the gradient is computed based on *all* N training instances, is highly expensive (Goodfellow et al., 2016, p. 275). The usually applied approach thus is to compute the gradient on the basis of a small random sample of S training set instances (Goodfellow et al., 2016, p. 275-276, 291):

$$\nabla_{\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}} \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}) = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}} \mathcal{L}(y_s, \mathsf{f}(\boldsymbol{x}_s, \tilde{\boldsymbol{\theta}}_{\boldsymbol{j}})) \tag{1.45}$$

This procedure is called mini-batch stochastic gradient descent (Goodfellow et al., 2016, p. 275-276, 291). The gradient computed on a mini-batch of training samples typically provides a good approximation of the gradient computed on the entire training set (Goodfellow et al., 2016, p. 275). S usually is set to a value in the range from two to a few hundred (Ruder, 2019a, p. 25).

The mini-batch size S and the learning rate η are hyperparameters that are watchfully tuned when training neural networks (Li et al., 2020b). If η is too high, the loss values may oscillate considerably (Goodfellow et al., 2016, p. 291). A too small η, in contrast, causes the learning process to be slow (Goodfellow et al., 2016, p. 291). Usually, the learning rate is not kept constant but altered throughout the optimization process according to a learning rate schedule (Goodfellow et al., 2016, p. 290-291). Moreover, there are modifications of gradient descent, such as AdaGrad (Duchi et al., 2011), RMSProp (Hinton et al., 2012), and Adam (Kingma & Ba, 2015), that adapt the learning rate for each individual parameter (Goodfellow et al., 2016, p. 303-305).

Backpropagation. This section explains in more detail the computation of the gradients. The following small neural network with a single hidden layer that contains a single neuron serves as an illustrative example:

$$q_1 = w_1 x + b_1 \tag{1.46}$$

$$h_1 = \sigma_1(q_1) \tag{1.47}$$

$$\hat{y} = w_2 h_1 \tag{1.48}$$

Here x serves as the input to an affine function, that in turn is fed to a nonlinear activation function, σ_1, computing hidden state h_1. The hidden state then is the input for the next layer which here is the output layer that computes prediction $\hat{y}$. Given the predicted $\hat{y}$ and the true y, the loss $L = \mathcal{L}(y, \hat{y})$ as well as the gradient of the loss function with respect to output $\hat{y}$, $\frac{\partial L}{\partial \hat{y}}$, can be computed. The gradient $\frac{\partial L}{\partial \hat{y}}$ tells how the loss L changes given an infinitesimally small change in the predicted output $\hat{y}$.

The aim in optimization is to find the set of parameter values that minimize the mean across the losses of individual training instances (Vapnik, 1991, p. 832-833). To do so, for each sampled training instance, gradient descent computes the negative gradient, that tells in which direction the loss function decreases fastest (Goldberg, 2016, p. 371). Then the mean across the sampled training instances' negative gradients is computed and the

Chain Rule. The chain rule is a computational rule that allows one to compute the derivatives in composite functions (Moore & Siegel, 2013, p. 119). For example, if $y = f(x)$ and $z = g(y)$ and thus the corresponding computation graph is (Johnson, 2017, p. 1-2):

$$x \xrightarrow{f} y \xrightarrow{g} z$$

then to know how z changes given a small change in x, one would have to compute the derivative of z with respect to x, denoted as $\frac{\partial z}{\partial x}$. The chain rule states that

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

This is, in order to know how an infinitesimal change in x affects z, one computes the derivative of y with respect to x (that gives the change in y due to a change in x) and multiplies this expression with the derivative of z with respect to y (that informs about the change in z due to a change in y). Since neural networks can be described as a set of composite functions (see Equation 1.31), the chain rule is required to compute the gradients.

parameter values are moved into the direction of this negative gradient (Goldberg, 2016, p. 371). To compute the gradients, the chain rule is applied (Goldberg, 2016, p. 371).

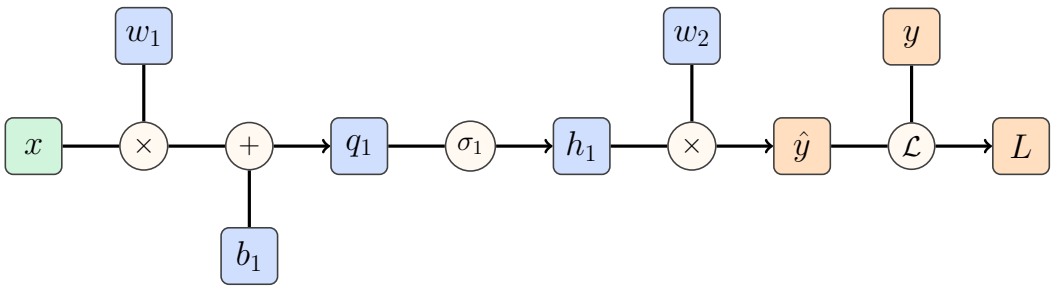


Figure 1.6: **Computation Graph.** Computation graph describing the neural network laid out in Equations 1.46 to 1.48.

With the help of Figure 1.6 that visualizes the computation graph of the neural network described in Equations 1.46 to 1.48, it can be established that in the jth iteration of the algorithm the gradients for the weights and biases in the given example are:

$$\frac{\partial L_j}{\partial w_{2j}} = \frac{\partial L_j}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial w_{2j}} \tag{1.49}$$

$$\frac{\partial L_j}{\partial w_{1j}} = \frac{\partial L_j}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial h_{1j}}\frac{\partial h_{1j}}{\partial q_{1j}}\frac{\partial q_{1i}}{\partial w_{1i}} \tag{1.50}$$

$$\frac{\partial L_j}{\partial b_{1j}} = \frac{\partial L_j}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial h_{1j}} \frac{\partial h_{1j}}{\partial q_{1j}} \frac{\partial q_{1j}}{\partial b_{1j}} \tag{1.51}$$

To finally update the parameters from iteration j to $j+1$, the value of each parameter is made to move into the direction of its negative gradient at j with step size η:

$$w_{2,j+1} = w_{2j} - \eta \frac{\partial L_j}{\partial w_{2j}} \tag{1.52}$$

$$w_{1,j+1} = w_{1j} - \eta \frac{\partial L_j}{\partial w_{1j}} \tag{1.53}$$

$$b_{1,j+1} = b_{1j} - \eta \frac{\partial L_j}{\partial b_{1j}} \tag{1.54}$$

This illustration emphasizes that in order to compute the gradient for a particular parameter, the gradients of all elements that depend on operations resulting from this parameter have to be known (Ruder, 2019a, p. 36). This illustration also makes clear that backpropagation of gradients can be viewed as a way via which for each parameter it is computed how a change in the parameter affects the loss function—and thus in which direction to change the parameter value to decrease the loss (Li et al., 2020c). The gradients hereby can be seen as a signal that runs backwards through the network telling each parameter to increase or decrease its value for the overall loss to decrease (Li et al., 2020c).

Moreover, the backpropagation illustration also helps to outline the effects of specific activation functions: Historically, the sigmoid function (see Equation 1.38 and Figure 1.5c) frequently has been used as an activation function in neural networks (Li et al., 2020a). Due to its property to saturate and its first derivative to approach a value of 0 for large positive and large negative input values, the sigmoid activation function may cause the gradients to vanish (i.e. to assume a value close to 0) if the input provided by the affine function assumes a large positive or negative value (Hansen, 2019; Li et al., 2020a). Because the gradients are multiplied with each other in the backpropagation process (see Equations 1.49 to 1.51), vanishingly small gradients can mute the traveling of signaling gradients through the network (Li et al., 2020a). If this vanishing gradient behavior occurs across many units in a large network, there will be hardly any training (Hansen, 2019). To illustrate, if σ_1 in Equation 1.47 were the sigmoid activation function and if in iteration j the output from the affine function q_{1j} would assume a large positive value, then $\frac{\partial h_{1j}}{\partial q_{1j}}$ in Equation 1.50 would become close to zero and thereby make the gradient $\frac{\partial L_j}{\partial w_{1j}}$ also assume a very small value (Hansen, 2019). The parameter update for weight w_{1j} (see Equation 1.53) consequently also would be extremely small (Hansen, 2019).

The ReLU activation function (see Equation 1.36 and Figure 1.5a) also is not free of shortcomings: As the first derivative of the ReLU function is 1 for $q_k > 0$, the gradients for $q_k > 0$ will not vanish but stay distinguishable from 0 (Hansen, 2019). For $q_k < 0$, however, the gradient becomes 0, thereby causing all parameters that depend on a unit

with a gradient of 0 to not update at all (Hansen, 2019). If σ_1 in Equation 1.47 would be the ReLU function, and if in the jth iteration $q_{1j} < 0$, then $\frac{\partial h_{1j}}{\partial q_{1j}}$ would be 0 and therefore also $\frac{\partial L_j}{\partial w_{1j}} = 0$. Consequently there would be no update for the weight w_1 as Equation 1.53 would reduce to $w_{1,j+1} = w_{1j}$. This is known as the dying ReLU problem (Li et al., 2020a).

1.2.3.4 Regularization

Due to their layered architecture and their use of nonlinear activation functions, neural networks have a high capacity, meaning that they can approximate a large range of complex functions (Goodfellow et al., 2016, p. 5, 110, 168).[43] The wider and deeper a neural network, the higher its capacity (Li et al., 2020a). The capacity of neural networks tends to be higher than that of conventional machine learning algorithms that operate without hidden layers and are often linear learners (Goodfellow et al., 2016, p. 194; Goldberg, 2016, p. 345, 355).

In general, depending on the complexity of the data structure and a machine learning algorithm's capacity, underfitting or overfitting can occur (Goodfellow et al., 2016, p. 109-110). An algorithm with reduced capacity may underfit the training data in the sense that it is not able to reach an acceptably low error rate on the training data set (Goodfellow et al., 2016, p. 109-110). A learning algorithm with a too high capacity, in contrast, may overfit on the training data and therefore have a reduced generalization performance on yet unused test data (James et al., 2013, p. 32; Goodfellow et al., 2016, p. 110).

Text data emerge from very complex data generating processes (Goodfellow et al., 2016, p. 225). And the functions to be approximated in NLP tasks (e.g. the mapping from a sequence of text to discrete class labels) are also highly complex. The empirically superior prediction accuracy of deep learning models over conventional models (Goldberg, 2016, p. 347-348) indicates that the high expressivity of deep learning models is central to approximating the complex functions involved in NLP tasks.

When applying neural networks in practice, it has been found that the highest generalization performances tend to be achieved by larger and deeper neural networks in which overfitting is controlled via regularization strategies (Goodfellow et al., 2016, p. 225; Li

[43] According to the universal approximation theorem, an FNN with a single hidden layer of high enough dimensionality can represent any continuous function defined on a compact subset of $\mathbb{R}^N$ (Goodfellow et al., 2016, p. 194). Hence, a large enough neural network can in theory approximate any continuous function f with desired accuracy (Goodfellow et al., 2016, p. 194-195). In practice, however, it is unclear how large a network has to be to achieve a satisfactory approximation to a given function f (Goodfellow et al., 2016, p. 195). And it may very well be the case that the unknown required size is infeasibly large (Goodfellow et al., 2016, p. 195). In general, deeper networks with more hidden layers but fewer units per layer provide more efficient approximations and thus seem preferable over shallow networks with many units per layer (Goodfellow et al., 2016, p. 195-197).

et al., 2020a). Regularization strategies are procedures that aim at reducing an algorithm's generalization error as estimated on the test set (Goodfellow et al., 2016, p. 224). In doing so, regularization strategies potentially cause an increase of the training error (Goodfellow et al., 2016, p. 224).

A very common regularization strategy, that is also used for neural networks, is weight decay. Here, a regularization term is added to the empirical risk in order to restrain the capacity of a learning algorithm (Goodfellow et al., 2016, p. 226). The regularized empirical risk, $\mathcal{R}_{emp,reg}(\tilde{\boldsymbol{\theta}})$, then is

$$\mathcal{R}_{emp,reg}(\tilde{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})) + \lambda\,\Omega(\tilde{\boldsymbol{\theta}}) \tag{1.55}$$

where $\Omega(\tilde{\boldsymbol{\theta}})$ is a parameter norm penalty serving as the regularization term and $\lambda \in [0, \infty)$ is a hyperparameter that balances the relative weight given to the penalty vs. the mean training loss (Goodfellow et al., 2016, p. 226). In the case of L^2 weight decay (also known as L^2 regularization or ridge regression) $\Omega(\tilde{\boldsymbol{\theta}}) = \frac{1}{2}\tilde{\boldsymbol{\theta}}^\top\tilde{\boldsymbol{\theta}}$ (Goodfellow et al., 2016, p. 227). As $\frac{1}{2}\tilde{\boldsymbol{\theta}}^\top\tilde{\boldsymbol{\theta}}$ is smaller for smaller values of the parameters, the penalty moves the parameter values closer to 0 (James et al., 2013, p. 215; Goodfellow et al., 2016, p. 227).[44]

Another regularization strategy frequently used in deep learning is dropout (Srivastava et al., 2014). In dropout, a random sample of neural units and their connections are dropped during the training process (Srivastava et al., 2014, p. 1929). More specifically, each time a training example is presented, each unit and its corresponding connections in the neural network are dropped from the network with probability $1 - p$ (Srivastava et al., 2014, p. 1930).[45] Thereby, a thinned network is sampled from the original network (Srivastava et al., 2014, p. 1931). Stochastic gradient descent then is applied on the thinned network (Srivastava et al., 2014, p. 1934). As for each training instance a new thinned network is sampled by randomly dropping units, dropout trains a collection of thinned networks (Srivastava et al., 2014, p. 1931). When making a prediction for a test instance after training, the original network without dropout is used and the weights for each unit are multiplied with the probability p of not being dropped (Srivastava et al., 2014, p. 1931).

[44] As λ increases, a smaller value of $\sqrt{\tilde{\boldsymbol{\theta}}^\top\tilde{\boldsymbol{\theta}}}$ (which is the L^2 norm of the parameter vector) is preferred (James et al., 2013, p. 216). Increasing λ in L^2 regularization thus has the effect of decreasing the combined parameter values (their L^2 norm) (James et al., 2013, p. 216). Single parameter values, however, may increase while λ is set to a higher value (James et al., 2013, p. 216). Note furthermore that typically only the weights, $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \dots\}$, of neural networks but not the bias terms, $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots\}$, are regularized (Goodfellow et al., 2016, p. 226).

[45] If $\boldsymbol{h}_{l-1} = [h_{l-1,1}, \dots, h_{l-1,k}, \dots, h_{l-1,K}]^\top$ is the K-dimensional hidden state vector that has been produced as an output by the $(l-1)$th hidden layer (see Equation 1.34), then each element $h_{l-1,k}$ of this vector is independently multiplied with a Bernoulli random variable that assumes a value of 1 with probability p and a value of 0 with probability $1 - p$ (Srivastava et al., 2014, p. 1933-1934). Hence, only units that survive dropout (as they have been multiplied with 1 rather than 0), pass on information to the next layer (Srivastava et al., 2014, p. 1933-1934).

Dropout thus can be viewed as a model averaging procedure in which regularization is achieved via averaging over the predictions of several models (here: many sampled thinned networks) that make different errors on the test set (Srivastava et al., 2014, p. 1941; Goodfellow et al., 2016, p. 253, 255). A second reason why dropout works as a regularization strategy is that by randomly dropping units, the procedure hinders the individual units from adapting too much to each other (Srivastava et al., 2014, p. 1932, 1943). This means that each unit cannot rely on the presence of other units to counteract its errors but has to generate useful representations across various contexts by itself (Srivastava et al., 2014, p. 1932, 1943).

1.2.3.5 Convolutional Neural Networks

FNNs that only contain fully connected layers (as does the network described in Equations 1.32 to 1.35) are rarely used as a standalone architecture in NLP. Yet they are frequently incorporated as elements within other architectures (see e.g. Vaswani et al., 2017). Furthermore, they form the basis of more complex architectures: Convolutional neural networks (CNNs) (LeCun & Bengio, 1995) are FNNs that in at least one layer—rather than having a fully connected layer with a matrix-vector multiplication as in Equations 1.32 to 1.35—conduct a convolution operation (Goodfellow et al., 2016, p. 326). Convolution layers are usually followed by pooling layers (Goodfellow et al., 2016, p. 326). A visualization of a convolution and pooling operation is given in Figure 1.7.

In a convolution layer, a filter of a given window size V slides over a document's sequence of tokens (Goldberg, 2016, p. 386). Step by step, the filter (which typically is a function that applies first a linear and then a nonlinear transformation and is characterized by trainable weights $\boldsymbol{W}$, bias terms $\boldsymbol{b}$, and nonlinear activation function σ) is applied to the concatenated embeddings of those V tokens covered by the filter window (Goldberg, 2016, p. 386). By doing so, for each of the S windows in a document that comprise V tokens, a single representation vector $\boldsymbol{h}_s$ is produced (Goldberg, 2016, p. 386). If $\boldsymbol{z}_s = [\boldsymbol{z}_{[a_s]}; \boldsymbol{z}_{[a_{s+1}]}; \ldots; \boldsymbol{z}_{[a_{s+V-1}]}]^\top$ is the column vector of concatenated representations of tokens in the sth window, then the operation is (Goldberg, 2016, p. 386):

$$\boldsymbol{h}_s = \sigma(\boldsymbol{W}\boldsymbol{z}_s + \boldsymbol{b}) \tag{1.56}$$

The filter's weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$ are parameters to be learned (Collobert et al., 2011, p. 2502). Each window representation is of dimension K: $\boldsymbol{h}_s = [h_{s1}, \ldots, h_{sk}, \ldots, h_{sK}]^\top$.

In the pooling layer, the information encoded in the S window representations $(\boldsymbol{h}_s)_{s=1}^S$ is aggregated. A frequently used pooling operation is max pooling (Goldberg, 2016, p. 389). Here, at each of the K vector elements, only the maximum value across the S window representations is retained such that a single vector $\boldsymbol{h}_{pool}$ of dimensionality K is obtained (Collobert et al., 2011, p. 2504; Goldberg, 2016, p. 387):

$$h_{pool,k} = \max_{1<s\leq S} h_{s,k} \tag{1.57}$$

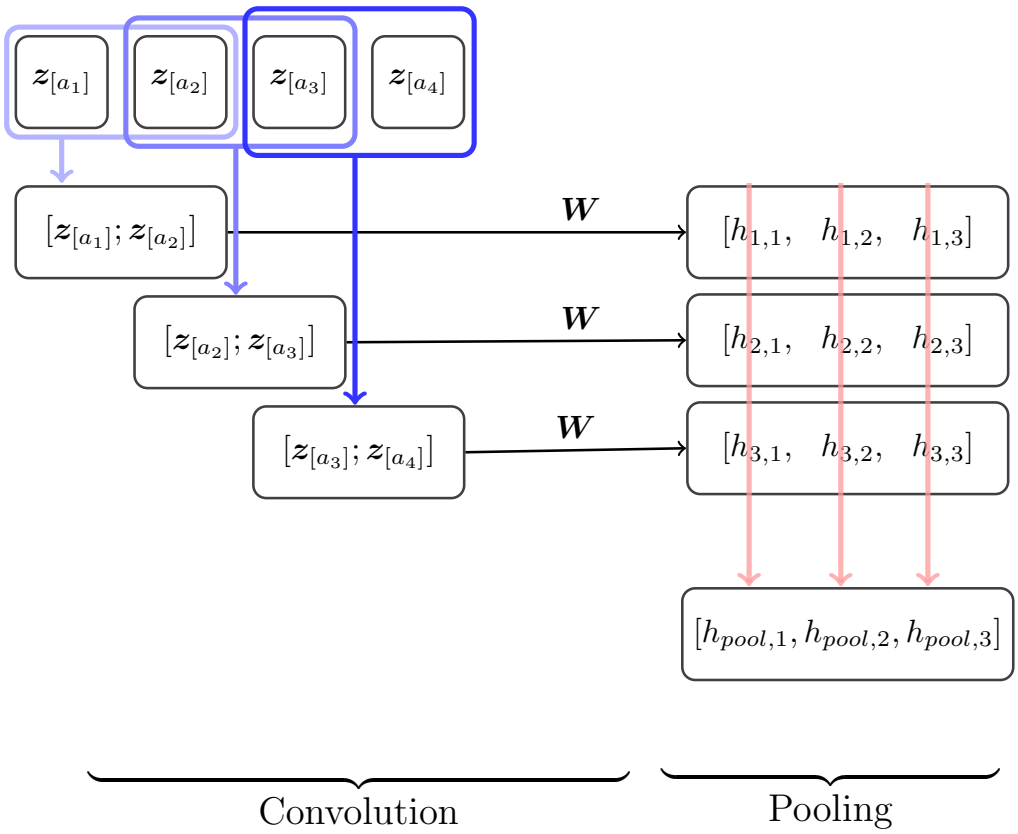


Figure 1.7: **Convolution and Pooling.** This figure illustrates the convolution operation followed by max pooling. Here, a convolution filter of size $V = 2$ slides over a sequence of four tokens, that are given by four representation vectors $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_4]})$. At each step, $V = 2$ vectors are concatenated and the filter is applied, thereby generating at each step a K-dimensional window representation vector $\boldsymbol{h}_s$. (Here $K = 3$ and thus $\boldsymbol{h}_1 = [h_{1,1}, h_{1,2}, h_{1,3}]$.) The filter of size $V = 2$ applied across the sequence of four tokens yields $S = 3$ window representation vectors, $(\boldsymbol{h}_s)_{s=1}^{3}$. Then in max pooling, for each of the $K = 3$ elements of the window representation vectors, the maximum value across the $S = 3$ window representation vectors is extracted to form the $K = 3$-dimensional pooled vector $\boldsymbol{h}_{pool} = [h_{pool,1}, h_{pool,2}, h_{pool,3}]$. This figure is adapted from "A primer on neural network models for natural language processing," by Y. Goldberg, 2016, *Journal of Artificial Intelligence Research, 57*(1), p. 387. Copyright 2016 by the AI Access Foundation.

Note that it is common for a convolution layer to have several filters with different window sizes (Kim, 2014, p. 1747). The obtained representations then typically serve as inputs to further layers (e.g. a fully connected FNN) to at some point enter an output layer that generates a prediction (Kim, 2014, p. 1747). Given the CNN's prediction and the true value, the loss is computed and the gradients are obtained via backpropagation through all layers (including the pooling and the convolution layers) (Goldberg, 2016, p. 387).

CNNs capture the local information that is most predictive for a task (e.g. token sequences that are most important for a sentiment classification task) (Goldberg, 2016, p. 385-386). In their seminal works, Collobert & Weston (2008) and Collobert et al. (2011) were the first to use CNNs for the processing of text data. Other important implementations of CNNs in the field of NLP are Kalchbrenner et al. (2014), Kim (2014), and Zhang et al. (2015b). CNNs, however, play a more central role in the field of computer vision where they are applied, for example, for the recognition of objects in images (Krizhevsky et al., 2012).[46]

1.2.3.6 Recurrent and Recursive Neural Networks

CNNs identify local predictive elements in a sequence, but in doing so information on the structure and order of textual tokens is largely only retained for local sequences (Goldberg, 2016, p. 348, 389). Recurrent neural networks (RNNs) (Elman, 1990), in contrast, can capture information across a document's entire token sequence. In an RNN, each token in a sequence $(a_1, \ldots, a_t, \ldots, a_T)$ is processed one step at a time, and at each step the hidden state $\boldsymbol{h}_t$ is updated based on the previous hidden state $\boldsymbol{h}_{t-1}$ and the embedding vector $\boldsymbol{z}_{[a_t]}$ of new input token a_t (Goldberg, 2016, p. 389-390). Moreover, at each time step, an output vector $\boldsymbol{y}_t$ is produced from the current hidden state $\boldsymbol{h}_t$ (Goldberg, 2016, p. 389-390). A basic formulation of an RNN is (Amidi & Amidi, 2019):

$$\boldsymbol{h}_t = \sigma_h(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ha}\boldsymbol{z}_{[a_t]} + \boldsymbol{b}_h) \tag{1.58}$$

$$\boldsymbol{y}_t = \sigma_y(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{1.59}$$

where $\boldsymbol{W}_{hh}$ is the "recurrent connection"(Graves, 2014, p. 4) that connects previous to current hidden states. $\boldsymbol{W}_{ha}$ links the input to the hidden layer, and $\boldsymbol{W}_{yh}$ relates the hidden state to the output (Graves, 2014, p. 4).[47] (For a visualization of an RNN that is depicted across time steps see Figure 1.8.) An RNN can be conceived as a network in which the hidden state is updated sequentially such that at a given time step t the hidden state $\boldsymbol{h}_t$ captures the information from the entire history of tokens from a_1 up until a_t (Goldberg, 2016, p. 391). By having one hidden layer representation per input, an RNN is a network that stretches deep in time (Goldberg, 2016, p. 391-392).

[46] In computer vision, usually a two-dimensional convolution is applied over a two-dimensional data representation (Goldberg, 2016, p. 386).

[47] Note that at each time step, the same set of parameters, $\{\boldsymbol{W}_{hh}, \boldsymbol{W}_{ha}, \boldsymbol{b}_h, \boldsymbol{W}_{yh}, \boldsymbol{b}_y\}$, are applied (Goldberg, 2016, p. 391).

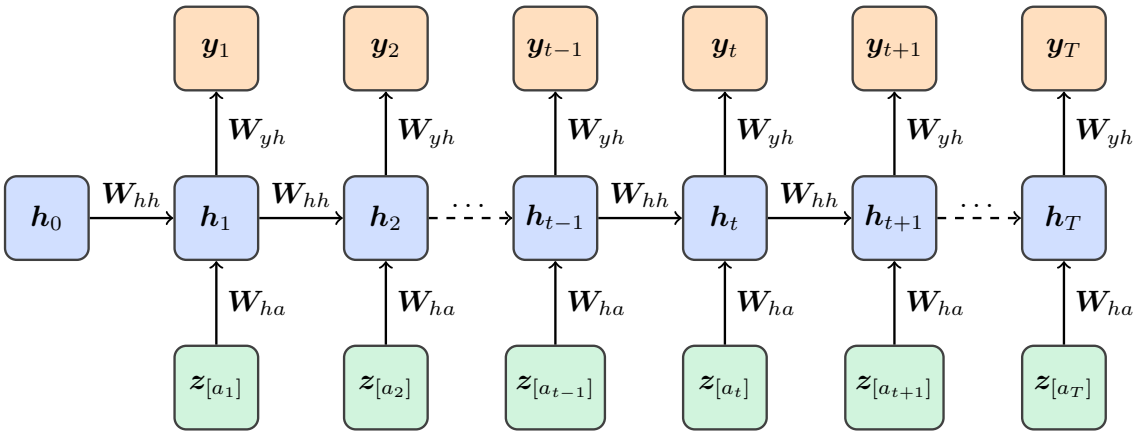


Figure 1.8: **Recurrent Neural Network.** This figure depicts a simple RNN as described in Equations 1.58 and 1.59 across T time steps.

In supervised learning, there are different ways how to impose the supervising information on the network (Goldberg, 2016, p. 392). First, if there is a document of T tokens, $d = (a_1, \ldots, a_t, \ldots, a_T)$, with a corresponding true label $\boldsymbol{y}$, then the prediction reached by the RNN for the last token $\hat{\boldsymbol{y}}_T$ can be compared against the true label $\boldsymbol{y}$ such that the loss L is $\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}_T)$ (Goldberg, 2016, p. 392). For example, if the task is multi-class sentiment classification, then σ_y is a softmax function and $\hat{\boldsymbol{y}}_T$ is a vector of predicted probabilities that serves as the RNN's sentiment prediction for the entire sentence which is compared to the true sentiment $\boldsymbol{y}$ (for a similar application see Wang et al., 2015, p. 1345-1346).

Second, RNNs can be used as encoders whose last hidden representation vector $\boldsymbol{h}_T$ captures the information from the entire sequence (Goldberg, 2016, p. 392, 394). $\boldsymbol{h}_T$, in turn, then can enter further model components (Goldberg, 2016, p. 392, 394). A prominent case here are encoder-decoder structures (Goldberg, 2016, p. 394). Sutskever et al. (2014) and Cho et al. (2014), for example, use an RNN-based encoder-decoder structure for machine translation. Here, an RNN encoder sequentially encodes the information from the input sequence in some language A to produce the last hidden state $\boldsymbol{h}_T$ (Cho et al., 2014, p. 1725; Sutskever et al., 2014, p. 3106). $\boldsymbol{h}_T$ then is the input to the RNN-based decoder that translates the input sequence token by token to another language O (Cho et al., 2014, p. 1725; Sutskever et al., 2014, p. 3106).

Third, in tasks in which the model is to produce an output for each token (e.g. sequence tagging tasks), there are true labels for each input token against which the RNN's predictions produced across the time steps $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_t, \ldots, \hat{\boldsymbol{y}}_T)$ are compared against (Goldberg, 2016, p. 393). The overall loss is the (weighted) sum of the losses at each of the time steps: $L = \sum_{t=1}^{T} \mathcal{L}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$ (Goldberg, 2016, p. 393).

In RNNs, the parameters are learned by backpropagating the gradients through the time

steps (Goodfellow et al., 2016, p. 378-380). This is known as backpropagation through time (BPTT) (Goldberg, 2016, p. 392). Because the gradients are multiplied with each other in the backpropagation process (see again Equations 1.49 to 1.51), and in RNNs the multiplication of the gradients can go over many time steps, the gradients can vanish (i.e. assume values close to 0) or explode (i.e. assume very large values) (Pascanu et al., 2013, p. 1311).[48] Exploding gradients can be easily handled by clipping techniques (Pascanu et al., 2013, p. 1315). Vanishingly small gradients make it difficult for learning signals to travel over long distances in time (Bengio et al., 1994, p. 161), which is why the simple RNNs presented in Equations 1.58 and 1.59 in practice fail to transmit information over sequences longer than 10 or 20 tokens (Goodfellow et al., 2016, p. 399).

To address this problem, special recurrent architectures, e.g. the long short-term memory (LSTM) model (Hochreiter & Schmidhuber, 1997) and the Gated Recurrent Unit (GRU) (Cho et al., 2014), have been developed (Goldberg, 2016, p. 378). These architectures direct the flow of gradients such that the vanishing gradient problem is mitigated and signals can travel over longer distances (Goldberg, 2016, p. 399-401). Instead of a single hidden operation (as in Equation 1.58), LSTMs have a memory cell that is composed of several operations (Goldberg, 2016, p. 399-400). LSTMs make use of gating mechanisms that regulate in how far incoming information updates the memory cell state that flows through the time steps and can keep information across longer token sequences (Goldberg, 2016, p. 399-400). GRUs also make use of gates but have a simpler architecture (Cho et al., 2014, p. 1726; Goldberg, 2016, p. 401).

Irrespective of whether the RNN architecture is a simple RNN as described in Equations 1.58 and 1.59 or whether the hidden units are more complex as in GRUs or LSTMs, RNNs can be stacked to form *deep RNNs* that have a multi-layered representation for each input (Graves, 2014, p. 3). In deep RNNs, the first RNN takes as an input $\boldsymbol{z}_{[a_t]}$ and produces $\boldsymbol{h}_t^1$ (Graves, 2014, p. 3-4). $\boldsymbol{h}_t^1$, in turn, serves as the input to the second RNN (Graves, 2014, p. 3-4). The hidden state of the lth RNN (with $l > 1$) at the tth time step then is (Graves, 2014, p. 4)[49]

$$\boldsymbol{h}_t^l = \sigma_h(\boldsymbol{W}_{h^l h^l} \boldsymbol{h}_{t-1}^l + \boldsymbol{W}_{h^l h^{l-1}} \boldsymbol{h}_t^{l-1} + \boldsymbol{b}_h^l) \tag{1.60}$$

Furthermore, recurrent architectures are relatively frequently applied in bidirectional form (Schuster & Paliwal, 1997; Graves, 2008; Goldberg, 2016, p. 395-396), and then are named biRNNs or biLSTMs respectively. In bidirectional recurrent models, there is one forward chain in which the hidden states are updated in a sequential manner from the first token

[48] If the activation function σ is the identity function, then the condition that the largest eigenvalue of $\boldsymbol{W}_{hh} < 1$, is a sufficient condition for the vanishing of the gradients as time steps $t \to \infty$ (Pascanu et al., 2013, p. 1311). It is necessary for the largest eigenvalue of $\boldsymbol{W}_{hh}$ to be > 1 for the gradients to explode as $t \to \infty$ (Pascanu et al., 2013, p. 1311). For nonlinear activation functions σ, one can say that vanishing gradients can be observed if the largest singular value of $\boldsymbol{W}_{hh}$ is small, and if the largest singular value of $\boldsymbol{W}_{hh}$ is large, gradients may explode (Pascanu et al., 2013, p. 1311-1312).

[49] Note that skip connections, that provide direct links between the inputs and the hidden states, can be additionally introduced such that $\boldsymbol{h}_t^l = \sigma_h(\boldsymbol{W}_{h^l h^l} \boldsymbol{h}_{t-1}^l + \boldsymbol{W}_{h^l h^{l-1}} \boldsymbol{h}_t^{l-1} + \boldsymbol{W}_{h^l a} \boldsymbol{z}_{[a_t]} + \boldsymbol{b}_h^l)$ (Graves, 2014, p. 4).

in a document's sequence, a_1, to the last token in the sequence, a_T (Goldberg, 2016, p. 396). And there is a separate and independent backward chain in which the hidden states are updated sequentially from the last token, a_T, to the first token, a_1 (Goldberg, 2016, p. 396). Thus, at each time step t, there is a forward state $\overrightarrow{\boldsymbol{h}_t}$ and a backward state $\overleftarrow{\boldsymbol{h}_t}$ (Goldberg, 2016, p. 396). Whereas the forward hidden state $\overrightarrow{\boldsymbol{h}_t}$ encodes the information accumulated from the past history of tokens a_1 to a_t, the backward hidden state $\overleftarrow{\boldsymbol{h}_t}$ encodes the information from the backward sequence of future tokens a_T to a_t (Goldberg, 2016, p. 395-396). $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ then are typically concatenated to obtain a single hidden state vector for time step t (Goldberg, 2016, p. 396):

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}] \tag{1.61}$$

$\boldsymbol{h}_t$ captures signals from the forward and the backward sequence. This can be advantageous compared to having information from only one direction (Graves, 2008, p. 22-23).[50]

Recurrent architectures have been frequently used as the backbone of NLP models across many NLP tasks (e.g. Mikolov et al., 2010; İrsoy & Cardie, 2014; Sutskever et al., 2014; Cho et al., 2014; Wang et al., 2015; Bahdanau et al., 2015; Lample et al., 2016; Peters et al., 2018a). In recent years, Transformer-based models (which are introduced in the next section) have been increasingly applied. Yet recurrent architectures still are used as model components in NLP (e.g. Liu et al., 2019a; Lei, 2021).

Recursive neural networks (RecNNs) can be conceived as a generalization of RNNs for tree structures (Goldberg, 2016, p. 402). An example of a tree structure that RecNNs operate on are parse trees (Goldberg, 2016, p. 402). A parse tree depicts the syntactic structure of a sentence (Smith, 2011, p. 9-11). For each node S in a parse tree, a representation $\boldsymbol{h}_s$ is learned (Goldberg, 2016, p. 402-404). $\boldsymbol{h}_s$ results as a function of the representation vectors for the children of node S (here named Q and R). For example (Socher et al., 2011, p. 153):

$$\boldsymbol{h}_s = \sigma(\boldsymbol{W}[\boldsymbol{h}_q; \boldsymbol{h}_r] + \boldsymbol{b}) \tag{1.62}$$

where $[\boldsymbol{h}_q; \boldsymbol{h}_r]$ is the concatenation of the representations for children nodes Q and R, σ is a nonlinear activation function, $\boldsymbol{b}$ is the bias vector, and $\boldsymbol{W}$ is a matrix of weights that linearly transforms the concatenation of $\boldsymbol{h}_q$ and $\boldsymbol{h}_r$ (Socher et al., 2011, p. 153). $\boldsymbol{W}$ may be the same in the entire network or it can be specific to the combination of labels assigned to nodes Q and R (Goldberg, 2016, p. 405). $\boldsymbol{h}_s$ thus encodes the information in the entire subtree over which node S resides (Goldberg, 2016, p. 404).

Widely known applications of RecNNs are Socher et al. (2011, 2013) and Iyyer et al. (2014). Socher et al. (2013), for example, apply a variant of RecNNs to sentiment analysis with movie reviews. They first parse each sentence and then learn for each node in each parse

[50]Note that $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ are generated in separate, independent processes. Thus $\boldsymbol{h}_t$ contains information from both directions, but the information from $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ are not combined or integrated in the sense that they influence each other. The vectors are simply concatenated.

tree a representation $\boldsymbol{h}_s$ that encodes the sentiment of the sub-sentence (Socher et al., 2013, p. 1631, 1634-1636).

1.2.3.7 Attention and the Transformer

Given a sequence of input tokens, $(a_1, \ldots, a_{t^*}, \ldots, a_{T^*})$, that are represented by a set of vectors, $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t^*}, \ldots, \boldsymbol{v}_{T^*})$, the attention mechanism computes a representation for a token a_t (this representation of a_t here is being denoted as $\boldsymbol{c}_t$) as the weighted sum of the input token representation vectors (Bahdanau et al., 2015, p. 3; Vaswani et al., 2017, p. 6000-6001):

$$\boldsymbol{c}_t = \sum_{t^*=1}^{T^*} \alpha_{t,t^*} \boldsymbol{v}_{t^*} \tag{1.63}$$

The attention weight α_{t,t^*} governs in how far the representation $\boldsymbol{c}_t$ incorporates information from (i.e. attends to) the representation of the t^*th input token $\boldsymbol{v}_{t^*}$. The core of the attention mechanism is the computation of α_{t,t^*} which is typically obtained as

$$\alpha_{t,t^*} = \frac{exp(score(\boldsymbol{q}_t, \boldsymbol{k}_{t^*}))}{\sum_{t^*=1}^{T^*} exp(score(\boldsymbol{q}_t, \boldsymbol{k}_{t^*}))} \tag{1.64}$$

(Vaswani et al., 2017, p. 6000-6001), where

- $\boldsymbol{k}_{t^*}$ is a representation of the t^*th input token a_{t^*} (Vaswani et al., 2017, p. 6002; Galassi et al., 2021, p. 4294). (In some implementations of the attention mechanism, $\boldsymbol{v}_{t^*}$ and $\boldsymbol{k}_{t^*}$ are two different representations of the same input token, in other implementations $\boldsymbol{v}_{t^*}$ and $\boldsymbol{k}_{t^*}$ are the same vector representation (Galassi et al., 2021, p. 4294).)
- $\boldsymbol{q}_t$ is the initial, pre-update representation of token a_t (Vaswani et al., 2017, p. 6002; Kobayashi et al., 2020, p. 7058).
- *score* is a function that produces as an output an indicator of the similarity between the t^*th input token representation $\boldsymbol{k}_{t^*}$ and token a_t's initial representation $\boldsymbol{q}_t$ (Bahdanau et al., 2015, p. 3; Luong et al., 2015, p. 1414). There is a vast set of scoring functions that can be applied (for an overview see Galassi et al., 2021, p. 4298). A common scoring function used is the scaled dot product: $(\boldsymbol{q}_t^\top \boldsymbol{k}_{t^*})/\sqrt{|\boldsymbol{k}_{t^*}|}$ (Vaswani et al., 2017, p. 6001). The higher the similarity between the input token representation vector $\boldsymbol{k}_{t^*}$ with a_t's initial representation vector $\boldsymbol{q}_t$, the higher the attention weight α_{t,t^*}.

$\boldsymbol{v}_{t^*}$, $\boldsymbol{k}_{t^*}$, and $\boldsymbol{q}_t$ are often named value, key, and query vector respectively (Vaswani et al., 2017, p. 6000-6001; Galassi et al., 2021, p. 4294).[51]

[51] Note that different implementations of the attention mechanism differ with regard to how key, query, and value vectors are obtained. Bahdanau et al. (2015) apply encoder-decoder attention in a recurrent

When the attention mechanism was introduced to NLP in the context of neural machine translation (NMT), it was incorporated within an encoder-decoder structure (Bahdanau et al., 2015). In general, an encoder-decoder operates as follows: Given a set of original data inputs, an encoder converts the inputs into a hidden representation and the decoder takes the representation and decodes data output from it. In NMT, a variable length sequence of tokens in language A, $(a_1, \ldots, a_{t^*}, \ldots, a_{T^*})$, constitutes the data input (Cho et al., 2014, p. 1724-1725). The encoder encodes the input into representation vectors and the decoder, based on the encoding provided by the encoder, produces a variable-length output of tokens in language O, $(o_1, \ldots, o_t, \ldots, o_T)$ (Cho et al., 2014, p. 1724-1725).[52] In doing so, the decoder proceeds in an autoregressive manner (Vaswani et al., 2017, p. 5999). This is, when generating the prediction for the tth output token, the decoder not only processes the representation vectors, that have been produced by the encoder, but also processes the previous output tokens $o_{<t}$ (Bahdanau et al., 2015, p. 3; Vaswani et al., 2017, p. 5999).

Within this encoder-decoder structure, the attention mechanism can enable the decoder to attend to *all* input token representations produced by the encoder. This is, the input tokens $(a_1, \ldots, a_{t^*}, \ldots, a_{T^*})$ are encoded by the encoder into representations $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t^*}, \ldots, \boldsymbol{v}_{T^*})$ and the decoder then generates the representation $\boldsymbol{c}_t$ of output token o_t as a weighted sum over the encoder-generated representations $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t^*}, \ldots, \boldsymbol{v}_{T^*})$ (see Figure 1.9 and Bahdanau et al., 2015, p. 3).

Another form of attention is self-attention. Here, the input tokens attend to themselves. If $(a_1, \ldots, a_t, \ldots, a_T)$ is a sequence of tokens that is being processed and if $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t, \ldots, \boldsymbol{v}_T)$ are the respective token representations, then for each token a_t in this sequence, a new representation $\boldsymbol{c}_t$ can be learned that captures information encoded in the representations of the tokens that are in the *same* sequence as a_t (see Figure 1.10 and Vaswani et al., 2017, p. 6001-6002).

The attention mechanism generates a contextualized token representation. This is, $\boldsymbol{c}_t$ is a representation of a token a_t (and not a term z_u) (Pilehvar & Camacho-Collados, 2020,

architecture and use the previous hidden state $\boldsymbol{h}_{t-1}$ as an initial representation of token a_t. Accordingly, $\boldsymbol{q}_t$ here is $\boldsymbol{h}_{t-1}$. The value vectors $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t^*}, \ldots, \boldsymbol{v}_{T^*})$ are the sequence of hidden states produced by a recurrent encoder (Bahdanau et al., 2015, p. 3). Bahdanau et al. (2015) furthermore make no distinction between key and value vectors. In the self-attention mechanism as implemented by Vaswani et al. (2017), key, query, and value vectors are linear transformations of input vectors: If a sequence of tokens $(a_1, \ldots, a_t, \ldots, a_T)$ is represented as $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t, \ldots, \boldsymbol{z}_T)$ (e.g. as a sequence of word embeddings or a sequence of already updated token representations), then for each token a_t, the corresponding input embedding $\boldsymbol{z}_t$ is transformed into a key vector $\boldsymbol{k}_t$, a query vector $\boldsymbol{q}_t$, and value vector $\boldsymbol{v}_t$ via (Vaswani et al., 2017, p. 6002):

$$\boldsymbol{k}_t = \boldsymbol{W}_k \boldsymbol{z}_t \qquad \boldsymbol{q}_t = \boldsymbol{W}_q \boldsymbol{z}_t \qquad \boldsymbol{v}_t = \boldsymbol{W}_v \boldsymbol{z}_t \tag{1.65}$$

[52] Note that the length of the input and output sequences may differ: *'He is giving a speech.'* translated to German is *'Er hält eine Rede.'*. The task of machine translation thus is one of sequence-to-sequence modeling, where the sequences are of variable length (Cho et al., 2014, p. 1724-1725).

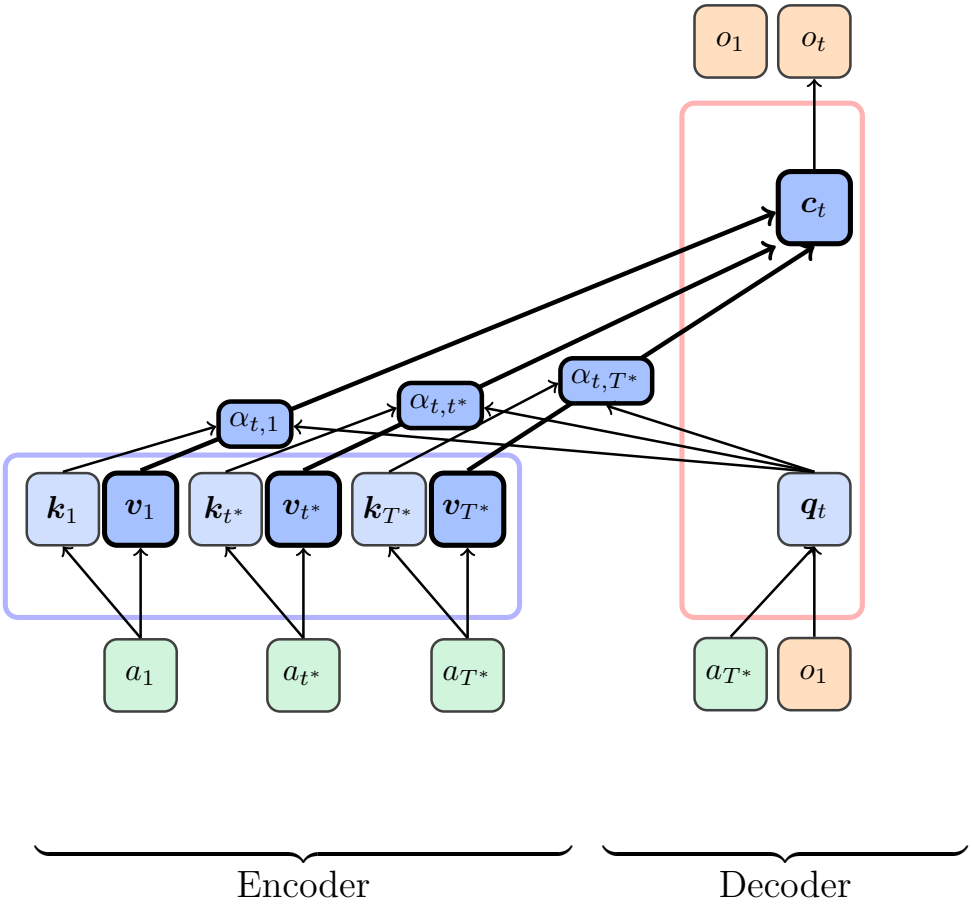


Figure 1.9: **Attention Within an Encoder-Decoder Structure.** This figure depicts the attention mechanism incorporated within an encoder-decoder structure. The input tokens (a_1, a_{t^*}, a_{T^*}) are encoded by the encoder into value vectors $(\boldsymbol{v}_1, \boldsymbol{v}_{t^*}, \boldsymbol{v}_{T^*})$ and key vectors $(\boldsymbol{k}_1, \boldsymbol{k}_{t^*}, \boldsymbol{k}_{T^*})$ (Vaswani et al., 2017, p. 6002). When the decoder is making a prediction for the tth output token o_t, vector $\boldsymbol{c}_t$ (on which the prediction for output token o_t will be based on) is computed via the attention mechanism as a weighted sum over the encoder-generated representations $(\boldsymbol{v}_1, \boldsymbol{v}_{t^*}, \boldsymbol{v}_{T^*})$ (see also Equation 1.63) (Bahdanau et al., 2015, p. 3; Vaswani et al., 2017, p. 6000-6002). The weight α_{t,t^*} results from a function assessing the similarity between key vector $\boldsymbol{k}_{t^*}$ and query vector $\boldsymbol{q}_t$ (see also Equation 1.64) (Bahdanau et al., 2015, p. 3-4). $\boldsymbol{q}_t$ is the initial, pre-update representation of output token o_t (Vaswani et al., 2017, p. 6002; Kobayashi et al., 2020, p. 7058). When making a prediction, an autoregressive decoder takes as an additional input the sequence of preceding output tokens (Vaswani et al., 2017, p. 5999). The pre-update vector $\boldsymbol{q}_t$ thus typically incorporates information from so far predicted output tokens (see e.g. Bahdanau et al., 2015, p. 3; Vaswani et al., 2017, p. 6000). The previous output tokens here are (a_{T^*}, o_1), which is why $\boldsymbol{q}_t$ is depicted to result from (a_{T^*}, o_1). (The last token of an input sequence, a_{T^*}, usually is an end-of-sequence symbol $<$ `EOS` $>$, that signals to the encoder the end of the sequence (Sutskever et al., 2014, p. 3106). Such an end-of-sequence symbol (or some other symbol that not only can be used to indicate the end but also the start of a sequence) then is used as a first input to the decoder (see e.g. Sutskever et al., 2014, p. 3105). This is why a_{T^*} here serves as an input to the decoder.)

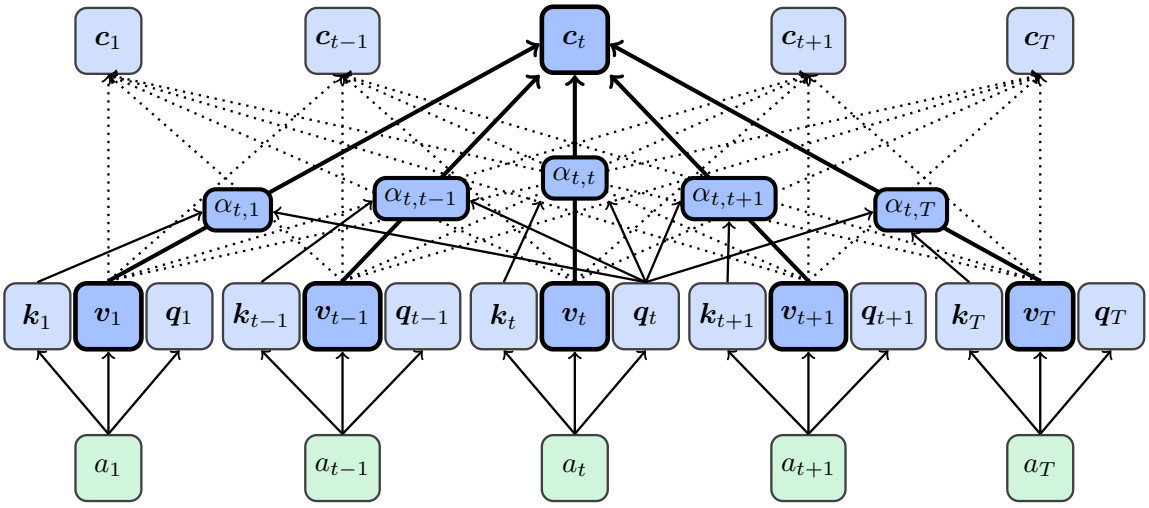


Figure 1.10: **Self-Attention.** This figure depicts the self-attention mechanism. Each token in the input sequence $(a_1, a_{t-1}, a_t, a_{t+1}, a_T)$ is transformed into a key vector, a value vector, and a query vector (Vaswani et al., 2017, p. 6002). Then, for each token in the sequence, an updated representation is created via the self-attention mechanism (see also Equation 1.63). Here, the self-attention mechanism is visualized for token a_t. Each attention weight results from a function that indicates the degree of similarity between an input token's key vector and a_t's query vector $\boldsymbol{q}_t$ (see also Equation 1.64) (Vaswani et al., 2017, p. 6001). $\boldsymbol{q}_t$ serves as a pre-update representation of a_t (Vaswani et al., 2017, p. 6002; Kobayashi et al., 2020, p. 7058).

p. 82). $\boldsymbol{c}_t$ is contextualized in the sense that it is a function of input token representations $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t^*}, \ldots, \boldsymbol{v}_{T^*})$ (Pilehvar & Camacho-Collados, 2020, p. 82). This implies that each time token a_t occurs within another textual sequence, it obtains another representation vector. The contextualized nature of the representation $\boldsymbol{c}_t$ allows for $\boldsymbol{c}_t$ to encode the context-dependent meaning of token a_t (i.e. the meaning that arises from the textual context token a_t is embedded in) and allows for $\boldsymbol{c}_t$ to encode syntactic or semantic dependencies between token a_t and input tokens $(a_1, \ldots, a_{t^*}, \ldots, a_{T^*})$ (Jawahar et al., 2019; Reif et al., 2019; Tenney et al., 2019; Kobayashi et al., 2020). Because the attention mechanism computes $\boldsymbol{c}_t$ as a weighted sum over all input token representation vectors—and thus how each input token representation vector $\boldsymbol{v}_{t^*}$ can contribute to $\boldsymbol{c}_t$ is independent of its position within the sequence—, the attention mechanism can capture information on syntactic or semantic dependencies regardless of the distance between tokens (Vaswani et al., 2017, p. 5999). This stands in contrast to recurrent architectures that have problems transmitting information over longer token sequences (see Section 1.2.3.6).

Early research articles that applied the attention mechanism for NMT used RNNs as their encoder and decoder (Bahdanau et al., 2015; Luong et al., 2015). In search of a computationally more efficient architecture in which the tokens of one sequence can be processed in parallel rather than one after another, Vaswani et al. (2017) invented the Transformer. This invention had a large impact on the field of NLP. Whilst until then recurrent architectures were in many circumstances regarded as the most adequate tool for the processing of natural language, now Transformer-based models constituted an even more effective tool (Pilehvar & Camacho-Collados, 2020, p. 23).

The Transformer is comprised of a stack of several encoders followed by a stack of several decoders (Vaswani et al., 2017, p. 6000). (In the original article there are six encoders and six decoders (Vaswani et al., 2017, p. 6000).) The first encoder takes as an input a sequence of positional word embeddings, conducts self-attention, and then feeds the updated token representations into an FNN that then passes the again updated representations to the next encoder (Vaswani et al., 2017, p. 6000, 6002). The next encoder applies the same operations (self-attention followed by an FNN), passes the newly updated token representations to the next encoder, and so on (Vaswani et al., 2017, p. 6000). Each decoder comprises a self-attention layer, followed by encoder-decoder attention, followed by an FNN (Vaswani et al., 2017, p. 6000, 6002). To make sure that the decoders work in an autoregressive manner, masking is performed in the self-attention mechanisms of the decoders (Vaswani et al., 2017, p. 6002). This is, when computing vector $\boldsymbol{c}_t$ for output token o_t, $\boldsymbol{c}_t$ can only attend to (and thus can only integrate information from) previous output tokens $o_{<t}$ but not from following output tokens $o_{>t}$ (Vaswani et al., 2017, p. 6000, 6002).[53]

The Transformer with its encoder-decoder structure has been developed for the sequence-

[53]Several details of the Transformer architecture (such as residual learning, layer normalization, and multi-head attention) are not mentioned here. A more thorough and detailed introduction to the Transformer is given in this dissertation's article *Introduction to neural transfer learning with Transformers for social science text analysis*.

to-sequence modeling task of machine translation. Today, a huge spectrum of models that make use of elements of the Transformer architecture and are applied to all sorts of NLP tasks exist. Transformer encoders are often used as the architectural basis in autoencoding models, that tend to perform well on natural language understanding tasks (see e.g. Devlin et al., 2019; Liu et al., 2019b; Lan et al., 2020). Transformer decoders can be the backbone of autoregressive models, which are naturally suited for natural language generation tasks (Radford et al., 2018, 2019). The full Transformer encoder-decoder structure is made use of in sequence-to-sequence models that are applied to sequence-to-sequence tasks such as translation or text summarization (Lewis et al., 2020; Raffel et al., 2020). (For a more detailed introduction to the Transformer and the concepts mentioned here see the article *Introduction to neural transfer learning with Transformers for social science text analysis* in this dissertation.)

1.2.3.8 Transfer Learning

In their survey on transfer learning, Pan & Yang (2010, p. 1346-1347) make the following definitions:[54]

> "a *domain* $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $p(\boldsymbol{X})$, where $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \in \mathcal{X}$. For example, if our learning task is document classification, and each term is taken as a binary feature, then $\mathcal{X}$ is the space of all term vectors, $\boldsymbol{x}_i$ is the ith term vector corresponding to some document, and $\boldsymbol{X}$ is a particular learning sample. [...] Given a specific domain, $\mathcal{D} = \{\mathcal{X}, p(\boldsymbol{X})\}$, a *task* consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$ (denoted by $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$), which is not observed but can be learned from the training data, which consist of pairs $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \boldsymbol{X}$ and $y_i \in \mathcal{Y}$".

Furthermore, the distinction is made between the source task $\mathcal{T}_\mathtt{S}$ and the source domain $\mathcal{D}_\mathtt{S}$, on the one hand, and the target task $\mathcal{T}_\mathtt{T}$ and the target domain $\mathcal{D}_\mathtt{T}$, on the other hand, (Pan & Yang, 2010, p. 1346-1347). A researcher is actually interested in conducting the target task in the target domain (Pan & Yang, 2010, p. 1346). In doing so, a researcher may make use of information from a source task in a source domain (Pan & Yang, 2010, p. 1346-1347).

Given these definitions, transfer learning can be defined as a learning procedure in which knowledge from a source task $\mathcal{T}_\mathtt{S}$ and source domain $\mathcal{D}_\mathtt{S}$ is used with the aim to better approximate the target task function $f_\mathtt{T}$ in the target domain $\mathcal{D}_\mathtt{T}$, where either $\mathcal{T}_\mathtt{S} \neq \mathcal{T}_\mathtt{T}$ or $\mathcal{D}_\mathtt{S} \neq \mathcal{D}_\mathtt{T}$ (Pan & Yang, 2010, p. 1347).

In the usual supervised learning setting, for each task in each domain, a new model is trained based on a new training data set—and the training set instances are (assumed

[54] The notation used in Pan & Yang (2010, p. 1346-1347) has been adapted to match the notation in this dissertation.

to be) i.i.d. observations from $p(\boldsymbol{x}, y)$ (Ruder, 2019a, p. 42). This learning procedure, however, runs into problems if for a given target task or target domain of interest there are too few labeled training instances available to train a model whose predictions are reliable and satisfyingly accurate (Ruder, 2019a, p. 3, 42). The idea of transfer learning then is to take the knowledge that has been acquired from another task or another domain and to use this knowledge in the learning process of the target task in the target domain (Ruder, 2019a, p. 42-44). The expectation is that the transferred knowledge (e.g. because it comprises information that captures general aspects and thus travels well beyond the source task and domain) is useful for the target task and domain and hence a model that is trained via transfer learning has a better generalization performance than models that are trained on data from the target task and domain alone (Ruder, 2019a, p. 3). In NLP, the *knowledge* that is transferred from source task or domain to target task or domain are typically learned model parameters of a trained model (Ruder, 2019a, p. 43).

Ideas, motivations, and procedures for transfer learning have been around for decades within the machine learning community (Caruana, 1993; Pan & Yang, 2010, p. 1346). In the field of NLP, Collobert & Weston (2008) applied transfer learning in their notable 2008 paper. Moreover, the NLP procedure of utilizing word embeddings that have been trained on large general corpora (Mikolov et al., 2013a,c; Pennington et al., 2014) for downstream target tasks also constitutes a form of transfer learning (Ruder, 2019a, p. 3). Yet only during the last few years transfer learning became an essential and nearly ubiquitous learning practice in NLP (Bommasani et al., 2021, p. 5). What is more, in the whole field of AI research, transfer learning has emerged as an important core learning procedure (Bommasani et al., 2021, p. 5). It is even considered to constitute a new "paradigm for building artificial intelligence (AI) systems" (Bommasani et al., 2021, p. 3).

In his Ph.D. thesis on *Introduction to neural transfer learning for natural language processing* Ruder (2019a) distinguishes four types of transfer learning:

- **Sequential transfer learning.** In sequential transfer learning, source and target tasks differ (Ruder, 2019a, p. 45). The tasks differ in the sense that the space of values the output variable can take in the source task is not the same space of values the output variable can take in the target task; i.e. $\mathcal{Y}_\mathtt{S} \neq \mathcal{Y}_\mathtt{T}$ (Ruder, 2019a, p. 45). For example, the source task could be language modeling, and then at each step the source task would be to predict the next textual token out of a vocabulary of U features given the sequence of previous tokens such that at each step $y_i \in \{z_1, \ldots, z_u, \ldots, z_U\}$. The target task, in contrast, could be a binary classification task in which the set of values for the output variable would be $y_i \in \{0, 1\}$. As $\mathcal{Y}_\mathtt{S} \neq \mathcal{Y}_\mathtt{T}$ consequently also $f_\mathtt{S} \neq f_\mathtt{T}$ (Ruder, 2019a, p. 45).

  The learning procedure in sequential transfer learning is such that a model first is trained on the source task (pretraining phase) and then the trained model (including its learned parameters) is used as an input for training on the target task (adaptation phase) (Ruder, 2019a, p. 64). The general idea is to choose a source task that is suitable for learning a model that functions as a very general, almost universal

language representation tool in order to then use this hopefully well-generalizing model as an effective basis for the learning process on the target task (Ruder, 2019a, p. 64). A once pretrained source model can be used as an input to not only one but various different target tasks (Ruder, 2019a, p. 63-64).

In order to train such a highly general language representation model, a large number of training examples and a suitable pretraining task are required (Ruder, 2019a, p. 65). In NLP, many language representation source models have been pretrained in a self-supervised fashion via (masked) language modeling tasks on sets of large corpora (such as the English Wikipedia, the BooksCorpus Dataset (Zhu et al., 2015), or web document-based corpora) (e.g. Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019b; Radford et al., 2019; Beltagy et al., 2020).

Whilst the process of pretraining requires vast amounts of training data and consumes considerable amounts of computational and time resources (for a comparative overview see Aßenmacher & Heumann, 2020, p. 6), the adaptation to a target task requires comparatively few resources (Ruder, 2019a, p. 63, 64). In the adaptation phase, a very common procedure in NLP is to *fine-tune* a pretrained language representation model to the target task (Ruder, 2021b).[55] Fine-tuning typically implies that the pretrained model is retained and simply added an output layer that matches the target task (Ruder, 2019a, p. 77). Then, the pretrained model with the target-task specific output layer is trained on the target task training data. During training the gradients are allowed to backpropagate through the layers of the pretrained model, thereby adapting the pretrained parameters to the target task (Ruder, 2019a, p. 77-79).

When applying sequential transfer learning, the same prediction performance on the target task can be achieved with a substantively lower proportion of training examples than when conducting the usual supervised learning procedure, in which no knowledge is transferred and a model is trained from scratch using only the target task training data (Howard & Ruder, 2018, p. 334). In settings in which only a limited number of labeled target task training set instances is available, transfer learning—by leveraging information acquired in pretraining—allows researchers to obtain higher generalization performances than when not using transfer learning (Howard & Ruder, 2018, p. 334-335). Across various NLP tasks, sequential transfer learning with pretrained models (many of which are based on the Transformer architecture) has led to substantive enhancements in the prediction performances that NLP models are able to reach (Ruder, 2019a, p. 75; Bommasani et al., 2021, p. 22-23). Transformer-based models that are used in a sequential transfer learning setting are presented

[55] An alternative yet meanwhile less frequently used procedure is feature extraction (Ruder, 2019a, p. 77). Here, the pretrained model parameters are extracted from the source model to function as the input to a second, independent model which is trained on the target task in the adaptation phase (Ruder, 2019a, p. 77). In contrast to fine-tuning, in feature extraction, the pretrained parameters remain unchanged during adaptation (Ruder, 2019a, p. 77, 79).

and applied in this dissertation's article *Introduction to neural transfer learning with Transformers for social science text analysis*.

- **Multitask learning.** In multitask learning (Caruana, 1993, 1997), source and target tasks differ in the same way as they differ in sequential transfer learning (i.e. $\mathcal{Y}_{\mathtt{S}} \neq \mathcal{Y}_{\mathtt{T}}$), but source and target tasks are learned simultaneously rather than sequentially (Ruder, 2019a, p. 44-45). (As learning is conducted simultaneously, in the context of multitask learning, the source tasks are also called auxiliary tasks.) There are two main ways of how multitask learning is implemented with deep neural networks in practice: hard vs. soft parameter sharing (Ruder, 2019a, p. 48). In hard parameter sharing, a single deep neural network is trained that has one separate output layer for each task (Ruder, 2019a, p. 48). This implies that the target task and the auxiliary tasks share the same set of hidden layers (Ruder, 2019a, p. 48). The hidden layer representations thus form the basis for not only one but several tasks and thereby are forced to be more general rather than task-specific (Ruder, 2019a, p. 48). This makes overfitting less likely and tends to improve generalization (Ruder, 2019a, p. 48). In soft parameter sharing, a separate model is trained for each task, but the parameters of each model are regularized to become not too different from each other (e.g. by adding to the loss function the L^2 norm between parameters as a regularizing component) (Duong et al., 2015, p. 846; Ruder, 2019a, p. 49).

- **Domain adaptation.** In domain adaptation, source and target domains are characterized by different distributions over the space of textual features and output labels (Kouw & Loog, 2019, p. 3; Ruder, 2019a, p. 44-45). This is, source and target domains share the same feature-label space $\mathcal{X} \times \mathcal{Y}$ but are characterized by different distributions over this shared feature-label space, i.e. $p_{\mathtt{S}}(\boldsymbol{x}, y) \neq p_{\mathtt{T}}(\boldsymbol{x}, y)$ (Kouw & Loog, 2019, p. 3; Ruder, 2019a, p. 44-45). For example, documents in the source and target domains could be about different topics and thus have different distributions over the feature space (Ruder, 2019a, p. 44-45). Or, the type of documents used for the source task could be different from those of the target task and hence source and target documents could exhibit different linguistic styles (which in turn are reflected in different distributions). The goal in domain adaptation is to leverage knowledge in the training data of the source domain for the learning process in the target domain in which none (or only a small set of) the data are labeled (Ruder, 2019a, p. 86). In NLP, domain adaptation has been addressed by (1) approaches that operate on the basis of the feature distributions (and e.g. try to learn representations in a united lower-dimensional space) (Ruder, 2019a, p. 87-93), (2) approaches that weight or select instances from the source domain based on the degree to which they are considered relevant for the target domain (Ruder, 2019a, p. 93-96), and (3) approaches that employ self-training or multi-view training (see Ruder, 2019a, p. 96-99).

  Several times in this dissertation, random oversampling is employed. In random oversampling, training documents that belong to classes that make up a small share within the training data are randomly sampled with replacement and appended to

the training data set such that the distribution of class labels within the training data is transformed into a more balanced distribution (Brownlee, 2020b). Oversampling can be regarded as one technique to address domain adaptation (Jiang, 2008, p. 4): In domain adaptation, the aim is to minimize the expected loss in the target domain (that has underlying joint distribution $p_{\mathtt{T}}(\boldsymbol{x}, y)$) (Jiang, 2008, p. 3). Hence, the aim is to minimize

$$\mathcal{R}_{\mathtt{T}}(\tilde{f}) = \int \int \mathcal{L}(y, \tilde{f}(\boldsymbol{x})) p_{\mathtt{T}}(\boldsymbol{x}, y) \, \mathsf{d}\boldsymbol{x} \, \mathsf{d}y \tag{1.66}$$

(see also Equation 1.1). But $p_{\mathtt{T}}(\boldsymbol{x}, y)$ is unknown and training data are only available for the source domain with underlying joint distribution $p_{\mathtt{S}}(\boldsymbol{x}, y)$ (Jiang, 2008, p. 3). The following re-formulations show that—under the assumption that the conditional distribution of $\boldsymbol{x}$ given class label y is the same in the source and the target domain, i.e. $p_{\mathtt{S}}(\boldsymbol{x}|y) = p_{\mathtt{T}}(\boldsymbol{x}|y)$—weighting instances from the source training data with $\frac{p_{\mathtt{T}}(y)}{p_{\mathtt{S}}(y)}$ during training allows for domain adaptation from source to target domain (see Jiang, 2008, p. 3-4; Kouw & Loog, 2019, p. 4-5, 7): Multiplying and dividing Equation 1.66 by $p_{\mathtt{S}}(\boldsymbol{x}, y)$ gives

$$\mathcal{R}_{\mathtt{T}}(\tilde{f}) = \int \int \mathcal{L}(y, \tilde{f}(\boldsymbol{x})) \frac{p_{\mathtt{T}}(\boldsymbol{x}, y)}{p_{\mathtt{S}}(\boldsymbol{x}, y)} p_{\mathtt{S}}(\boldsymbol{x}, y) \, \mathsf{d}\boldsymbol{x} \, \mathsf{d}y \tag{1.67}$$

$$\mathcal{R}_{\mathtt{T}}(\tilde{f}) = \int \int \mathcal{L}(y, \tilde{f}(\boldsymbol{x})) \frac{p_{\mathtt{T}}(\boldsymbol{x}|y) p_{\mathtt{T}}(y)}{p_{\mathtt{S}}(\boldsymbol{x}|y) p_{\mathtt{S}}(y)} p_{\mathtt{S}}(\boldsymbol{x}, y) \, \mathsf{d}\boldsymbol{x} \, \mathsf{d}y \tag{1.68}$$

and assuming that $p_{\mathtt{S}}(\boldsymbol{x}|y) = p_{\mathtt{T}}(\boldsymbol{x}|y)$ gives

$$\mathcal{R}_{\mathtt{W}}(\tilde{f}) = \int \int \mathcal{L}(y, \tilde{f}(\boldsymbol{x})) \frac{p_{\mathtt{T}}(y)}{p_{\mathtt{S}}(y)} p_{\mathtt{S}}(\boldsymbol{x}, y) \, \mathsf{d}\boldsymbol{x} \, \mathsf{d}y \tag{1.69}$$

(Jiang, 2008, p. 3-4; Kouw & Loog, 2019, p. 4-5, 7). Thus, when approximating the risk $\mathcal{R}_{\mathtt{T}}(\tilde{f})$ with the empirical risk $\mathcal{R}_{\mathtt{W},emp}(\tilde{f})$ on the basis of training data instances drawn from source domain distribution $p_{\mathtt{S}}(\boldsymbol{x}, y)$ (see again Equation 1.2), then adaptation to the target domain can be achieved by weighting each training instance from the source domain with $\frac{p_{\mathtt{T}}(y)}{p_{\mathtt{S}}(y)}$ (Jiang, 2008, p. 4; Kouw & Loog, 2019, p. 7).

The connection between domain adaptation via instance weighting and random oversampling is as follows: Random oversampling alters the marginal distribution of y in the data. The marginal distribution $p_{\mathtt{T}}(y)$ in the oversampled training data no longer equals $p_{\mathtt{S}}(y)$ in the source training data. Yet it can be safely assumed that $p_{\mathtt{S}}(\boldsymbol{x}|y) = p_{\mathtt{T}}(\boldsymbol{x}|y)$. Oversampling thus has a similar effect as weighting each instance with $\frac{p_{\mathtt{T}}(y)}{p_{\mathtt{S}}(y)}$ (Kouw & Loog, 2019, p. 7).

- **Cross-lingual learning.** In cross-lingual learning, source and target domains also differ, but this time source and target domains do not share the same feature space because they come from different languages (i.e. $\mathcal{X}_{\mathtt{S}} \neq \mathcal{X}_{\mathtt{T}}$) (Ruder, 2019a, p. 44-45). Cross-lingual learning is highly important to address the discrepancies that

exist between low- and high-resource languages regarding the availability and performance of NLP technologies (Ruder, 2019b). High-resources languages are a small number of languages for which large amounts of labeled and unlabeled text data are available (Joshi et al., 2020, p. 6282-6285). For low-resource languages, which constitute the large majority of languages spoken in the world, only small amounts of labeled and unlabeled text data exist (Joshi et al., 2020, p. 6282-6285). As the usage of vast amounts of text data during (pre)training is key to the generalization performance of NLP models, speakers of high-resource languages benefit from various high-performing NLP technologies, whereas for low-resource languages these technologies either are not available or only in mediocre quality (Ruder, 2019b; Joshi et al., 2020, p. 6282). The NLP community is becoming aware of this problem, yet so far the primary focus of NLP researchers has been on engineering methods for the processing of spoken and written English (Ruder, 2020).[56]

Given a (possibly high-resource) language A in which labeled and unlabeled data relevant for a given task exist and given a second (possibly low-resource) language B in which only unlabeled data are accessible, one way for cross-lingual transfer learning is to proceed as follows (Ruder, 2019b): First, cross-lingual representations are learned. Second, the cross-lingual representations serve as the backbone of a model that is trained on the labeled training data in language A in order to learn parameters that are functional in solving the task. Finally, the entire trained model comprising the cross-lingual representations and the task-functional parameters is applied without further adaptation on unlabeled data instances in language B to generate predictions for these instances in language B.[57]

1.2.3.9 Representations

After this overview of deep learning architectures and transfer learning, this subsection in more detail focuses on the real-valued vector representations that deep neural networks operate on and produce. In doing so, this subsection fits many aspects that have been mentioned up to this point into an overall picture.

Neural networks internally represent entities as real-valued vectors (see, for example, Equations 1.32 to 1.35 that describe the FNN). In NLP, the represented entities can be the terms in a vocabulary (then called word embeddings) (Pilehvar & Camacho-Collados, 2020, p. 5). But the entities also, for example, can be single characters (Akbik et al., 2018), subwords

[56] This narrow focus on the English language (and perhaps a few other high-resource languages such as Spanish and German) is not only problematic because of the availability of NLP technologies for practical use but also from a machine learning perspective: Model architectures and modeling procedures that have been identified as performing better than others are not necessarily language agnostic and thus need not be suitable for languages with substantively different structural properties than English (Ruder, 2020).

[57] This last step is referred to as zero-shot cross-lingual transfer (Ruder, 2019b; Wu & Dredze, 2019; Nozza et al., 2020).

(Bojanowski et al., 2017), word senses and synsets (Rothe & Schütze, 2015; Kumar et al., 2019),[58] the nodes in a network (Kipf & Welling, 2017), sentences, documents (Le & Mikolov, 2014; Reimers & Gurevych, 2019), or tokens (McCann et al., 2018; Peters et al., 2018a).

The representation vectors position the entities within a continuous space and thereby encode some information about the entities. For example, when working with word embeddings, researchers usually seek to have representation vectors that encode the terms' meanings such that the continuous space is a semantic space and the relative positioning of terms within the semantic continuous space reflects the semantic similarity of terms (see again Figure 1.3b) (Pilehvar & Camacho-Collados, 2020, p. 4-6).

To give a formal expression: Given a set of U entities (e.g. a set of U vocabulary terms) denoted by $\{z_1, \ldots, z_u, \ldots, z_U\}$, each entity z_u is represented by a vector $\boldsymbol{z}_u$ of dimensionality K whose elements are real-valued: $\boldsymbol{z}_u \in \mathbb{R}^K$. The set of entities thus are represented by a set of vectors $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_u, \ldots, \boldsymbol{z}_U\}$ that position the entities in a K-dimensional Euclidean space (which here is referred to as continuous space).

In Section 1.2.2 it was pointed out that when researchers apply conventional machine learning methods, each feature constitutes a dimension of the representational space and is represented as a one-hot encoded vector (Goldberg, 2016, p. 349-351). As the number of textual features in any NLP application (even after feature exclusion and normalization operations) is usually very high, the usage of one-hot encodings implies that feature representations are sparse and have a high dimensionality (Goldberg, 2016, p. 349-351). Deep learning models, in contrast, operate on features that are represented as vectors within a continuous representational space of dimensionality K (where K is typically much smaller than the number of textual features U) (Goldberg, 2016, p. 349-351).[59] The representation of features as vectors in a continuous space implies low-dimensional and dense feature representations (Goldberg, 2016, p. 349-351). Low-dimensional, dense representations facilitate generalization (Goldberg, 2016, p. 351). Moreover, the values of the elements of the representation vectors are computed from model parameters that are learned during training (Goldberg, 2016, p. 349). Hence, neural networks *learn* representations.

In NLP, early representation learning efforts were directed at learning real-valued vector representations for vocabulary terms, known as word embeddings (see Bengio et al., 2003). Seminal early model architectures to learn word embeddings are Global Vectors (GloVe) (Pennington et al., 2014) and word2vec (which comprises two distinct but related models named continuous bag-of-words (CBOW) and Skip-gram (Mikolov et al., 2013a)). All of

[58] Synsets are the building blocks of the WordNet database (Miller, 1995). In WordNet, "[n]ouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept" (Princeton University, 2010).

[59] Depending on the model architecture, the embedding dimension (i.e. the dimensionality of the representation vectors) currently can be expected to range between 100 (Mikolov et al., 2013a; Pennington et al., 2014) and 1600 (Radford et al., 2019; Li et al., 2020d). Yet in the extreme case of the GPT-3 model (Brown et al., 2020) the embedding dimension is as large as 12,288.

these procedures learn an embedding for a vocabulary term z_u, that corresponds to target word a_t, by making use of context words that occur in a window around a_t (Pennington et al., 2014, p. 1533-1535). By choosing to use word contexts for learning word embeddings, these procedures employ the distributional hypothesis (Firth, 1957) which states that a term's meaning can be learned from the context of words it typically co-occurs with and semantically similar terms tend to occur in similar contexts—which is why the embedding vectors of semantically similar terms are expected to be similar (Goldberg, 2016, p. 365; Spirling & Rodriguez, 2020, p. 4).

CBOW and Skip-gram. The CBOW and Skip-gram architectures train a model to predict a word given a symmetric window of its surrounding words (CBOW) or to predict a set of words surrounding the input word (Skip-gram) (Mikolov et al., 2013a, p. 4-5). Given training data consisting of a sequence of T words $(a_1, \ldots, a_t, \ldots, a_T)$, the Skip-gram model seeks to minimize the following negative log-likelihood (Mikolov et al., 2013b, p. 3112):

$$\mathcal{L}_{Skip-gram} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-C\leq j\leq C, j\neq 0} log\, p(a_{t+j}|a_t) \tag{1.70}$$

where C is the number of context words to the left and right of input word a_t for which predictions are to be made. $p(a_{t+j}|a_t)$ is the probability of context word a_{t+j} to occur given input word a_t and is modeled as[a]

$$p(a_{t+j}|a_t) = \frac{exp(\tilde{\boldsymbol{z}}_{[a_{t+j}]}^{\top}\boldsymbol{z}_{[a_t]})}{\sum_{u=1}^{U} exp(\tilde{\boldsymbol{z}}_u^{\top}\boldsymbol{z}_{[a_t]})} \tag{1.71}$$

$\boldsymbol{z}_{[a_t]}$ is the embedding for the term corresponding to word a_t, $\tilde{\boldsymbol{z}}_{[a_{t+j}]}$ is the embedding corresponding to context word a_{t+j}, U is the size of the vocabulary, and $\tilde{\boldsymbol{z}}_u$ is the embedding of context term z_u (Mikolov et al., 2013b, p. 3113).

GloVe. GloVe minimizes the objective

$$\mathcal{L}_{GloVe} = \sum_{u=1}^{U}\sum_{j=1}^{U} f(c_{uj})(\boldsymbol{z}_u^{\top}\tilde{\boldsymbol{z}}_j + b_u + \tilde{b}_j - log\, c_{uj})^2 \tag{1.72}$$

where again U is the number of unique terms in the vocabulary, $\boldsymbol{z}_u$ is the embedding of term z_u, $\tilde{\boldsymbol{z}}_j$ is the embedding of context term z_j, b_u and $\tilde{b}_j$ are bias terms, and c_{uj} is the number of times that term z_j is present in the context of term z_u (Pennington et al., 2014, p. 1535).

[a] Due to the high costs involved in computing the softmax function in Equation 1.71, Mikolov et al. (2013b, p. 3113-3114) in practice approximate the softmax via negative sampling.

GloVe. (cont.) The size of the context window for which term co-occurrences are recorded may, for example, comprise 10 words to the left and right of each word (Pennington et al., 2014, p. 1538). Context windows can be chosen to be symmetric or asymmetric (Pennington et al., 2014, p. 1538). When computing the co-occurrence count c_{uj}, Pennington et al. (2014, p. 1538) furthermore use decreasing weights such that a context term z_j that is q positions away from a target term z_u, increases c_{uj} by $1/q$. $f(c_{uj})$ is a weighting function that seeks to ensure that term pairs that co-occur rarely and term pairs that co-occur frequently do not get overweighted (Pennington et al., 2014, p. 1535).

Therefore, GloVe effectively is a weighted least squares regression model (Pennington et al., 2014, p. 1535). GloVe seeks to find real-valued vector representations for term z_u and context term z_j such that the squared difference between the dot product of these representations, $\boldsymbol{z}_u^\top \tilde{\boldsymbol{z}}_j$, and the log of the number of times z_u and z_j co-occur in a context, $log\, c_{uj}$, is minimized (Pennington et al., 2014, p. 1535; Ruder, 2019a, p. 73).

Note that for each term z_u in the vocabulary, GloVe learns two separate embeddings: an embedding when z_u is the target term, $\boldsymbol{z}_u$, and an embedding when z_u is a context term, $\tilde{\boldsymbol{z}}_u$. If the context window is chosen to be symmetric, $\boldsymbol{z}_u$ and $\tilde{\boldsymbol{z}}_u$ should only differ due to slightly different random initializations (Pennington et al., 2014, p. 1538). To reduce variance and increase generalization performance, Pennington et al. (2014, p. 1538-1539) compute the final GloVe embedding vector for a term z_u as $\boldsymbol{z}_u + \tilde{\boldsymbol{z}}_u$.

The central shortcoming of these early procedures for learning embeddings is that for each feature a single embedding vector, $\boldsymbol{z}_u \in \mathbb{R}^K$, is learned. There are two consequences that result from representing each feature by a single vector. First, it implies that the features are mapped into one representational space that encodes one information about the features (Ruder, 2019a, p. 74). The representational space may, for example, primarily encode the syntactic relatedness between features but not very well capture their semantic similarity (Mikolov et al., 2013a, p. 7). Or, the encoded information may be a combination of semantic and syntactic aspects that, however, are fused into one representational space (Mikolov et al., 2013a, p. 7). In recent years, deep learning models have been utilized to learn *deep representations* in which each entity is represented by one vector at each hidden layer (Peters et al., 2018a, p. 2228-2230; Ruder, 2019a, p. 74). For example, when pretraining a deep biLSTM with two hidden layers on a language modeling task, one obtains for the tth token a_t one vector representation from each hidden layer such that a_t is not only represented by the input embedding $\boldsymbol{z}_{[a_t]}$ but also by $\boldsymbol{h}_t^1$ and $\boldsymbol{h}_t^2$ (see again Equations 1.58 and 1.60) (Peters et al., 2018a, p. 2229). By using deep neural networks, a hierarchy of feature representations is learned in which higher-level representations draw from lower-level representations (Ruder, 2019a, p. 74). This allows for learning different types of

information at different levels and for learning complex, abstract, task-specific higher-level representations that can make use of information encoded at lower levels (Ruder, 2019a, p. 74; Tenney et al., 2019).

Which specific type of information is encoded at which layer will depend on the model architecture and the task a model is trained on. Research that inspects the information captured by deep neural networks that have been trained on (variants of) language modeling tasks suggests that core information on textual building blocks (e.g. morphological information, or information about the presence or absence of words) tends to be encoded at lower layers, syntactic information at middle layers, and semantic information at higher layers (Peters et al., 2018a,b; Jawahar et al., 2019; Tenney et al., 2019). Moreover, deeper layers seem crucial for learning long-range syntactic or semantic dependencies (Peters et al., 2018b; Jawahar et al., 2019; Tenney et al., 2019). Empirical evidence indicates that many NLP tasks benefit from models that operate on deep hierarchical representations rather than on shallow one-layer representations (Peters et al., 2018a, p. 2232-2234).

The second consequence of representing each feature by a single vector is that different meanings of the same feature are subsumed into one representation (Pilehvar & Camacho-Collados, 2020, p. 60). This phenomenon is called *meaning conflation deficiency* (Pilehvar & Camacho-Collados, 2020, p. 60). To illustrate, according to the WordNet database (Miller, 1995), the term *'party'* denotes several different concepts (synsets). Two of these synsets are (1) "an organization to gain political power" (Princeton University, 2010) and (2) "an occasion on which people can assemble for social interaction and entertainment" (Princeton University, 2010). When representing the term *'party'* with a single vector, the vector representation will pool these two meanings. Hence, in a single continuous semantic representational space, the vector for *'party'* will be placed somewhere between vectors of terms that are semantically close to the first synset (e.g. *'politics'*, *'policy'*) and vectors of terms that are semantically close to the second synset (e.g. *'festivity'*, *'celebration'*) (see Figure 1.11) (Schütze, 1998, p. 102).[60]

The meaning conflation deficiency results from the fact that word embedding models learn one representation for each feature (Neelakantan et al., 2014, p. 1059). This is, for each class of tokens, that are composed of an identical sequence of characters and symbols, one representation is learned. This procedure, however, ignores that terms that have the same spelling can have multiple meanings (Neelakantan et al., 2014, p. 1059). There are homonyms: terms that have the same spelling but are conceptually distinct and hence have a different meaning (e.g. *'suit'*: lawsuit vs. garment) (Manning & Schütze, 1999, p. 110). And there are polysemes: terms that have the same spelling and different but

[60]Moreover, the meaning conflation deficiency can have negative consequences for the entire semantic space: Semantically unrelated terms that are semantically similar to different senses of a third polysemous term are likely to be drawn together in space (Neelakantan et al., 2014, p. 1059). For example, due to their similarity to the polysemous term *'mouse'* the terms *'rat'* and *'computer'* are drawn closer in semantic space as would be appropriate given their level of semantic dissimilarity (Neelakantan et al., 2014, p. 1059-1060; Pilehvar & Camacho-Collados, 2020, p. 61).

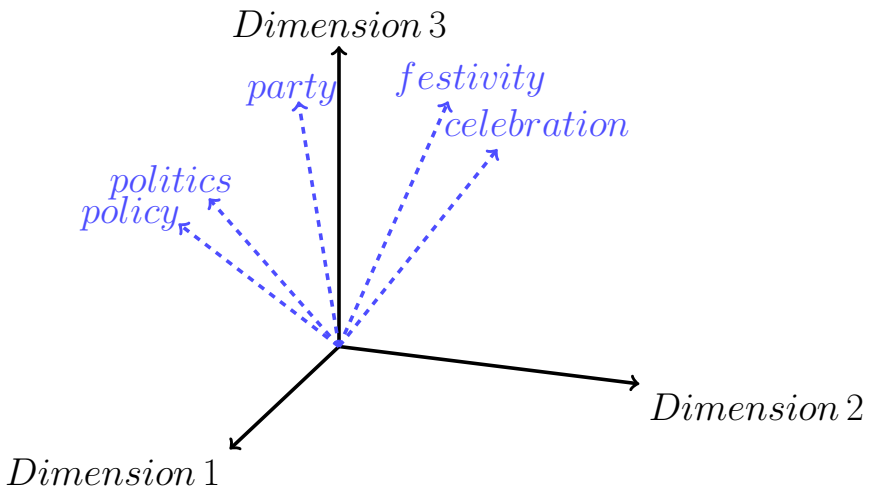


Figure 1.11: **Meaning Conflation Deficiency.** The term *'party'* is a homonym. It has several distinct meanings. Two of them are (1) "an organization to gain political power" (Princeton University, 2010) and (2) "an occasion on which people can assemble for social interaction and entertainment" (Princeton University, 2010). When the term *'party'* is represented by a single vector, the vector fuses these distinct meanings and is positioned somewhere between vector representations of terms that are semantically close to the first concept (e.g. *'politics'*, *'policy'*) and vector representations of terms that are semantically close to the second synset (e.g. *'festivity'*, *'celebration'*).

related meanings (e.g. *'branch'*: the branch of a plant vs. the branch of an organization) (Manning & Schütze, 1999, p. 110).

One approach to address the meaning conflation deficiency is to learn representations not for individual terms but for individual meanings (i.e. senses) of terms (Camacho-Collados & Pilehvar, 2018, p. 748). The idea of *sense representations* is to have one representation per sense rather than one representation per term (Camacho-Collados & Pilehvar, 2018, p. 748). Methods for learning sense representations can be roughly grouped into (1) unsupervised approaches in which the differentiation between distinct senses of a term, as well as representations for the different senses, are learned in an unsupervised fashion directly from text corpora (e.g. Schütze, 1998) and (2) knowledge-based approaches that use sense inventories such as WordNet to learn sense representations (e.g. Rothe & Schütze, 2015) (Camacho-Collados & Pilehvar, 2018, p. 744, 749).[61]

After having learned sense representations, a word sense disambiguation algorithm can be applied to a sequence of tokens that is to be processed by an NLP model (Pilehvar & Camacho-Collados, 2020, p. 76). The word sense disambiguation algorithm can identify the senses of ambiguous tokens in the sequence such that then the respective sense embeddings (rather than the word embeddings) are used to represent the tokens (Pilehvar & Camacho-Collados, 2020, p. 76). Although such a procedure helps in incorporating information about senses of terms, word sense disambiguation is not without error and introduces additional

[61] For a detailed overview of sense representation learning techniques see Camacho-Collados & Pilehvar (2018).

noise (Pilehvar & Camacho-Collados, 2020, p. 76).

At present, the predominantly used approach to overcome the meaning conflation deficiency and to acquire representations that capture contextualized meanings is to learn *contextualized embeddings* (Pilehvar & Camacho-Collados, 2020, p. 74). Word embeddings are static in the sense that regardless of a feature's context—and hence regardless of the context-specific meaning of a feature—there is one representation for each feature (Pilehvar & Camacho-Collados, 2020, p. 74). Contextualized embeddings, in contrast, are adaptive: There is one representation per token rather than one representation per feature (Pilehvar & Camacho-Collados, 2020, p. 82). The embedding for a token a_t is a function of the embeddings of those tokens that are positioned before and/or after token a_t (Pilehvar & Camacho-Collados, 2020, p. 82). Hence, the embedding for token a_t is a function of a_t's context, which is different for each token in each sequence (Pilehvar & Camacho-Collados, 2020, p. 82).

How are these context-specific token representations learned? Contextualized embeddings simply are the hidden representations in deep neural networks (Pilehvar & Camacho-Collados, 2020, p. 85). In an RNN, for example, the hidden representation for the tth token, $\boldsymbol{h}_t$, is obtained by sequentially processing all tokens up until the tth token (see again Section 1.2.3.6 and Equation 1.58). $\boldsymbol{h}_t$ hence is not only a function of the static input word embedding $\boldsymbol{z}_{[a_t]}$, it is also a function of the sequence of its preceding tokens and thus provides a contextualized representation of the tth token. Similarly, the representation for the tth token produced as the output from a Transformer encoder (which is an updated version of $\boldsymbol{c}_t$ in Equation 1.63) results from the self-attention mechanism that is specifically designed to let the representation for the tth token incorporate information from surrounding tokens (see again Section 1.2.3.7 and Equation 1.63). Consequently, if two tokens consist of the same character string but are surrounded by a different set of tokens, their hidden representations will be different. The different token representations may reflect the tokens' (slightly) different meanings that are invoked by the contexts in which they are embedded in (Reif et al., 2019). The different representations, however, also may reflect the tokens' different syntactic properties or their semantic or syntactic relations to other tokens in the sequence (Clark et al., 2019; Hewitt & Manning, 2019; Tenney et al., 2019).

The utilization of representations that are *deep* and *contextualized* (e.g. in ELMo by Peters et al., 2018a) introduced major advancements within the field of NLP. Moreover, combining research on sequential transfer learning, on the one hand, with deep contextualized representations, on the other hand, it became a common practice within NLP to use deep neural networks in pretraining in order to learn model parameters that are functional in producing contextualized representations and then to take the entire pretrained model (including the pretrained model parameters) for fine-tuning on the target task (Pilehvar & Camacho-Collados, 2020, p. 85; Ruder, 2021b).[62] The idea in sequential transfer learning

[62]Because in deep neural networks, that are used for generating deep contextualized token representations, the representation of token a_t will be different each time a_t occurs with another sequence of tokens,

is to pretrain, transfer, and make use of model architectures with model parameters that are effective in creating representations that are applicable across a wide spectrum of tasks. Deep contextualized token representations are context-specific, but their representational form—and the model architecture with pretrained model parameters that are involved in creating them—can be a highly general, well-generalizing tool.

This dissertation presents and applies several deep learning architectures that are used in a transfer learning setting and produce deep contextualized representations (see this dissertation's article *Introduction to neural transfer learning with Transformers for social science text analysis*).

1.2.4 Supervised Learning in Text-Based Political Science Research

Section 1.2 has mapped the fundamentals of modern NLP techniques. Against this background, there is the question of which methods political scientists use when measuring concepts from texts. How do political scientists represent textual entities (words, documents, ...), and which supervised learning algorithms do they apply? In order to answer these questions based on a systematic review (instead of relying on single examples) the following procedure was conducted:

1. In Political Analysis—which arguably still is the leading method-focused political science journal—all articles that were published in issues 01/2020 to 02/2022 as well as all articles that were listed on April 6th, 2022 as recently published online *FirstView* articles were collected. Among these 113 collected articles, those articles that contained *any* of the following keywords within their full text were retained for further inspection: *"machine learning", "text analysis", "text data", "natural language processing", "deep learning", "neural network".*

2. For each of the 36 articles that resulted from the search in step 1, it was inspected whether the article applied supervised machine learning on text data. For each of the 14 articles that did, it was recorded separately for each presented application (i.e. for each target task)

    - which type of text was used,
    - which concept was measured,
    - how documents were represented, and
    - which learning algorithms were used for training on the target task.

a token representation is not a static representation that can be learned once in pretraining and then can be transferred. Rather, a token representation is dynamically created and is specific to a token sequence. Hence, in transfer learning, it is not that token representations themselves are transferred but rather the model architecture with pretrained model parameters (weight matrices, bias terms), that are functional in producing token representations, are transferred.

The results of this literature search are presented in Table 1.3.[63] In all applications of supervised learning on text data that have been identified with this literature search procedure, documents are represented as feature vectors in a document-feature matrix. The document-feature matrix hereby often result from a very basic preprocessing process (see e.g. Di Cocco & Monechi, 2021, p. 5; Osnabrügge et al., 2021, p. 6). Most articles combine document-feature matrix representations with conventional machine learning algorithms. SVMs, generalized linear models (GLMs) (mostly: logistic regression), and random forests are particularly frequently applied.[64] In some applications, researchers also experiment with other representational forms. In these instances, a document is represented as the (weighted) mean across pretrained word embeddings, as a vector representation obtained via the Paragraph Vector method (Le & Mikolov, 2014), or as a lower-dimensional representation obtained from a topic model.

Neural networks are applied less frequently than conventional machine learning algorithms. And if deep neural networks are used as learning algorithms, then they are not always used in ways that are likely to yield desired performance effects: Di Cocco & Monechi (2021) and Erlich et al. (2021), for example, employ very small neural networks (e.g. an FNN with two hidden layers (Di Cocco & Monechi, 2021, Appendix p. 6), or a CNN with one convolution layer and embedding dimension 32 (Erlich et al., 2021, Appendix p. 15)). Moreover, no pretraining is employed in these cases. Only Chang & Masterson (2020) present an adequate application of LSTMs—and then empirically also observe performance enhancements. There is no application that employs Transformer-based models for supervised learning.[65]

Hence, in the major method-focused journal for political science research, scientists primarily apply methods for supervised learning on text data that stem from the past era of statistical NLP and not the era of neural NLP (see again Section 1.2.2). The literature search conducted here does not uncover a single application of Transformer-based models (let alone Transformers plus sequential transfer learning with fine-tuning) for the purpose of measuring concepts from text. The article *Introduction to neural transfer learning with Transformers for social science text analysis* seeks to close this gap by providing an introduction to deep learning, sequential transfer learning, and the Transformer architecture.

[63] Note that the literature search procedure only focuses on applications of supervised learning on text data. Table 1.3 hence does not list articles that exclusively apply self-supervised learning or unsupervised learning on text data (Bussell, 2020; Rheault & Cochrane, 2020; Enamorado et al., 2021; Ying et al., 2021, e.g.). Table 1.3 also does not present information on articles that apply supervised learning on data other than text (e.g. Torres & Cantu, 2022), or articles that use text data but do not apply supervised learning (e.g. Kim & Kunisky, 2021). Two notable articles within this group of not listed articles are Torres & Cantu (2022) and Porter & Velez (2021). Torres & Cantu (2022) introduce the processing of image data with CNNs to a political science audience. Porter & Velez (2021) use the Transformer-based GPT-2 model for natural language generation to construct placebo texts for survey experiments.

[64] A random forest is a tree-based bagging algorithm (see Breiman, 2001a).

[65] Note, however, that Porter & Velez (2021) utilize the natural language generation-abilities of the Transformer-based GPT-2 to generate texts that are then used as placebos in survey experiments.

			Representation					Learning Algorithm Architecture									
Paper	**Document Type**	**Concept**	document-feature matrix	mean of word embeddings	Paragraph Vector embeddings[a]	topic model representations	sequence of word embeddings	SVM	(regularized) GLM	naive Bayes (or related)	random forest (or related)	tree-based boosting	ReadMe / ReadMe2[b]	FNN (or related)	CNN	LSTM	Transformer-based
Bustikova et al. (2020)[c]	parties' opinion articles	partisan responsiveness	X		X	X			X	X	X						
Rodman (2020)	newspaper articles	equality topics	X										X				
Chang & Masterson (2020)[d]	Weibo posts	(a)political content	X	X			X	X		X	X	X				X	
Chang & Masterson (2020)	newspaper articles	rape culture	X				X	X								X	
Huang et al. (2020b)[e]	parliamentary speeches	speech distinctiveness	X							X							
Miller et al. (2020)	tweets	refugee topic	X					X									
Miller et al. (2020)	Wikipedia talk pages	toxicity	X					X									
Miller et al. (2020)[f]	news articles	Muslim identity	X					X	X	X				X			
Mozer et al. (2020)	news articles	document similarity	X	X		X			X								
Barberá et al. (2021)	newspaper articles	tone	X					X	X		X	X					
Sebők & Kacsuk (2021)	newspaper articles	policy topics	X					X		X	X						
Fong & Tyler (2021)	Reddit comments	uncivility	X					X									
Di Cocco & Monechi (2021)	election manifestos	populism	X						X		X	X		X			
Erlich et al. (2021)	human rights reports	human rights violations	X					X	X		X						
							X								X	X	
Erlich et al. (2021)	information requests	request attributes	X						X		X	X					
Jerzak et al. (2022)[g]	73 different corpora	concepts in 73 tasks	X	X				X	X	X	X		X				
Kaufman & Klevs (2021)	entity names	matching string	X								X						
Osnabrügge et al. (2021)	parliamentary speeches	speech topics	X						X			X					

Table 1.3: **Supervised Learning on Text Data in Political Analysis.** This table lists all applications of supervised learning on text data in articles that have been published in Political Analysis between issue 01/2020 and issue 02/2022 (including *FirstView* articles) as identified by the research procedure described on page 125. A row in the table reports the information on one application (i.e. one target task). For each application, the table notes the type of document analyzed, the concept that researchers seek to measure, the representational form that was used to represent documents, and the learning algorithm employed. The table lists even those representations and learning algorithms that are only mentioned in the articles' appendices. If an article uses one representational form as an input for one learning algorithm but not for other algorithms, then a separate row for each combination of representation and algorithm is printed here. The footnotes a to g are given on the next page.

1.3 Measuring Attitudes from Texts

So far, core concepts and methods in machine learning and NLP have been introduced. The article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* has been motivated on pages 65 to 67 whilst discussing selection processes. The article *Introduction to neural transfer learning with Transformers for social science text analysis* has been motivated in Section 1.2.4 by showing that text-based supervised analyses in political science do not widely use current NLP methods. This section now motivates the article *How to estimate continuous sentiments from texts using binary training data* by more closely inspecting a specific concept and the ways in which this concept is usually measured from text.

1.3.1 Defining Attitudes

In the computer science and NLP literature, sentiment is a loosely defined concept. Concepts that relate to affect (e.g. emotion/feelings) as well as concepts that relate to cognition (e.g. opinion) are seen as components of sentiment or are handled as if they were the same as sentiment (Pang & Lee, 2008, p. 5-6; Liu, 2015, p. 1-2, 20-21; Mohammad et al., 2017, p. 2). When viewing these vague definitions together, however, they correspond very well with the concept of an *attitude* as used in social psychology (Liu, 2015, p. 2). Hence, here sentiments are taken to be attitudes.

Across all attitude definitions that have been brought up in social psychology, the consistent core notion is that an attitude is an evaluation of an entity (Banaji & Heiphetz, 2010, p. 352; Albarracin et al., 2019, p. 3). A widely used definition reflecting this core notion is given by Eagly & Chaiken (1993, p. 1) in which an attitude is conceptualized as "a psychological tendency that is expressed by evaluating a particular entity with some degree of favour or disfavour".

According to this definition, an attitude is an inner latent tendency that can find expression

[a]Paragraph Vector embeddings have been introduced by Le & Mikolov (2014).

[b]ReadMe refers to the supervised learning procedure described in Hopkins & King (2010). ReadMe2 is presented in Jerzak et al. (2022).

[c]Bustikova et al. (2020, p. 50) use "a gradient descent approach known as SLEP". In fact, this approach is regularized logistic regression optimized via a gradient descent algorithm (Liu et al., 2009).

[d]Chang & Masterson (2020) apply the extra-trees classifier that is related to the random forest classifier (see Chang & Masterson, 2020, p. 402).

[e]Huang et al. (2020b) apply a specifically developed learning method that is related to multinomial naive Bayes (see Huang et al., 2020b, p. 420).

[f]Miller et al. (2020) apply the perceptron classifier, which is a linear algorithm for binary classification (Bishop, 2006, p. 192-193). In contrast to a multilayer perceptron (also known as FNN), the perceptron classifier has a single layer and no hidden layers.

[g]Jerzak et al. (2022) apply eight further classifiers for quantification that are not listed here but are presented in Firat (2016).

in an evaluative response (Eagly & Chaiken, 2007, p. 585-587). It is important to differentiate between the attitude itself and an evaluative response toward an entity (Eagly & Chaiken, 2007, p. 586-587; Fabringar et al., 2019, p. 111). Attitudes can give rise to evaluative responses, but there are additional influencing situational cues and contextual factors that have an effect on what evaluative judgment is expressed (Eagly & Chaiken, 2007, p. 587). Researchers can only observe manifested responses but not the latent attitudes themselves (Eagly & Chaiken, 2007, p. 584-585).

Attitudes, furthermore, are considered to be learned structures that are stored in memory and can be accessed from there (Fabringar et al., 2019, p. 110-111).[66] An attitude emanates from various attitude-relevant information bases that can be of an affective, cognitive, and behavioral nature (Fabringar et al., 2019, p. 112). When expressed in mathematical terms, an attitude is a continuous variable resulting from a function that maps from various input variables (the information bases) to a single continuous output (the attitude value) (see Figure 1.12) (Cacioppo et al., 1997, p. 10, 13). The input variables (the information bases) can be, for example, affective feelings toward the entity, beliefs about the entity, or past behavioral responses toward the entity (Albarracin et al., 2019, p. 9; Fabringar et al., 2019, p. 112). Here, ambivalences may arise: A person may have positive and negative feelings toward an entity or may associate a positive emotion with the entity but at the same time may have unfavorable thoughts toward the entity (Fabringar et al., 2019, p. 115-116). Attitudes are the aggregated summary evaluation of these various attitude-relevant information bases.

Attitudes are of high importance for social scientists because attitudes are among the most central factors that influence what behavioral intentions humans form and which actions they undertake (Ajzen, 1991). Attitudes, for example, define the values individuals attach to outcomes. By comparing the attitude toward an outcome with the attitude toward another outcome and the attitude toward yet another outcome, ranked preferences over a set of outcomes emerge (Druckman & Lupia, 2000, p. 4). From economics to social psychology, several theories combine the subjective value an individual attaches to an outcome with an individual's beliefs about self-efficacy and action-outcome relationships to explain individual behavior (see e.g. the expected utility hypothesis in rational choice theory, or the theory of planned behavior) (Ajzen, 1991; Shepsle & Bonchek, 2010).

[66] Note that the here outlined conceptualization of attitudes follows the prevailing view among social psychologists studying attitudes (Fabringar et al., 2019, p. 110-111). In the opposing constructivist view, attitudes are not learned structures in memory but rather are regarded to be constructions that are made on-the-spot and arise from the situational context and readily accessible information (Banaji & Heiphetz, 2010, p. 352). The prevailing view, which is adopted in this dissertation, argues that evaluative reactions indeed are constructed in a given situation from various influencing factors (one of which is attitudes), but attitudes themselves are not on-the-spot creations but rather already existing, learned tendencies (Eagly & Chaiken, 2007, p. 585, 587). Attitudes here are seen as "mental residues of past experience with the attitude object" (Eagly & Chaiken, 2007, p. 587). As such repositories, attitudes are more or less enduring. They can change over time due to ongoing experiences with the attitude entity (Fabringar et al., 2019, p. 111).

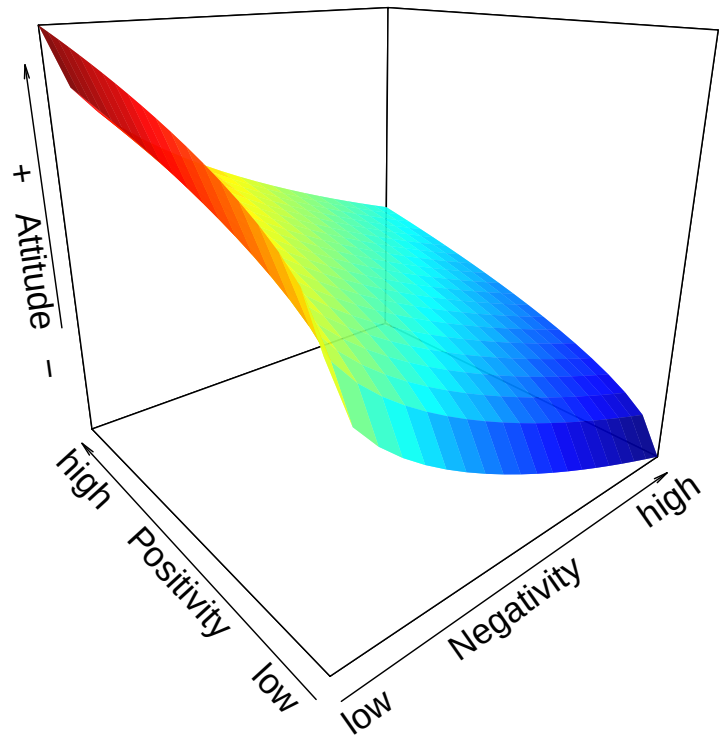


Figure 1.12: **Attitude as a Continuous Concept.** In social psychology, an attitude is conceptualized as a continuous variable whose value is a function of the activation of positive and negative information bases associated with the attitude entity (Cacioppo et al., 1997, p. 10, 13; Fabringar et al., 2019, p. 112). The level of activation of positive and negative information bases here runs from low to high. This figure is adapted from "Relationship between attitudes and evaluative space: A critical review, with emphasis on the separability of positive and negative substrates," by J. T. Cacioppo and G. G. Berntson, 1994, *Psychological Bulletin,115*(3), p. 412. Copyright 1994 by the American Psychological Association. Cacioppo & Berntson (1994, p. 412) note that the attitudes in the middle of the value range depicted here can result from a low activation of positive and negative information bases (neutrality) or from a high activation of positive and negative information bases (ambivalence).

Through the aggregation of individual actions, attitudes also affect the social level. The distribution of voters' attitudes toward political parties may materialize in election results. Moreover, policy-makers are likely to sense the direction in which a population's attitudes toward a policy issue are heading and, to improve their re-election chances, may adapt their political decisions accordingly (Stimson et al., 1995, p. 544-545). Hence, attitudes are central for individual action and aggregate social outcomes.

Traditionally, the measurement of attitudes is conducted via self-reports (e.g. in surveys or as part of experiments) or via implicit measures (e.g. in implicit association tests). In the latter, attitude values are deduced from the speed or accuracy with which humans conduct tasks when being exposed to different attitude objects and value-laden stimuli (Gawronski & Brannon, 2019, p. 158ff.). Especially if the texts are not produced for the purpose of being studied but rather in a real-life social context as a means of communication, the measurement of attitude expressions from texts—in contrast to self-reports—constitutes an unobtrusive method of observation. In contrast to implicit measures employed in lab experiments, the textual analysis of attitudes allows for studying attitude expressions directly in the contexts in which they emerge and the contexts they may have an effect upon (e.g. via processes of social influence). For social scientists, the identification of attitudes on the basis of textual data therefore is an interesting method for the measurement of attitudes, and it also is a significant one: As large amounts of communication, social interaction, and discourse take place in digital settings, text data that contain expressions of attitudes are abundant. Being able to identify expressed attitudes in data from these digital sources allows researchers to gain insights into these prominent contexts of human interaction.

1.3.2 Measuring Attitudes

The outlined conceptualization of attitudes has four consequences on the measurement of evaluative responses from texts. First, an attitude is devised as a latent concept (a tendency) that then can be expressed in some form. Texts thus do not contain attitudes themselves. Texts only can be expressions of attitudes—or rather evaluative responses affected by attitudes but also other factors. This implies that an evaluation expressed in a text need not necessarily correspond to the true overall evaluation captured by the underlying attitude. In a given situation and prompted with given stimuli and cues, only parts of the attitude that draws from multifaceted attitude-relevant information may be activated such that only parts of the attitude are revealed (Eagly & Chaiken, 2007, p. 595-598). There also may be a strategic mismatch between what is conveyed and the true position of the person writing the text. That a textually expressed attitude may not equal the true underlying attitude is one of several steps in the data generating process and the following data selection process that may induce uncertainty or bias when trying to infer attitudes from texts (Benoit et al., 2009, p. 497-498) (see also pages 57 to 59 in this thesis).

Second, in social psychology, an attitude is defined as an evaluation of a specific target entity (Eagly & Chaiken, 2007, p. 583). In the computer science and NLP literature, sentiment is also defined to refer to an entity (Liu, 2015, p. 17-18). Entities can be abstract (e.g. an idea expressed in an election manifesto) or tangible (e.g. a political candidate) (Eagly & Chaiken, 2007, p. 583). They can be highly specific (e.g. the remark expressed by candidate A on issue X in an interview given yesterday) or very general (e.g. the political system) (Albarracin et al., 2019, p. 5), but there is always a target entity (Eagly & Chaiken, 2007, p. 583-584). Attitude expressions are not targetless expressions of moods (Eagly & Chaiken, 2007, p. 583-584). Hence, textual analyses of sentiment expressions have to be oriented toward a specific target.

In practice, however, this is not always the case. Sentiment analysis sometimes simply means assessing the general tone of a document—which comes down to analyzing a document's polarity as revealed by the positivity vs. negativity of the used language (Nakov et al., 2016; Mohammad et al., 2017; Patwa et al., 2020). In cases in which the analysis of expressed sentiments implies assessing the expressed sentiments toward given targets, the NLP literature differentiates between *target-based sentiment analysis* and *stance detection* (Küçük & Can, 2020, p. 6). In target-based sentiment analysis, the aim is to estimate the sentiment toward a prespecified target, that is typically explicitly mentioned in the text, based on information revealed in the text (Küçük & Can, 2020, p. 6; ALDayel & Magdy, 2021, p. 5). In stance detection, in contrast, the target entity is not necessarily referred to explicitly, and determining the position toward a target may involve moving beyond the information provided in the text (Mohammad et al., 2017, p. 2; Küçük & Can, 2020, p. 6; ALDayel & Magdy, 2021, p. 5). Table 1.4 illustrates the different conclusions that untargeted sentiment analysis, target-based sentiment analysis, and stance detection might draw from identical texts. Table 1.4 points out that attitude expressing statements are embedded in larger knowledge structures, can be made within conversational contexts, and can be implied rather than being stated explicitly (e.g. by making comparisons). For example, understanding that the second statement expresses a negative attitude toward Donald Trump requires the knowledge that the statement was made within the context of the 2016 presidential election in which Jeb Bush was one of the candidates competing in the Republican presidential primaries alongside Donald Trump. This highly varied and strongly contextual nature of linguistic expressions of sentiment is accounted for in stance detection but not in the other presented types of sentiment analysis tasks.

The third aspect in the outlined conceptualization of attitudes, that has an effect on the textual measurement of attitude expressions, is that an attitude is linked to and is derived from a multi-dimensional structure of attitude-relevant information. Studies that examine these structures make use of the fact that humans can report a single overall evaluative attitude toward an entity but also can report the underlying attitude-relevant information (see e.g. Wood, 1982). In textual evaluations of entities, humans—even if not explicitly asked to do so—occasionally present underlying information that they feel causes them to hold the attitude. In product reviews, for example, attributes of the product are assessed and different experiences are reported that then form an overall evaluation. In political

Text	Target	Untargeted Sentiment	Target-Based Sentiment	Stance
I cannot believe there are still people in this century who are opposed to women having rights over their own bodies! #disgusted #repealthe8th	Legalization of Abortion	−	+	+
Jeb Bush is the only sane candidate in this republican lineup.	Donald Trump	0	*N/A*	−

Table 1.4: **Sentiment and Stance.** This table illustrates the differences between untargeted sentiment analysis, target-based sentiment analysis, and stance detection. The first text is a tweet adapted from the Stance Dataset provided by Mohammad et al. (2017). The second text is directly taken from Mohammad et al. (2017, p. 2). The first statement expresses a favorable position toward the legalization of abortion and also explicitly mentions this target in the form of the hashtag #repealthe8th. Yet it uses negative language which is why untargeted sentiment analysis would predict a negative polarity. The language used in the second statement is neither particularly positive nor negative. If Donald Trump were the prespecified target, a target-based sentiment analysis would conclude that the statement does not express a sentiment toward the target, whereas in stance detection the prediction should be that the statement reveals a negative stance toward the target.

speeches, an overall attitude toward a political issue can be expressed together with beliefs and attitude-relevant experiences from which the policy position is derived from.

The subfield of sentiment analysis that seeks to extract evaluative attitude-relevant information from texts is aspect-based sentiment analysis. Whereas target-based sentiment analysis focuses on the overall sentiment expressed toward a monolithic entity, aspect-based sentiment analysis seeks to identify attitudes toward *aspects* of an entity from text (Liu, 2015, p. 22). Aspects can be parts or attributes of an entity (Liu, 2015, p. 22). In addition to the subtasks that have to be addressed in sentiment analysis, aspect-based sentiment analysis also has to identify textual expressions that mention an aspect, has to group aspect expressions referring to the same aspect together, resolve which aspect belongs to which entity, and determine the respective sentiments (Liu, 2015, p. 26-28; Pontiki et al., 2014, 2015). By analyzing sentiments not only towards an entity as a whole but also towards the aspects of an entity, aspect-based sentiment analysis draws a more nuanced picture. It can reveal the positive and negative evaluative processes relating to parts of an entity that contribute to the overall evaluative assessment of that entity.[67]

Another NLP task to be mentioned here is *argument mining*. In argument mining, the aim is not only to identify the position[68] that a person expresses in a text but also to extract the provided line of reasoning (Lawrence & Reed, 2020, p. 765-766). Argument mining involves the identification of the textual components of an argument, the identification of each component's properties and function, and the mapping of relations between the components (Lawrence & Reed, 2020, p. 766, 787, 791).[69]

Target-based sentiment analysis, stance detection, aspect-based sentiment analysis, and argument mining are approaches that do much more justice to the conceptualization of attitudes and their underpinnings than the untargeted analyses of sentiments. Especially stance detection is of high importance because it aims at a comprehensive extraction of all sentiments toward a prespecified target—no matter whether the sentiments are expressed in explicit or implicit form. A comprehensive identification of all sentiments expressed toward an entity, in turn, is a prerequisite for making valid inferences, e.g. about how expressed sentiments toward an entity are distributed in a given population.

There is, however, a fourth aspect that results from the conceptualization of attitudes that

[67] Aspect-based sentiment analysis tasks every now and then are included as evaluation tasks in the International Workshop on Semantic Evaluation (Pontiki et al., 2014, 2015, 2016).

[68] A position expressed in a text could be an attitude toward an entity (which is why argument mining is mentioned here). A position could, however, also be a belief. A belief "is a mental state that has as its proposition *that X*" (Leitgeb, 2017, p. 2; emphasis in the original). This is, a belief is an individual's internal representation of the world (Leitgeb, 2017, p. 3). An individual, for example, can hold the belief *that Angela Merkel is a pragmatic woman* or *that CO2-emissions cause global warming*. Beliefs do not have to be correct, yet beliefs contain what individuals assume the world to be, i.e. what individuals hold to be true (Leitgeb, 2017, p. 3).

[69] An early important work is Moens et al. (2007). A recent survey of argument mining is given by Lawrence & Reed (2020). Current research, for example, is regularly presented in the Workshop on Argument Mining (Stein & Wachsmuth, 2019; Cabrio & Villata, 2020).

these advanced approaches do not address and that so far is not widely considered in the NLP literature: In their definition of attitudes Eagly & Chaiken (1993, p. 1) refer to the notion of *degree*. In mathematical treatments of attitudes, attitudes are conceptualized to have a continuous character (see again Figure 1.12) (Cacioppo et al., 1997). They are assumed to be continuous variables. In the NLP literature, correspondingly, a sentiment is seen as having a certain level of intensity (Liu, 2015, p. 20-21).

Despite this conceptualization of sentiment as a continuous quantity, sentiment analysis is treated as a classification task in a vast majority of studies. In binary sentiment classification tasks, the aim is to predict whether a document falls into the positive vs. the negative sentiment category (e.g. Pang et al., 2002; Turney, 2002). Often, a third neutral category is added, thereby turning sentiment analysis into a multi-class classification task (e.g. Nakov et al., 2013; Pontiki et al., 2015; Zhang et al., 2015a; Patwa et al., 2020). At times, sentiment analysis also is treated as an ordinal classification problem, where typically three or five discrete and ordered sentiment categories are distinguished (e.g. Pang & Lee, 2005; Socher et al., 2013; Cheang et al., 2020)

Predicting discrete sentiment labels rather than generating continuous sentiment estimates not only disregards the conceptualization of sentiments but also implies a loss of information. In the extreme, researchers that only have discrete sentiment categorizations cannot differentiate between several possible but substantively entirely different underlying continuous sentiment distributions (e.g. a unimodal vs. a bimodal distribution of sentiments).

Nevertheless, there are methods that produce continuous sentiment estimates for texts, namely lexicon-based approaches and regression approaches. Lexicon-based approaches, however, rely on access to an extensive sentiment lexicon that matches the specific context it is applied to and also rely on an accurate modeling of the compositional process via which sentiment is constructed in texts. Regression approaches require (possibly prohibitively) costly to create training labels that are so fine-grained that they can be treated as if they were continuous. Hence, there is a need for a method that generates continuous sentiment estimates with less information or resources. The article *How to estimate continuous sentiments from texts using binary training data* of this doctoral thesis addresses this aspect and seeks to close this gap.

The article presents the *c*lassifier-based *b*eta *m*ixed *m*odeling procedure (CBMM for short) that I have developed together with Christian Heumann. CBMM estimates continuous, real-valued positions of text documents on a unidimensional latent variable from mere binary training labels and consists of three steps: First, binary training labels are obtained for the training set documents. Each binary training label should indicate whether the document's position on the latent sentiment variable is closer to the positive or the negative extreme. Second, a set of classifiers is trained on the binary training data, and then each classifier produces for each test set document a predicted probability for the document to belong to the positive class. Third, given the predicted probabilities, a beta mixed model with document and classifier random intercepts is estimated. The random intercepts for

Wordscores. Political scientists also have developed supervised methods that generate continuous estimates for text documents. The best known method is Wordscores (Laver et al., 2003). In Wordscores, the training set documents (called reference texts) have to be mapped to continuous values that specify their position on the latent variable (Laver et al., 2003, p. 315). Based on these values assigned to the reference texts and the frequency of word type occurrences across the reference texts, for each word type that occurs at least once in the reference texts, a *wordscore* is computed (Laver et al., 2003, p. 316, 319). Subsequently, these wordscores are used to estimate the positions of the test documents (called virgin texts) (Laver et al., 2003, p. 316). The Wordscores approach (Laver et al., 2003) thus is only useful for applications in which continuous values can be easily assigned to training documents.

A related alternative method, the Class Affinity Model (Perry & Benoit, 2017), requires the reference texts to be archetype documents positioned at the extreme points of the latent variable of interest (Perry & Benoit, 2017, p. 3-4, 13). In this approach, the required information is not continuous but still highly detailed. The training data cannot be a random sample from the population of documents under study but are required to be "clearly polar examples of each reference class" (Perry & Benoit, 2017, p. 13). The training documents have to be selected in such a way that they not only conceptually but also linguistically represent the extremes of the latent variable (Perry & Benoit, 2017, p. 13). In applications with a large corpus, this is likely to be a difficult to unsolvable task. Wordscores and the Class Affinity Model thus share the disadvantages of regression-based approaches when it comes to producing continuous estimates for texts.

the documents serve as the estimates of the documents' continuous sentiment values. An evaluation across four data sets shows that the continuous sentiment estimates generated by CBMM are passably close to the true sentiment values and perform similarly to (or only slightly less well than) continuous sentiment estimates that are produced by regression approaches that operate on fine-grained training data. Hence, researchers that lack the resources to create highly fine-grained training labels but are able to produce a binary labeling of training documents, can apply CBMM and are likely to get continuous sentiment estimates that are not far from the estimates they would have obtained with fine-grained labels.

Whereas the other articles of this thesis seek to advance text-based political science research by increasing the accuracy with which supervised classification tasks are conducted, the article on CBMM aims at solving a problem by creating a new method. The major contribution of CBMM is its ability to generate continuous estimates based on binary training data. The article not only emphasizes that sentiments are continuous concepts but also provides a resource-efficient way to produce continuous estimates for textual sentiment expressions.

CBMM can also be used as a plug-in for classification components within more complex NLP model architectures developed for target-based sentiment analysis, aspect-based sentiment analysis, or stance detection: CBMM assumes that one text sequence expresses one evaluative statement towards one already known entity. Whenever more complex model architectures apply a classification module to a text sequence that already has been identified to express a single attitude toward an already identified (aspect of an) entity, CBMM can be used instead.[70] For example, Hu et al. (2019) conduct open-domain targeted sentiment analysis. In a first step, they identify text spans that refer to target entities (Hu et al., 2019, p. 539-540). In a second step, based on the representations of the identified spans, they use the attention mechanism and an FNN to classify the expressed sentiment (Hu et al., 2019, p. 539-540). Instead of classification, CBMM could be applied in the second step.

CBMM not only contributes to the field of NLP but also to political science: It is likely that there will be political science research projects that, on the one hand, seek to obtain estimates on a continuous latent variable from texts but, on the other hand, lack large amounts of annotation resources. Moreover, whereas the starting point for CBMM is sentiment analysis, CBMM can be applied in any study that aims to produce continuous estimates from binary training labels. Examples of application contexts beyond sentiment analysis are the measurement of actors' ideological positions on an a priori-defined ideological dimension or the text-based measurement of the degree to which people hold certain beliefs.

1.4 Limitations and Further Research

This section comprises two subsections.[71] Subsection 1.4.1 works out one central limitation of this thesis and NLP research in general: shortcomings regarding the drawing of inferences about the performance effects of methods. Subsequently, subsection 1.4.2 points out important directions for future text-based political science research.

1.4.1 Inference

At the beginning of this introductory chapter on page 39 the contribution of this dissertation was stated as follows: "Each of the three articles presents a set of methods that are likely to provide a concrete improvement over specific current text analytic practices in political science." This is what this dissertation does. This dissertation, however, does not

[70] CBMM can be used in these cases—provided that resources for training an ensemble of classifiers are available.

[71] Large parts of this section are published in Wankmüller, Sandra. 2022. Drawing Causal Inferences About Performance Effects in NLP. arXiv. https://arxiv.org/abs/2209.06790

(and arguably cannot) fully reach a larger underlying goal that goes beyond the *presentation* of methods that "are likely to" constitute an improvement. This larger underlying goal is inference.

1.4.1.1 Drawing Inferences in NLP

The actual goal in NLP is to infer how well one method (compared to another method) performs in solving a certain NLP task.[72] Yet, due to the usual research procedures in NLP, this goal is often not achieved. The common approach in NLP is as follows (Reimers & Gurevych, 2018, p. 1-3):

1. The available annotated data are separated into one training, one validation, and one test set.
2. For each method that is to be compared, a small set of models are trained on the training set and are evaluated on the validation set.
3. For each method, the model that performs best on the validation set subsequently is evaluated on the test set and the model's performance on the test set is reported.

The described NLP research procedure has two problems: First, the fact that training is typically conducted on one specific training data set and evaluation is typically conducted on one specific test data set implies that the reported performance values are estimates of the test error (see Equation 1.18) and not estimates of the expected generalization error (see Equations 1.17 and 1.19).

The aim of a task in supervised machine learning is to approximate the true underlying function f which describes the mapping from inputs $\boldsymbol{x}$ to outputs y for units drawn from joint distribution $p(\boldsymbol{x}, y)$ (see again Section 1.1). When evaluating how well a learning method is able to approximate function f, researchers ideally would want to know the expected generalization error that is the expectation of the loss function used for evaluation under the data generating distribution $p(\boldsymbol{x}, y)$ (see also Equation 1.17):[73]

$$\mathcal{EGE}(\hat{f}) = \int \int \mathcal{L}(y, \hat{f}(\boldsymbol{x}))p(\boldsymbol{x}, y)\, \mathsf{d}\boldsymbol{x}\, \mathsf{d}y \tag{1.73}$$

[72] Often the aim is not only to make inferences with regard to one task but across a range of tasks. But in the following, in order to reduce complexity, the focus will be on one task.

[73] Note that for reasons of readability, the notation here does not include parameter values $\boldsymbol{\theta}$. Note furthermore that the term *loss function* $\mathcal{L}(y_i, \hat{y}_i)$ typically denotes a function that measures the *discrepancy* between true and predicted values. If this is the case, then the loss function really captures an *error* and the smaller the loss value, $L = \mathcal{L}(y_i, \hat{y}_i)$, the better. However, in the context of evaluation, $\mathcal{L}(y_i, \hat{y}_i)$ is often a function that measures the *agreement* or *closeness* between true and predicted values. If this is the case, then the higher the value returned by the function, $L = \mathcal{L}(y_i, \hat{y}_i)$, the better. To consider all loss functions in a consistent framework, in the following, the terms *loss* or *error* are used even if the loss function also can indicate agreement or closeness.

As $p(\boldsymbol{x}, y)$ is unknown, the expected generalization error has to be approximated on the basis of observed data (Bischl et al., 2012, p. 251-252). In practice, a researcher only has at her disposal a single annotated data set of finite size. She can use one part of the observed data to train a model and then can use another part of the data (then named the test set) to estimate the generalization error of the trained model. The generalization error estimate produced from one such train-test split of the annotated data can be used as an estimate of the expected generalization error. This estimate of the expected generalization error, however, always will depend on the particular composition of the training set and the test set employed (James et al., 2013, p. 178). In order to produce an estimate of the expected generalization error that takes into account the variability of the loss function value that arises from different train-test set compositions, resampling techniques (e.g. cross-validation, bootstrapping) can be applied (see Section 1.1.1.4) (Bischl et al., 2012, p. 352). Since resampling techniques are rarely used in NLP, statements about NLP model performances are often based on single train-test splits, and thus are likely to be influenced by idiosyncrasies of the train-test split.

Note that this does not imply that statistical hypothesis tests are generally not conducted when comparing the performances of models in NLP: In some research articles and for some benchmark tasks, statistical hypothesis tests are carried out (Reimers & Gurevych, 2018, p. 1-2). Typically, the null hypothesis is that the performance of two models A and B is equal in the population and the alternative hypothesis is that model performances differ (Reimers & Gurevych, 2018, p. 3). One common way in NLP to implement a statistical hypothesis test is by means of making use of the bootstrap (Reimers & Gurevych, 2018, p. 4). Here, for a given test set D_{test} of size M, the difference in the prediction performance score of model A on D_{test} and the prediction performance score of model B on D_{test} is recorded (Riezler & Maxwell, 2005, p. 61). This difference here is indicated by $\delta(\mathsf{D}_{test})$. Then, K bootstrap samples of size M are drawn at random with replacement from D_{test} (Efron & Tibshirani, 1993, p. 45). On each of the K bootstrap test sets, the prediction performance of model A and model B is determined and their performance difference $\delta(\mathsf{D}^{\mathsf{k}}_{test})$ is calculated (Riezler & Maxwell, 2005, p. 61). Hence, one obtains a distribution of the differences in prediction performance values. This bootstrap sampling distribution then is shifted such that it is centered at zero and thus can be used to approximate the distribution of performance differences under the null hypothesis (which assumes that the expectation of performance differences in the population is zero) (Riezler & Maxwell, 2005, p. 61-62). The shifted $\delta(\mathsf{D}^{\mathsf{k}}_{test})$ here is indicated by $\delta(\mathsf{D}^{\mathsf{k}*}_{test})$. The null hypothesis is rejected if the share of bootstrap test sets for which $\delta(\mathsf{D}^{\mathsf{k}*}_{test}) \geq \delta(\mathsf{D}_{test})$ is smaller than a prespecified significance level (e.g. $\alpha = 0.05$) (Riezler & Maxwell, 2005, p. 61).[74]

This and similar hypothesis tests take into account that the test set is finite. Yet these tests do not take into account that for one test set data point x^*_m the same method when trained on another training set will yield a (slightly) different prediction for x^*_m. Bootstrapping on

[74]For a general introduction to the bootstrap for hypothesis testing see Efron & Tibshirani, 1993 chapter 16.

the test set thus is not a proper resampling procedure. The bootstrap sampling distribution of the differences in prediction performance values is not a distribution of the expected generalization error of the learning method. To obtain an adequate estimate of the expected generalization error, a resampling technique in which the method is repeatedly trained on different compositions of the training set and is evaluated on different compositions of the test set is required.

This first problem of NLP research procedures can be viewed as a specific instance of the more general second problem. The second problem is that in NLP inference-like statements are often made about methods although models (and not methods) are compared in analyses. This second problem is explicated in the following before the connection between the two problems will be elucidated.

In NLP shared task challenges, the aim is to build a processing system that learns the systematic mapping f from inputs $\boldsymbol{x}$ to outputs y on the basis of a training data set and then makes as accurate as possible predictions for instances in a test set. Therefore, in an NLP challenge, the aim is to build the best performing processing system. But *NLP as a science* seeks to develop—or improve upon—the components of such processing systems. Consequently, *NLP as a science* seeks to infer how one method (vs. another method) affects prediction performance.

A processing system is not just a learning algorithm, it rather is a procedural pipeline that, for example, may start with pretraining, move on to preprocessing of the target task training documents, and then implement hyperparameter tuning before finally conducting the training process on the target task. A whole collection of methods is involved in implementing such a pipeline. A processing system thus here is conceived of as an object under study that is composed of several methods.

A *method* can be defined at several levels of granularity. Methods that NLP researchers seek to make inferences about can be general and varied entities such as, for example, a learning approach (e.g. Collobert et al., 2011), a model architecture (e.g. Vaswani et al., 2017), or a set of pretraining resources (e.g. Liu et al., 2019b), but methods also can be more specific processing elements such as, for example, a pretraining objective (e.g. Devlin et al., 2019; Yang et al., 2019), or a hyperparameter setting in fine-tuning (e.g. Mosbach et al., 2021).

When implemented, a processing system produces a trained model $\hat{f}$ that (more or less well) approximates the true underlying function f. Based on an independent test set, function $\mathcal{L}(y, \hat{f}(\boldsymbol{x}))$ measures the discrepancy or agreement between the true values y and the values $\hat{y} = \hat{f}(\boldsymbol{x})$ that are predicted by the trained model. A trained model and its prediction performance as measured by $\mathcal{L}(y, \hat{f}(\boldsymbol{x}))$ thus can be understood as the materialized output of a processing system.

Given a population of processing systems of interest, the goal now is to infer the expected value of the effect that using method A compared to method B has on the prediction per-

formance of processing systems in the population. Just as other researchers want to make inferences about the effect of treatment A versus control B in a population of individuals, NLP researchers seek to draw inferences about the effect of treatment method A versus control method B in a population of processing systems. While individuals are characterized by their values on variables, processing systems are characterized by the specific methods they consist of.

In the simple case, a method indeed is like a value on a variable: A processing system consists of numerous elements and each of these elements can be conceived of as a variable v whose values are drawn from a set of methods, e.g. $v \in \{\textit{method}\, A, \textit{method}\, B\}$. For example: One element of a processing system is the pretraining objective. The pretraining objective can be conceived of as a variable that can take on values from a set of methods, which here could be for example {*language modeling*, *masked language modeling*}.

Even though a whole processing system, that is composed of various methods, is involved in approximating function f, the goal in NLP is to make inferences regarding the effect that one of these methods (compared to another method) has on the performance of the system. In a resource-rich world in which resources are plentiful but not unlimited, the drawing of inferences regarding the performance of one method compared to another method on a given task, could proceed as follows:[75] Two versions of one processing system are constructed. The versions are composed of identical methods, except at one point, where the first version applies method A and the second version implements method B. These two versions can be viewed as one individual processing system s that is once observed in the treatment group (method A) and once in the control group (method B). Let $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{A}(\boldsymbol{x}))]$ be the expectation of the performance of processing system s when implementing method A (where the expectation is with respect to the population of data points that are drawn from $p(\boldsymbol{x}, y)$). And let $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{B}(\boldsymbol{x}))]$ be the expectation of the performance of processing system s when implementing method B. $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{B}(\boldsymbol{x}))]$ then is the individual treatment effect (Holland, 1986, p. 947). The individual treatment effect gives the effect on the expected performance that the treatment of applying method A compared to method B has for individual processing system s (Holland, 1986, p. 947). (The fundamental problem of causal inference states that it is not possible to assess $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{A}(\boldsymbol{x}))]$ and $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{B}(\boldsymbol{x}))]$ on the same entity s at the same point in time (Holland, 1986, p. 947). There are several strategies that—when paired with specific assumptions—constitute ways via which the fundamental problem of causal inference can be overcome and inferences can be drawn (Holland, 1986, p. 947). With regard to the individual treatment effect, inference is possible under the assumptions of *temporal stability* and *causal transience* (Holland, 1986, p. 947), which here can be assumed to hold: The values of $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{A}(\boldsymbol{x}))]$ and $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{B}(\boldsymbol{x}))]$ do not depend on the particular point in time at which the treatment method A or the control method B are applied. $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathsf{s}}^{B}(\boldsymbol{x}))]$ thus will be constant over time (temporal stability). Moreover, if the two processing system

[75] In a world with unlimited resources, drawing inferences about the population would no longer be necessary because the entire population could be analyzed.

versions are implemented independently of each other without information (e.g. learned parameters) being able to pass from one system to the other, then the expected performance of processing system $\mathtt{s}$ when applied with treatment method A will not be affected by previously measuring the expectation of the performance of processing system $\mathtt{s}$ when applied with method B (causal transience). Hence, $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]$ here can be regarded as the causal effect on processing system $\mathtt{s}$'s expected performance due to the application of treatment method A compared to control method B.)

Overall, however, researchers are not interested in the individual treatment effect a method has on a particular processing system $\mathtt{s}$. Instead, researchers are interested in the average treatment effect of the method in the population of processing systems that a researcher seeks to infer to.[76] This is, NLP researchers do not seek to make statements like: 'For this particular processing system (i.e. with this particular configuration of methods in pretraining, preprocessing, hyperparameter tuning, and training) method A yields a better expected performance on task $\mathcal{T}$ than method B by $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]$'. Rather, NLP researchers seek to make statements like: 'On average method A yields a better expected performance on task $\mathcal{T}$ than method B by $E_{\mathtt{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]]$' (where the expectation is not only with respect to a population of data points but also a population of processing systems) (see also Reimers & Gurevych, 2018, p. 1-2). This is, in NLP the aim is to draw inferences about methods as tools, that can be plugged into a larger population of processing systems that is of interest when approaching some task. The aim is not to draw inferences about methods as parts of a single, concrete processing system.

As the size of the effect that method A vs. method B has on the performance is unlikely to be the same across different processing systems in the population (i.e. there is no unit homogeneity), the individual treatment effect $E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]$ is not an indicator of the average treatment effect in the population of processing systems (Holland, 1986, p. 948).[77] Thus, making inferences regarding the performance of one method vs. another method in a resource-rich world would imply drawing a random sample of processing systems from the population of processing systems and then applying each processing system once with method A and once with method B. The difference between the expectation of the expected value of the performance of the processing systems with method A and the expectation of the expected value of the performance of processing systems with method B, $E_{\mathtt{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))]] - E_{\mathtt{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]] = E_{\mathtt{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]]$, then gives an estimate of the average treatment effect in the population (Holland, 1986,

[76] All units in the population potentially have to be exposable to both compared methods (Holland, 1986, p. 946). For example, if method A is discriminative fine-tuning in which the global learning rate during fine-tuning is different for each layer and if in method B the global learning rate is the same for each layer, then the population of processing systems cannot include conventional machine learning methods that do not have a layered architecture like deep neural networks and thus cannot be exposed to methods A and B.

[77] Aßenmacher et al. (2021, p. 4), for example, show that the performance effect of increasing a model's depth or width is different for different model architectures.

p. 947). Subsequently, one could then conduct a hypothesis test with the null hypothesis being that $E_{\mathtt{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))] - E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{B}(\boldsymbol{x}))]] = 0$ in the population. Because each processing system is observed once in the treatment and once in the control group, this is a two-dependent-samples problem (Heumann et al., 2016, p. 210).

An alternative way would be to observe each individual processing system only under method A or under method B instead of in both states. If the processing systems are randomly assigned to the treatment or control group (and thus the assignment to method A or B is independent of other implemented methods of a processing system), then the difference between the average performance value of the processing systems in the treatment group, $E_{\mathtt{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}_{\mathtt{s}}^{A}(\boldsymbol{x}))]]$, and the average performance value of the processing systems in the control group, $E_{\mathtt{r}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \mathtt{f}_{\mathtt{r}}^{B}(\boldsymbol{x}))]]$, can be used as an estimate of the average treatment effect in the population (Holland, 1986, p. 948-949). In this case one would have a two-independent samples problem (Heumann et al., 2016, p. 210).

In a resource-rich world, the procedure described so far could be implemented to draw inferences about the performance of method A vs. method B in the population. In the real world with limited resources, the actually implemented NLP research procedures usually differ from what has been described here because of one or both of the following practices.

- Two processing systems are compared that not only differ with regard to whether method A or method B is applied but differ with regard to several methods (Aßenmacher & Heumann, 2020). Therefore, comparability is not given and it is unclear to which variation of a method a change in performance can be attributed to (Aßenmacher & Heumann, 2020).
- The comparison of applying method A vs. method B is conducted by incorporating one vs. the other method in only one (or a few) specific processing systems. Nevertheless, inferences are drawn about the performance of method A vs. B as general tools within an entire population of processing systems (Reimers & Gurevych, 2018). For example: Assume that a processing system is applied once with method A and once with method B. Furthermore assume that the performance of the processing system with method A is found to exhibit a higher performance than the processing system with method B and assume that the null hypothesis stating that the performance difference equals zero in the population is rejected on the basis of a hypothesis test that utilizes bootstrapping on the test set. The problematic aspect in NLP research now is that in such a setting the inference is drawn that method A in general (and not only when embedded in this particular processing system) will yield a higher performance than method B (Reimers & Gurevych, 2018).

When both of these problematic practices are combined in a research procedure, this is as if one would expose one individual to treatment A and would expose another individual, that differs from the first individual with regard to various relevant characteristics, to control B and then would conclude that the higher performance observed for the first

individual compared to the second is caused by the treatment of A vs. B and that thus, in the population in general, applying A compared to B will yield higher performance values.

Up to this point, two problems of NLP research practices have been identified. The first problem is that training and evaluation is often conducted on a single train-test split. Thus, reported performance measures are estimates of the generalization error instead of the expected generalization error. The second problem is that based on the comparison of a few models (that arise from probably incomparable processing systems) conclusions about the causal effect of methods in a larger population of processing systems are drawn.

Above it was stated that the first problem is an instance of the second more general problem (see page 140). Why is this the case? The question of how a training and a test set is created from a provided annotated data set and thus which data instances the training and the test set are composed of can be regarded as one method within a processing system. The train-test set composition is a characterizing element of a processing system. The processing systems in the population vary regarding the methods that they apply and they vary regarding the train-test set composition they use for training and evaluation. If one processing system were one individual that is observed across a set of variables and if the train-test set composition were one of these variables, then a researcher would like to know what the causal effect of treatment A vs. control B is not just for an individual with a particular value on the train-test set composition variable. The researcher would like to know what the average effect of treatment A vs. control B is in a population of individuals, where individuals in this population vary with respect to their values on the observed variables (including the variable of train-test set composition). The problem is that often exactly this knowledge about the population is not generated, because training and evaluation are often conducted on the basis of a single train-test set composition.

One can also pull this up from the other direction: So far, the expected generalization error

$$\mathcal{EGE}(\hat{f}) = \int \int \mathcal{L}(y, \hat{f}(\boldsymbol{x}))p(\boldsymbol{x}, y)\, \mathsf{d}\boldsymbol{x}\, \mathsf{d}y \tag{1.74}$$

(see also Equation 1.17) has been defined to be the expectation of the loss function of a learning method in a population of data points that are drawn from the data generating distribution $p(\boldsymbol{x}, y)$. Applying the terminology that is used in this subsection, this definition can be changed as follows: The expected generalization error is the expectation of the loss function of a processing system in a population of data points that are drawn from the data generating distribution $p(\boldsymbol{x}, y)$. Thus, the expected generalization error generalizes over a population of data points (which is what one wants here), but it still is the expected error of a *specific processing system* that is specific to the specific methods that it is composed of. To emphasize this, one can write $\hat{f}$ in Equation 1.74 as a specific processing system that results from applying a set of specific methods.

For this purpose, the following assumptions are introduced here: Assume that all processing systems in a population of interest can be described as $\mathsf{g}(v_1, v_2, v_3, v_4, \mathsf{D}_{train}, \mathsf{d}_{test})$.

The processing systems in the population consist of four processing elements v_1, v_2, v_3, v_4, where each element is a variable that can take on values over a set of methods, e.g. $v_1 \in \{A, B\}; v_2 \in \{C, G, H\}; v_3 \in \{D, I\}; v_4 \in \{F, J, Q, U\}$. The variables here are assumed to be independent and thus the joint probability of a specific method combination, e.g. $p(v_1 = A, v_2 = C, v_3 = D, v_4 = F)$, is $p(v_1 = A)p(v_2 = C)p(v_3 = D)p(v_4 = F)$. Also, each variable is assumed to take on each of its methods with equal probability. Thus, e.g. $p(v_1 = A) = p(v_1 = B)$ and $p(v_1 = A) + p(v_1 = B) = 1$. The joint distribution $p(v_1, v_2, v_3, v_4)$ hence assigns equal probability to each possible combination of methods.

Moreover, the processing systems operate on data. A single processing system s in this population of processing systems hence is not only characterized by the specific methods it applies but also by the specific data it uses. A processing system can be described as a specific combination of methods and data, e.g. $\mathsf{g}(v_1 = A, v_2 = C, v_3 = D, v_4 = F, \mathsf{D}^{\mathsf{s}}_{train}, \mathsf{D}^{\mathsf{s}}_{test})$. The input to a processing system s is a training set of raw data, $\mathsf{D}^{\mathsf{s}}_{train} = (d_i, y_i)_{i=1}^{N_{\mathsf{s}}}$, with a given composition and a given size N_{s}. The methods of a processing system (e.g. $\{A, C, D, F\}$) process and use the raw training data to produce a trained model $\hat{f}_{\mathsf{s}}$. $\hat{f}_{\mathsf{s}}$ then can make predictions for instances in the test set, $\mathsf{D}^{\mathsf{s}}_{test} = (d^*_m, y^*_m)_{m=1}^{M_{\mathsf{s}}}$, that likewise has a given composition and size M_{s}. The instances in the training and test set are assumed to be i.i.d. samples from $p(d, y)$, which is the joint distribution over raw data d and output values y. The performance of the processing system is evaluated by the loss function $\mathcal{L}$ that compares the true values for the instances in the test set with the values predicted by the system's trained model.

Note that a processing system thus has a data input and a method input. The assumption here is that these input components are mutually independent. (This implies, for example, that whether $v_1 = A$ or $v_1 = B$ will not affect the composition and/or size of $\mathsf{D}^{\mathsf{s}}_{train}$ and $\mathsf{D}^{\mathsf{s}}_{test}$.) The methods can be conceived of as functions that process the data and the data can be regarded as the inputs that are being processed.[78] The input variables to a processing system, $\{v_1, v_2, v_3, v_4, \mathsf{D}_{train}, \mathsf{D}_{test}\}$, can take values independently without influencing each other. As $\mathsf{D}^{\mathsf{s}}_{train}$ and $\mathsf{D}^{\mathsf{s}}_{test}$ are i.i.d. samples from $p(d, y)$, the joint distribution over input components of processing systems in the population can be described as $p(v_1, v_2, v_3, v_4, d, y)$. The expectation of the loss function in the population of processing systems of interest under raw data distribution $p(d, y)$ thus can be described as[79]

$$\sum_{v_4}\sum_{v_3}\sum_{v_2}\sum_{v_1}\int\int \mathcal{L}(y, \mathsf{g}(v_1, v_2, v_3, v_4, d, y))p(v_1, v_2, v_3, v_4, d, y)\,\mathsf{d}d\,\mathsf{d}y \tag{1.75}$$

Whereas Equation 1.75 is the expectation of the loss over data and methods, Equation

[78] Once the system's methods have begun to process the data, independence is no longer given: The parameters and the data representations that the system learns arise as a function of data and methods. But the inputs to the system, so the assumption here, are independent.

[79] The representation logic in Equation 1.75 here is as close as possible to the representation logic of Equation 1.74, which is known in the literature. $\sum_{v_1}$ here means the sum over all possible values that v_1 can take.

1.74, that is used in the machine learning literature, describes the expected generalization error of an individual processing system that generalizes over data $p(d, y)$ but is specific to the specific combination of methods that it applies. Using the framework introduced here, Equation 1.74 can be written as:[80]

$$\mathcal{EGE}(\mathsf{g}(v_1 = A, v_2 = C, v_3 = D, v_4 = F)) = \int\int \mathcal{L}(y, \mathsf{g}(v_1 = A, v_2 = C, v_3 = D, v_4 = F, d, y))p(d, y)\,\mathsf{d}d\,\mathsf{d}y \quad (1.76)$$

For NLP researchers, that aim at making inferences about the performance effects of single methods (rather than specific processing systems), the conditional expectation in Equation 1.76 is not particularly useful. Researchers that seek to estimate the expected loss of applying method A not only in a population of data points but also across a population of processing systems wish to have an estimate for

$$\mathcal{EGE}(\mathsf{g}(A)) = \sum_{v_4}\sum_{v_3}\sum_{v_2}\int\int \mathcal{L}(y, \mathsf{g}(v_1 = A, v_2, v_3, v_4, d, y))p(v_2, v_3, v_4, d, y)\,\mathsf{d}d\,\mathsf{d}y \quad (1.77)$$

which is the expected generalization error of a method A within a population of processing systems that are trained and evaluated on data points from data generating distribution $p(d, y)$. In contrast to the expected generalization error of a specific processing system presented in Equations 1.74 and 1.76, here all method components (except for method A) are averaged over.

The expected generalization error of method A in Equation 1.77 can be approximated via sampling methods (Bishop, 2006, p. 524): A sample of S processing systems is drawn independently from $p(v_2, v_3, v_4, d, y)$. The sampled processing systems vary with regard to the train-test set compositions they operate on and vary with regard to methods $\{v_2, v_3, v_4\}$ that they apply along their procedural pipeline. All processing systems, however, have in common that they apply method A. Each sampled processing system is implemented to produce a trained model that then generates predictions for instances in the test set. Then, for each sampled processing system, the loss function value is computed. If $\mathcal{L}(y^*_m, \mathsf{g}(A, v_2^{\mathsf{s}}, v_3^{\mathsf{s}}, v_4^{\mathsf{s}}, \mathsf{D}^{\mathsf{s}}_{train}, d^*_m))$ denotes the value of the loss function for the sth sampled processing system on an individual data point from the test set of processing system s, i.e. $(d^*_m, y^*_m) \in \mathsf{D}^{\mathsf{s}}_{test}$, then Equation 1.77 can be approximated as

$$\widehat{\mathcal{EGE}}(\mathsf{g}(A)) = \frac{1}{\mathsf{S}}\sum_{\mathsf{s}=1}^{\mathsf{S}} \frac{1}{|\mathsf{D}^{\mathsf{s}}_{test}|} \sum_{(d^*_m, y^*_m)\in \mathsf{D}^{\mathsf{s}}_{test}} \mathcal{L}(y^*_m, \mathsf{g}(A, v_2^{\mathsf{s}}, v_3^{\mathsf{s}}, v_4^{\mathsf{s}}, \mathsf{D}^{\mathsf{s}}_{train}, d^*_m)) \quad (1.78)$$

As described on pages 141 to 143 above,[81] given $\widehat{\mathcal{EGE}}(A)$ one can then estimate the causal

[80] Equation 1.74 uses $p(\boldsymbol{x}, y)$ whereas here—in order to emphasize that it is the processing system that transforms raw data into representations—$p(d, y)$ is used. Equation 1.76 is the expected generalization error of a processing system with the following combination of methods: $v_1 = A, v_2 = C, v_3 = D, v_4 = F$.

[81] What has been denoted as $E_{\mathsf{s}}[E_{\boldsymbol{x},y}[\mathcal{L}(y, \hat{f}^A_{\mathsf{s}}(\boldsymbol{x}))]]$ on page 143 above, here is estimated via Equation 1.78.

effect in the population that is due to using method A rather than method B by also applying all sampled processing systems $(1, \ldots, \mathsf{s}, \ldots, \mathsf{S})$ with method B and then estimating the expected generalization error of method B as

$$\widehat{\mathcal{EGE}}(\mathsf{g}(B)) = \frac{1}{\mathsf{S}} \sum_{\mathsf{s}=1}^{\mathsf{S}} \frac{1}{|\mathsf{D}^{\mathsf{s}}_{test}|} \sum_{(d^*_m, y^*_m) \in \mathsf{D}^{\mathsf{s}}_{test}} \mathcal{L}(y^*_m, \mathsf{g}(B, v^{\mathsf{s}}_2, v^{\mathsf{s}}_3, v^{\mathsf{s}}_4, \mathsf{D}^{\mathsf{s}}_{train}, d^*_m)) \tag{1.79}$$

For an individual sampled processing system s

$$\begin{aligned} \frac{1}{|\mathsf{D}^{\mathsf{s}}_{test}|} \sum_{(d^*_m, y^*_m) \in \mathsf{D}^{\mathsf{s}}_{test}} & \mathcal{L}(y^*_m, \mathsf{g}(A, v^{\mathsf{s}}_2, v^{\mathsf{s}}_3, v^{\mathsf{s}}_4, \mathsf{D}^{\mathsf{s}}_{train}, d^*_m)) \\ & - \frac{1}{|\mathsf{D}^{\mathsf{s}}_{test}|} \sum_{(d^*_m, y^*_m) \in \mathsf{D}^{\mathsf{s}}_{test}} \mathcal{L}(y^*_m, \mathsf{g}(B, v^{\mathsf{s}}_2, v^{\mathsf{s}}_3, v^{\mathsf{s}}_4, \mathsf{D}^{\mathsf{s}}_{train}, d^*_m)) \end{aligned} \tag{1.80}$$

gives an estimate of the individual treatment effect. The difference

$$\widehat{\mathcal{EGE}}(\mathsf{g}(A)) - \widehat{\mathcal{EGE}}(\mathsf{g}(B)) \tag{1.81}$$

gives an estimate of the average treatment effect in the population.

Alternatively, an estimate of the average treatment effect can be obtained by drawing an i.i.d. sample of S processing systems from $p(v_2, v_3, v_4, d, y)$ and then randomly assigning each processing system in the sample to either treatment method A or control method B. Subsequently, $\widehat{\mathcal{EGE}}(\mathsf{g}(A))$ is computed as in Equation 1.78 on the basis of those processing systems that have been assigned to treatment method A and $\widehat{\mathcal{EGE}}(\mathsf{g}(B))$ is computed as in Equation 1.79 using the processing systems assigned to control method B. Equation 1.81 then gives an estimate of the average treatment effect. This time, however, $\widehat{\mathcal{EGE}}(\mathsf{g}(B))$ and $\widehat{\mathcal{EGE}}(\mathsf{g}(A))$ come from two independent (rather than two dependent) samples.

Thus far, a situation has been described in which researchers seek to draw inferences about the performance effects of a single, fixed method. There are, however, situations in which researchers want to make inferences about methods in the sense of general and varied entities (e.g. learning approaches, model architectures, pretraining procedures). In this case, things get a bit more complicated. A method in this context is not a concrete value of a variable. A method in this context is a set of variables. A method in this more general broader sense can be described as $\boldsymbol{\mathcal{A}} = \{\mathcal{A}_{v_1}, \mathcal{A}_{v_2}, \ldots\}$, where $\mathcal{A}_{v_1}, \mathcal{A}_{v_2}, \ldots$ are variables and the values of each variable are drawn from a set of concrete method components, e.g. $\mathcal{A}_{v_1} \in \{A, B\}, \mathcal{A}_{v_2} \in \{C, G, H\}$. Another method $\boldsymbol{\mathcal{B}}$ can be denoted by $\boldsymbol{\mathcal{B}} = \{\mathcal{B}_{v_1}, \mathcal{B}_{v_2}, \ldots\}$, where e.g. $\mathcal{B}_{v_1} \in \{D, I\}, \mathcal{B}_{v_2} \in \{F, Q\}$. Hence, the variables in $\boldsymbol{\mathcal{B}}$ as well as the sets from which the variables in $\boldsymbol{\mathcal{B}}$ draw, need not necessarily be the same as for $\boldsymbol{\mathcal{A}}$. For example, if $\boldsymbol{\mathcal{A}}$ were transfer learning with Transformers and $\boldsymbol{\mathcal{B}}$ were conventional machine learning, then one may have something as $\boldsymbol{\mathcal{A}} = \{\mathcal{A}_{v_1} = \textit{pretraining objective}, \mathcal{A}_{v_2} = \textit{pretraining corpus}, \mathcal{A}_{v_3} = \textit{Transformer architecture}, \ldots\}$ and e.g. $\mathcal{A}_{v_1} = \textit{pretraining objective} = \{\textit{language modeling}, \textit{masked language modeling}\} \ldots$ whereas for $\boldsymbol{\mathcal{B}}$ this could

be $\boldsymbol{\mathcal{B}} = \{\mathcal{B}_{v_1} = \textit{lowercasing}, \mathcal{B}_{v_2} = \textit{weighting of elements in document-feature matrix}, \mathcal{B}_{v_3} = \textit{learning algorithm,}\dots\}$ and e.g. $\mathcal{B}_{v_1} = \textit{lowercasing} = \{\textit{yes}, \textit{no}\}\dots$ Note that there is a set of possible method component combinations for each method. $\boldsymbol{\mathcal{A}}^{\mathsf{s}} = \{\mathcal{A}_{v_1} = A, \mathcal{A}_{v_2} = G, \dots\}$ here denotes one possible combination of method components under approach $\boldsymbol{\mathcal{A}}$.

If the population of processing systems of interest is $\mathsf{g}(\mathcal{V}, v_3, v_4, \mathsf{D}_{train}, \mathsf{D}_{test})$, where $\mathcal{V}$ can be method $\boldsymbol{\mathcal{A}}$ or $\boldsymbol{\mathcal{B}}$, i.e. $\mathcal{V} \in \{\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}\}$, and if $\boldsymbol{\mathcal{A}}^{\mathsf{s}}$ is a specific combination of method components under approach $\boldsymbol{\mathcal{A}}$, then the estimation of the expected generalization error of applying method $\boldsymbol{\mathcal{A}}$ in a population of processing systems under $p(d, y)$ can be achieved by drawing a sample of S processing systems from $p(\boldsymbol{\mathcal{A}}, v_3, v_4, d, y)$ and then computing

$$\widehat{\mathcal{EGE}}(\mathsf{g}(\boldsymbol{\mathcal{A}})) = \frac{1}{\mathsf{S}} \sum_{\mathsf{s}=1}^{\mathsf{S}} \frac{1}{|\mathsf{D}^{\mathsf{s}}_{test}|} \sum_{(d^*_m, y^*_m) \in \mathsf{D}^{\mathsf{s}}_{test}} \mathcal{L}(y^*_m, \mathsf{g}(\boldsymbol{\mathcal{A}}^{\mathsf{s}}, v^{\mathsf{s}}_3, v^{\mathsf{s}}_4, \mathsf{D}^{\mathsf{s}}_{train}, d^*_m)) \tag{1.82}$$

Hence it is sampled from all possible method component combinations that arise under method $\boldsymbol{\mathcal{A}}$ and then it is averaged also over these combinations.[82] The same procedure can be repeated under method $\boldsymbol{\mathcal{B}}$ and then $\widehat{\mathcal{EGE}}(\mathsf{g}(\boldsymbol{\mathcal{A}})) - \widehat{\mathcal{EGE}}(\mathsf{g}(\boldsymbol{\mathcal{B}}))$ gives an estimate of the average treatment effect of implementing method $\boldsymbol{\mathcal{A}}$ vs. $\boldsymbol{\mathcal{B}}$ in the population.

The outlined research procedures would allow drawing inferences about the performance effect of method A (or $\boldsymbol{\mathcal{A}}$) compared to method B (or $\boldsymbol{\mathcal{B}}$) in some population of processing systems for some task $\mathcal{T}$. One reason why in the field of NLP these outlined procedures are not implemented and instead rather problematic research practices (that have been described on page 143) are used, is that resources are limited. The procedures described here require the training of a sample of processing systems, and the sample should have an appropriate size. But already the implementation of one pretraining run usually consumes considerable amounts of resources (see Aßenmacher & Heumann, 2020, p. 6). And also the training process on the target task can (depending on the model architecture, the data set size, and document lengths) take a substantive amount of time and computational resources (see p. 53 in this dissertation's article *Introduction to neural transfer learning with Transformers for social science text analysis*).

1.4.1.2 Research Procedures in This Thesis

Because of resource constraints, the procedures implemented in this dissertation are much closer to the usual NLP research practices than to the procedures described here. The following paragraphs give an overview of the research procedures used in this dissertation. A tabular summary is given in Table 1.5.

[82] Note that the assumption here is that each combination of method components $\boldsymbol{\mathcal{A}}^{\mathsf{s}}$ has equal probability.

Article	Comparison of ...	Number of applied processing systems	Sampling from population of processing systems?	Number of applied train-test set compositions	Can an estimate of Equation 1.76 be computed?	Can an estimate of Equation 1.77 be computed?
Introduction to neural transfer learning with Transformers for social science text analysis	Approach $\boldsymbol{\mathcal{A}}$: Transfer learning with Transformer-based models	2	no	*Ethos*, *Abortion*: 1 *Wikipedia*: 5×5	*Ethos*, *Abortion*: no *Wikipedia*: yes	no
	Approach $\boldsymbol{\mathcal{B}}$: Conventional machine learning	4	no	*Ethos*, *Abortion*: 1 *Wikipedia*: 5×5	*Ethos*, *Abortion*: no *Wikipedia*: yes	no
How to estimate continuous sentiments from texts using binary training data	Approach $\boldsymbol{\mathcal{A}}$: CBMM	2	no	1	no	no
	Approach $\boldsymbol{\mathcal{B}}$: lexicon-based	2	no	1	no	no
	Approach $\boldsymbol{\mathcal{C}}$: regression	2	no	1	no	no
A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis	Approach $\boldsymbol{\mathcal{A}}$: active learning	2	no	10×15 $[5 \times 15]$	yes	no
	Approach $\boldsymbol{\mathcal{B}}$: passive learning	2	no	10×15 $[5 \times 15]$	yes	no
	Approach $\boldsymbol{\mathcal{C}}$: topic model-based	1,706,900	no	does not apply	does not apply	no
	Approach $\boldsymbol{\mathcal{D}}$: query expansion	1800	no	does not apply	does not apply	no
	Approach $\boldsymbol{\mathcal{E}}$: keyword list	100	no	does not apply	does not apply	no

Table 1.5: **Research Procedures in This Dissertation.** Tabular overview of the content presented in the text. The number of applied processing systems here is the number of systems that are applied on a single train-test set composition. Since all articles compare methods in terms of larger method compositions (approaches), estimates of the expectations expressed in Equations 1.76 and 1.77 must be calculated with respect to these larger approaches (as in Equation 1.82).

The article *Introduction to neural transfer learning with Transformers for social science text analysis* compares conventional machine learning methods to transfer learning with Transformer-based models. Thus, here the aim is not to compare a single method A to a single method B but to compare a larger strand of methods $\boldsymbol{\mathcal{A}}$ to a larger strand of methods $\boldsymbol{\mathcal{B}}$. Especially with regard to pretraining and text preprocessing, processing systems that apply conventional machine learning have a fundamentally different structure than processing systems using deep learning in a transfer learning setting. In each of the three applications reported in the article, two processing systems with Transformers and transfer learning are compared to four processing systems with conventional learning methods. Hence, there are six processing systems, each with a specific pretraining method (if any), a specific text preprocessing method, followed by a specific hyperparameter tuning procedure, and a specific method for training. None of these six processing systems are sampled from a distribution over processing systems. (Defining such a distribution would be highly difficult and would raise numerous additional questions.) Rather, the method components of each processing system are selected deliberately on the basis of various motives such as, for example,

- personal experience (A binary weighting of the elements in the document-feature matrix had proven to perform quite well across various classification tasks in the past.)
- the goal to demonstrate the basics of transfer learning and Transformers (Hence, the seminal Transformer-based BERT model (Devlin et al., 2019) within a common sequential transfer learning setting is applied.)
- practical reasons (The Longformer (Beltagy et al., 2020) was one of the few pretrained Transformer models that was able to process more than 512 tokens and was available at Hugging Face's Transformer library (Wolf et al., 2020) at that time. Therefore, the Longformer is applied.)
- the intention to connect with the literature (GloVe word embeddings rather than word2vec embeddings are used because GloVe tends to be more frequently employed in social science (Rodriguez & Spirling, 2022, p. 104). The search over hyperparameter values is conducted within ranges that are common in the NLP literature.)

In the *Ethos* and *Legalization of Abortion* applications in the article *Introduction to neural transfer learning with Transformers for social science text analysis*

1. the available annotated data came with a predefined division into a training data set and a test data set. This single train-test set split is maintained.
2. Hyperparameter tuning is conducted via five-fold cross-validation on the training data set.
3. The processing system with the best performing hyperparameter setting is selected and trained once on the entire training data set. The trained model then is evaluated on the test set. The trained model's prediction performance on the test set is

reported.

Even if the here implemented five-fold cross-validation is likely to be a more reliable method for hyperparameter tuning than merely using a single division into one training and one validation set as is typically done in NLP, the core elements of the procedures applied here follow the research practices in NLP (see again page 138). Hence, each reported performance value is an estimate of the generalization error of a specific model that has been trained on a single specific training set, is evaluated on a specific test set, and results from applying a processing system of specific methods. Based on the small number of six deliberately selected processing systems operating on a single train-test split, inferences cannot be drawn about the general performance of Transformer-based models for transfer learning vs. conventional methods for the population of data points from $p(d, y)$. Statements can only be made for the specific processing systems compared and the specific training-test set composition used.

The research procedure applied in the *Wikipedia Toxic Comment* application of the article *Introduction to neural transfer learning with Transformers for social science text analysis* allows for slightly more far-reaching statements to be made. Here, five training-test set compositions are drawn uniform at random from the set of all available labeled data. For each of the six processing systems compared in the article, training and evaluation thus are conducted five times, each time on a differently composed training and test set. For each of the six processing systems, the mean of the performance values across the five train-test set pairs is reported. Even though the number of five train-test set compositions is not particularly high, the reported mean value still gives an estimate of the expected generalization error of a processing system as in Equation 1.76. Conclusions that can be drawn here are still specific to the specific methods that a processing system is built from, but they are no longer specific to a specific train-test set composition.

To additionally inspect the effect that differently sized training data sets have on prediction performance, four increasingly smaller training data sets are randomly sampled from each of the five originally sampled training data sets. For each of the six processing systems, training and evaluation thus is conducted on $5 \times 5 = 25$ different train-test set pairs that vary in composition and size. This procedure, in which training set sizes and compositions are taken into account, already differs substantively from usual NLP practices and is one step closer to the more adequate research procedures outlined above.

Closer to the usual NLP practices is the procedure taken in the article *How to estimate continuous sentiments from texts using binary training data*. The article first introduces CBMM and then empirically assesses the performance of CBMM in producing continuous sentiment estimates across three applications. CBMM is compared to lexicon-based approaches and is compared to regression approaches that are fed with fine-grained data in training. Hence, the article likewise does not compare individual, specific methods but larger method compositions (approaches $\boldsymbol{\mathcal{A}}$ = CBMM, $\boldsymbol{\mathcal{B}}$ = lexicon-based, and $\boldsymbol{\mathcal{C}}$ = regression). However, this comparison of approaches is conducted on the basis of a few specific processing systems (two systems for CBMM, two systems for lexicon-based approaches,

and two systems for regression approaches) whose components are deliberately chosen. Therefore, the processing systems that are applied and evaluated in the article are not randomly drawn from a population of processing systems. Moreover, the comparison for all three applications is based on a single train-test split of the available labeled data. Consequently, the reported performances are estimates of generalization errors of specific processing systems that are based on specific train-test set compositions.

Nevertheless, several individual treatment effects can be observed in the article. For example, an individual CBMM processing system is implemented once with and once without a dispersion formula.[83] Because all other elements of the CBMM processing system are held constant, the observed performance difference between the CBMM processing system with and without a dispersion formula is the individual treatment effect of the dispersion formula on the processing system's performance. Moreover, when comparing CBMM with regression approaches, care is taken to ensure that for one processing system all method components that can be the same for CBMM and regression approaches indeed are the same. For example, for CBMM as well as for regression approaches, the pretrained representation model RoBERTa (Liu et al., 2019b) is used as an input for fine-tuning on the sentiment prediction target task.[84] Therefore, since all components of a processing system are the same across CBMM and regression approaches (except for those components that necessarily have to be different), the performance difference observed between CBMM and regression can be understood as the individual treatment effect of applying CBMM vs. regression on the performance of the applied processing system.

The article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* aims at comparing the retrieval performances of different approaches: keyword lists, query expansion techniques, topic model-based classification rules, and active as well as passive supervised learning. Hence, again the aim is to compare not single specific methods but larger categories of methods (here called approaches $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{E}}$). In the article, considerable efforts are made to have not only one processing system per approach but to have several processing systems per approach. This is, efforts are made to have not only one single processing system that implements one specific combination of method components $\boldsymbol{\mathcal{A}}^{\mathsf{s}}$ as a representation for one approach $\boldsymbol{\mathcal{A}}$ but to have several processing systems with varied method component combinations $\{\boldsymbol{\mathcal{A}}^{\mathsf{s}}\}_{\mathsf{s}=1}^{\mathsf{S}}$ as a representation for one approach $\boldsymbol{\mathcal{A}}$. In each of its three applications, the article compares the performances of 100 different keyword lists with the performances of 1,800 query expansion results, 1,706,900 different topic model-based classification rules, 300 models obtained via passive supervised learning, and 300 models obtained via active supervised learning.[85] With regard to active and passive supervised learning, for example,

[83] The dispersion formula is a part of the mixed model that is estimated on the basis of the classifiers' predictions.

[84] Moreover, in both cases the predictions from the RoBERTa models for the test set documents are aggregated via a mixed model. However, in order to use adequate models, a beta mixed model is estimated for CBMM and a linear mixed model is estimated for regression approaches.

[85] In the application based on the Reuters-21578 corpus (Lewis, 1997), 150 rather than 300 models trained

two different learning algorithms (SVM and BERT) are applied on ten differently composed tests sets and initial training sets.[86] For each of the ten initial training sets, a model is trained on the initially sampled training data set and then generates predictions for the pool of data instances that neither belong to the test nor the training set. Given these predictions, additional instances from the pool are either selected randomly (passive learning) or by uncertainty sampling (active learning) to be added to the training set. This process is repeated 15 times, and the training set increases with each iteration.

Again, the processing systems are not drawn from an a priori, clearly defined population of processing systems. Within one approach, some method components are made to vary over a predefined range of values while other method components are held constant. For practical reasons, components for which it is easier to let them vary (because letting them vary consumes few resources), are allowed to vary over a larger range of values than method components for which letting them vary is more resource intensive. A good example of the approach taken in the article is the procedure implemented with regard to topic model-based classification rules: Let $\boldsymbol{\mathcal{A}} = \{\mathcal{A}_{v1}, \mathcal{A}_{v2}, \mathcal{A}_{v3}, \mathcal{A}_{v4}\}$ indicate the approach of applying topic model-based classification rules, where $\mathcal{A}_{v1} \in \{$*LDA, CTM, STM,*$\dots\}$ is the type of topic model, $\mathcal{A}_{v2} \in \{1, 2, 3, 4, \dots\}$ is the topic model's number of topics, $\mathcal{A}_{v3} \in \{\{$*Topic 1*$\}, \{$*Topic 1, Topic 2*$\}, \{$*Topic 1, Topic 3*$\}, \dots\}$ is the set of topics that are considered relevant, and $\mathcal{A}_{v4} \in [0, 1]$ is the threshold value that indicates the minimum share of a document that has to be assigned to the topics in the relevant set for the document to be categorized as relevant. The procedure in the article selects for each variable a set of values—namely: $\mathcal{A}_{v1} = \{CTM\}$, $\mathcal{A}_{v2} \in \{5, 15, 30, 50, 70, 90, 110\}$, $\mathcal{A}_{v3} \in \{$*all sets of one, two, and three topics that can be drawn from* $\mathcal{A}_{v2}$ *topics* $\}$, $\mathcal{A}_{v4} \in \{0.1, 0.3, 0.5, 0.7\}$—and then inspects the retrieval performance for each combination of these values. Thus, there is no random sampling from the entire population of processing systems of interest. Rather, all possible method combinations over sets of predefined ranges of values are applied and evaluated. The procedure described here is similar for the other approaches that the article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* compares. This procedure makes it possible to compare the approaches based on ranges of method combinations. (And thus there is a substantively broader comparison basis than if only one processing system per approach were applied). Yet the comparison is only possible with respect to the selected method combination ranges. Reported mean performances are not an estimate of Equation 1.82.

To conclude: In all of this dissertation's articles, larger strands of methods (approaches $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \dots$) are compared against each other. In each article, at least one of the approaches (in the following denoted with $\boldsymbol{\mathcal{A}}$) is one that is not yet widely used in political science and could valuably extend the toolbox that is employed in text-based political

by passive and active supervised learning are compared.

[86] In the application based on the Reuters-21578 corpus five rather than ten differently composed tests sets and initial training sets are applied.

science research. Using the available amount of resources, efforts were made to compare method $\boldsymbol{\mathcal{A}}$ to the other more established methods $\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \ldots$ in a way that would allow making useful comparative statements. The results of these efforts are as follows: On the basis of the reported performance values in the article *How to estimate continuous sentiments from texts using binary training data* individual treatment effects can be computed. With regard to the *Wikipedia Toxic Comment* application of the article *Introduction to neural transfer learning with Transformers for social science text analysis* estimates of expected generalization errors that generalize over train-test set compositions are presented. And the article *A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis* implements a broad comparative approach. The research procedures applied in this dissertation thus partly go beyond research practices in NLP. However, none of the articles fully applies the research procedures described above, which would allow drawing inferences about the causal effect on performance of applying method $\boldsymbol{\mathcal{A}}$ in comparison to method $\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \ldots$ within the population of processing systems of interest.

What could have been done better in the dissertation articles with the given limited resources? The available resources could have been distributed better. For example, in each article whenever a supervised learning algorithm is employed, a grid search across hyperparameter value combinations is implemented via five-fold cross-validation on the training set to determine the best performing hyperparameter setting. Since this combination of grid search and cross-validation involves training the algorithm five times for each explored combination of hyperparameter values on a portion of the training data set, this step requires substantial resources. Moreover, the usefulness of this step is questionable, since there are often small performance differences within certain hyperparameter value ranges. Additionally, from a scientific point of view, knowing the mean performance of a method across ranges of hyperparameter values is arguably more of interest than knowing the performance of a method with the best performing hyperparameter setting. The resources used for hyperparameter tuning could therefore have been employed more efficiently. This is especially the case with regard to those applications in which hyperparameter tuning is followed by training and evaluation on a single train-test split of the labeled data. Instead of training a processing system with tuned hyperparameter values once on a specific training data set and evaluating it once on a specific test data set, it would have been more insightful for a processing system (with fixed, non-tuned hyperparameter values) to be trained and evaluated on different train-test set compositions because then an estimate of the expected generalization error of the processing system (see Equation 1.76) could have been computed. Future research projects thus should apply provided resources such that the broadest possible scientific inferences can be drawn.

1.4.1.3 NLP as a Science

In general, the field of NLP would benefit if it perceived itself as a science and would move away from engineering the best performing processing system for a task and move toward applying scientific research procedures that allow drawing inferences about the performance effects of methods. Studies that make processing systems comparable and then examine individual treatment effects of single method components (e.g. Aßenmacher et al., 2021) are an important first step in this direction. A further step then would be to apply the research procedures described here in order to estimate average treatment effects. For this purpose, NLP researchers have to (begin to) see themselves as scientists. They first have to discuss

- what advances it could bring to the field if they saw themselves as scientists (on the benefits of scientific engineering see e.g. Montgomery, 2012, p. 1-5),
- what their scientific research interest is (namely drawing causal inferences about the performance effects of methods), and
- how to operate as scientists (i.e. what research procedures to use such that causal inferences can be drawn).

This discussion process has begun already: In a recent, award-winning article, Ulmer et al. (2022) summarize suggestions for the improvement and adoption of scientific research practices in NLP.

Once this discussion process is over, new research practices are established, and respective studies are planned, a difficulty will be to define populations of processing systems from which the sampling methods draw and that the resulting inferences refer to. Against the background of limited resources, it might be helpful in a concrete application to first define a smaller population than the one actually of interest, and then—as outlined above—to sample from this somewhat smaller population and estimate an average treatment effect accordingly. This would then allow drawing causal inferences at least with respect to a smaller population.[87]

[87]For example: The CBMM approach introduced in the article *How to estimate continuous sentiments from texts using binary training data* can be conceived of as a general category of processing systems $\boldsymbol{\mathcal{A}}$ that produces continuous estimates based on binary training data. When estimating the expected generalization error of this approach $\boldsymbol{\mathcal{A}}$ (as in Equation 1.82), then one would first have to define a population of processing systems with $\boldsymbol{\mathcal{A}}$ from which to sample from. This population is very large: In its second step, the CBMM procedure trains and applies an ensemble of classifiers and in the article it says: "The classifiers in the ensemble may differ regarding the type of algorithm, hyperparameter settings, or merely the seed values initializing the optimization process." (p. 185). Additionally, the number of classifiers that are employed in the ensemble can vary across a vast (potentially infinite) range of positive integer values $\{2, 3, 4, \dots\}$ and in the third step of CBMM the mixed model can be applied with or without a dispersion formula. The population of interest furthermore is characterized by $p(d, y)$. Drawing and applying an appropriately sized sample of processing systems from this population, is likely to be prohibitively costly. A feasible option, however, would be to define a subpopulation of this population such that a manageable number of processing systems can be randomly sampled and applied. The mean across the sampled processing

Why are all these mentioned aspects also important for political science? The goal of this dissertation is to present and explain methods that are not yet widely used (or are just beginning to be used) in text-based political science research such that political scientists understand these methods and the political science research toolbox is extended and improved. This is an important contribution. Yet it is equally important for political scientists to not only have an extended toolbox, but also to know which elements in this toolbox are likely to show which performance with regard to which task. If political scientists apply methods from NLP for the purpose of measuring a priori-defined concepts from texts, then the primary goal is to reach as high as possible prediction performances. Consequently, political scientists rely on valid inferences about the causal performance effects of NLP tools.

1.4.2 Future Research in Political Science

For political scientists there are two further future research activities: First, political scientists should ensure that concepts and methods continuously travel from NLP to political science. There is exchange between the field of political science and the field of NLP, but this exchange does not reach far enough within political science. Big, important NLP conferences such as ACL, EMNLP, or NAACL have sessions on computational social science and every now and then political scientists (often in collaboration with computer scientists) participate and apply current NLP methods (e.g. Kim et al., 2021; Rehbein et al., 2021b,c). Furthermore, there are tutorials and workshops that explicitly aim at promoting the exchange between political scientists and the NLP community (Glavaš et al., 2019; Rehbein et al., 2021a). The larger part of text-based political science researchers, however, seems disconnected from the continuing stream of developments within NLP. One likely reason for this disconnect is that the importance of different publication formats is weighted differently in NLP and political science. In NLP, publication at prestigious conferences plays an all-important role. In political science, publication in prestigious journals is considered most important. For political science—which is an interdisciplinary discipline par excellence and relies on theories and methods from other scientific fields—it would be very important not to disregard NLP publications only because they come in the form of conference papers.

A continuous exchange between political science and NLP is especially important for political scientists in order not to lose touch with the concepts and methods that are developed

system's performance values then is an estimate of the expected generalization error of CBMM within the defined subpopulation. In a concrete, exemplary case, the joint distribution of a subpopulation could be $p(\boldsymbol{\mathcal{A}} = \textit{CBMM}, d, y)$ where $\textit{CBMM} = \{\textit{type of classifiers, number of classifiers, seed values, dispersion formula}\}$ and where $\textit{type of classifiers} \in \{\textit{BERT, RoBERTa, BART}\}$, $\textit{number of classifiers} \in \{5, 10, 20\}$, $\textit{seed values} \in \{1111, \ldots, 9999\}$, $\textit{dispersion formula} \in \{\textit{yes, no}\}$. Moreover, the underlying data distribution $p(d, y)$ in practice is approximated via concrete pairs of training and test data sets, $(\mathsf{D}_{train}, \mathsf{D}_{test})$. Here, for example, $(\mathsf{D}_{train}, \mathsf{D}_{test}) \in \{\textit{all possible train-test set pairs that can be drawn from the available data of size } N, \textit{ where } |\mathsf{D}_{train}| = 0.8 \times N \textit{ and } |\mathsf{D}_{test}| = N - |\mathsf{D}_{train}|$.

and applied in NLP. This doctoral thesis attempts to fill a gap that has emerged: NLP had moved from using conventional machine learning methods to employing deep neural networks and transfer learning, but these changes had not yet fully reached political science. The specific methods presented in this dissertation very soon will be out of date. The underlying concepts and methods presented in this dissertation (deep learning, contextualized representations, transfer learning, ...) are likely to be longer in use and are likely to be the basis from which new methods in NLP will (and already do) arise. But, eventually, new fundamental developments of methods will come and new gaps between the state-of-the-art in NLP and research practice in political science will emerge. Insofar as political science research benefits from advancements in NLP (e.g. because the invented methods facilitate the empirical analysis of political science research questions), political scientists should take care not to be left behind by the progress in NLP.

Recent developments within NLP and related fields (Ruder, 2021a, 2022) that this dissertation has not addressed (or not addressed extensively) but that could be further valuable tools in the political science toolbox are

- multimodal models (covering modalities such as text and speech (Bapna et al., 2021), text and image (Radford et al., 2021), or text and video (Fu et al., 2021)),
- speech representation models (Baevski et al., 2020; Liu et al., 2020; Babu et al., 2021),
- cross-lingual learning and multilingual models (Devlin, 2019; Conneau et al., 2020; Xue et al., 2021),
- in general: better and/or more efficient ways to learn language representations and to process natural language, e.g. via
  - more efficient Transformer architectures (Tay et al., 2021),
  - few-shot learning, zero-shot learning, and in-context-learning (Yin et al., 2019; Brown et al., 2020; Gao et al., 2021; Schick & Schütze, 2021; Wei et al., 2022),
  - the increased application of multitask learning (Brown et al., 2020; Wei et al., 2022; Aribandi et al., 2022),
  - new neural network architectures that extend or move beyond the Transformer (Jaegle et al., 2021, 2022),
  - token-free models that directly take in raw text and do not require textual inputs to be separately processed by tokenization algorithms (Tay et al., 2022; Clark et al., 2022; Xue et al., 2022).

The second issue that political science research has to address is the handling of prediction error (as well as other forms of measurement error) that can occur when applying a learning algorithm for the purpose of measuring an a priori-specified concept from text: Assume that a political scientist seeks to estimate the average causal effect that a latent individual

property H has on some outcome Γ in a population of observational units. The scientist observes an i.i.d. sample of observational units $(1, \ldots, v, \ldots, V)$ from the population. She records the value that each unit in the sample assumes on variable Z (which serves as an indicator of outcome Γ). $(\eta_1, \ldots, \eta_v, \ldots, \eta_V)$ are the realized but unobserved values of the units on latent property H. In order to measure the latent property, the researcher makes use of text data. Each observational unit in the sample is the author of one text document d_v. (Note here that for the vth unit, d_v does not directly reflect η_v but θ_v. Whereas η_v is the true value of unit v's latent characteristic, θ_v is the value of the latent characteristic that unit v communicates in text d_v (see pages 57 to 59 and Figure 1.1).) The researcher reads a subsample of documents $(d_i)_{i=1}^N$ (the training set) and assigns to each d_i in the training set a value y_i. y_i is the value of the latent characteristic of i that the researcher assigns by means of trying to decode θ_i from d_i while reading. Given $(d_i, y_i)_{i=1}^N$, the researcher trains a supervised learning algorithm. Then, she uses the trained model $\hat{f}$ to generate predictions. These predictions, however, will not be without error. The discrepancies between the predictions $\hat{y}$ and the true assigned values y for units in the training set is the training error. Due to overfitting, these discrepancies are likely to be larger in the entire population as captured by the expected generalization error. If the political scientist now takes the trained model's predictions $\hat{y}$ as her indicator of latent concept H and uses the predictions in her subsequent analysis of the effect of H on Γ without adjustments, then there are two problematic aspects here:

First, the political scientist does not take into account the uncertainty in the machine learning model's predictions. In doing so, she is not alone: In political science research practice, it is usually the case that predictions from trained machine learning models are included in regression models without considering the prediction error (e.g. Theocharis et al., 2016; Katagiri & Min, 2019; Mitts, 2019; Fowler et al., 2021).[88] Ignoring prediction error, however, can bias the substantive conclusions drawn in an analysis (Fong & Tyler, 2021, p. 480-481). Techniques that account for this form of measurement error are discussed and presented in political science (e.g. Fong & Tyler, 2021). (And closely related here is the research field on missing data and imputation methods (Little & Rubin, 2002; Toutenburg et al., 2004).) But there are still open questions. With regard to the method proposed in Fong & Tyler (2021), for example, it is unclear how best to proceed if the units' values on the dependent variable (rather than an independent variable) are produced by a prediction model or how to apply their method to the case of nonlinear regression. The task of future research, therefore, is to develop, refine, and apply techniques that account for the prediction error from machine learning models.

Second, the political scientist in the example furthermore should be aware that prediction error is not the only source of measurement error here. It is not only that the predicted values $\hat{y}$ may deviate from true assigned values y but that the y themselves arise from a stochastic process. This process maps an individual unit's true latent characteristic η_v into a value of the characteristic that unit v seeks to communicate, ψ_v, and then maps

[88] This is also true beyond political science.

ψ_v into the value that the unit actually communicates, θ_v, by means of producing text d_v (see pages 57 to 59 and Figure 1.1 above). y_v then is the result of a stochastic coding process in which a human while reading d_v seeks to decode θ_v from d_v. So far, this aspect has received little attention in political science. There are very few studies that (1) have devised methods that take into account stochastic elements of this process that maps η_v into y_v and/or (2) demonstrate the problems for inferences drawn if these elements are not accounted for (Benoit et al., 2009; Hopkins & King, 2010; Mikhaylov et al., 2012). Further research is needed to develop methods by which to account for this process from η_v to y_v.

Bibliography

Agrawal, H., Desai, K., Wang, Y., Chen, X., Jain, R., Johnson, M., Batra, D., Parikh, D., Lee, S., & Anderson, P. (2019). nocaps: Novel object captioning at scale. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, 8947–8956. https://doi.org/10.1109/ICCV.2019.00904

Ahlquist, J. S. & Breunig, C. (2012). Model-based clustering and typologies in the social sciences. *Political Analysis*, *20*(1), 92–112. https://doi.org/10.1093/pan/mpr039

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211. https://doi.org/10.1016/0749-5978(91)90020-T

Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In E. M. Bender, L. Derczynski, & P. Isabelle (Eds.), *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 1638–1649). Association for Computational Linguistics. https://aclanthology.org/C18-1139

Albarracin, D., Sunderrajan, A., Lohmann, S., Chan, M.-p. S., & Jiang, D. (2019). The psychology of attitudes, motivation, and persuasion. In D. Albarracin & B. T. Johnson (Eds.), *The Handbook of Attitudes* (pp. 3–44). Routledge. https://doi.org/10.4324/9781315178103

ALDayel, A. & Magdy, W. (2021). Stance detection on social media: State of the art and trends. *Information Processing & Management*, *58*(4), 102597. https://doi.org/10.1016/j.ipm.2021.102597

Alrababa'h, A. & Blaydes, L. (2021). Authoritarian media and diversionary threats: Lessons from 30 years of Syrian state discourse. *Political Science Research and Methods*, *9*(4), 693–708. https://doi.org/10.1017/psrm.2020.28

Amador, J., Collignon-Delmar, S., Benoit, K., & Matsuo, A. (2017). Predicting the Brexit vote by tracking and classifying public opinion using Twitter data. *Statistics, Politics and Policy*, *8*(1), 85–104. https://doi.org/10.1515/spp-2017-0006

Amidi, A. & Amidi, S. (2019). *Recurrent neural networks cheatsheet* [Lecture Notes]. Stanford University. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Amsalem, E., Fogel-Dror, Y., Shenhav, S. R., & Sheafer, T. (2020). Fine-grained analysis of diversity levels in the news. *Communication Methods and Measures*, *14*(4), 266–284. https://doi.org/10.1080/19312458.2020.1825659

Anastasopoulos, L. J. & Bertelli, A. M. (2020). Understanding delegation through machine learning: A method and application to the European Union. *American Political Science Review*, *114*(1), 291–301. https://doi.org/10.1017/S0003055419000522

Aribandi, V., Tay, Y., Schuster, T., Rao, J., Zheng, H. S., Mehta, S. V., Zhuang, H.,

Tran, V. Q., Bahri, D., Ni, J., Gupta, J., Hui, K., Ruder, S., & Metzler, D. (2022). Ext5: Towards extreme multi-task scaling for transfer learning. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=Vzh1BFUCiIX

Aßenmacher, M. & Heumann, C. (2020). On the comparability of pre-trained language models. In S. Ebling, D. Tuggener, M. Hürlimann, M. Cieliebak, & M. Volk (Eds.), *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS 2020)*. CEUR-WS.org. http://ceur-ws.org/Vol-2624/paper2.pdf

Aßenmacher, M., Schulze, P., & Heumann, C. (2021). Benchmarking down-scaled (not so large) pre-trained language models. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 14–27). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.2

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., & Auli, M. (2021). XLS-R: Self-supervised cross-lingual speech representation learning at scale. arXiv. https://arxiv.org/abs/2111.09296

Baden, C., Kligler-Vilenchik, N., & Yarchi, M. (2020). Hybrid content analysis: Toward a strategy for the theory-driven, computer-assisted classification of large text corpora. *Communication Methods and Measures*, *14*(3), 165–183. https://doi.org/10.1080/19312458.2020.1803247

Baerg, N. & Lowe, W. (2020). A textual Taylor rule: Estimating central bank preferences combining topic and scaling methods. *Political Science Research and Methods*, *8*(1), 106–122. https://doi.org/10.1017/psrm.2018.31

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 12449–12460). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf

Bagozzi, B. E. & Berliner, D. (2018). The politics of scrutiny in human rights monitoring: Evidence from Structural Topic Models of US State Department human rights reports. *Political Science Research and Methods*, *6*(4), 661–677. https://doi.org/10.1017/psrm.2016.44

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. https://arxiv.org/abs/1409.0473

Banaji, M. R. & Heiphetz, L. (2010). Attitudes. In S. T. Fiske, D. T. Gilbert, & G.

Lindzey (Eds.), *Handbook of Social Psychology* (pp. 348–388). John Wiley & Sons. https://doi.org/10.1002/9780470561119

Bapna, A., Chung, Y.-a., Wu, N., Gulati, A., Jia, Y., Clark, J. H., Johnson, M., Riesa, J., Conneau, A., & Zhang, Y. (2021). SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training. arXiv. https://arxiv.org/abs/2110.10329

Barberá, P. (2015). Birds of the same feather tweet together: Bayesian ideal point estimation using Twitter data. *Political Analysis*, *23*(1), 76–91. https://doi.org/10.1093/pan/mpu011

Barberá, P., Boydstun, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated text classification of news articles: A practical guide. *Political Analysis*, *29*(1), 19–42. https://doi.org/10.1017/pan.2020.8

Barberá, P., Casas, A., Nagler, J., Egan, P. J., Bonneau, R., Jost, J. T., & Tucker, J. A. (2019). Who leads? Who follows? Measuring issue attention and agenda setting by legislators and the mass public using social media data. *American Political Science Review*, *113*(4), 883–901. https://doi.org/10.1017/S0003055419000352

Barberá, P., Jost, J. T., Nagler, J., Tucker, J. A., & Bonneau, R. (2015a). Tweeting from left to right: Is online political communication more than an echo chamber? *Psychological Science*, *26*(10), 1531–1542. https://doi.org/10.1177/0956797615594620

Barberá, P. & Steinert-Threlkeld, Z. C. (2020). How to use social media data for political science research. In L. Curini & R. Franzese (Eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations* (pp. 404–423). SAGE Publications. https://www.doi.org/10.4135/9781526486387.n26

Barberá, P., Wang, N., Bonneau, R., Jost, J. T., Nagler, J., Tucker, J., & González-Bailón, S. (2015b). The critical periphery in the growth of social protests. *PLoS ONE*, *10*(11), e0143611. https://doi.org/10.1371/journal.pone.0143611

Beauchamp, N. (2017). Predicting and interpolating state-level polls using Twitter textual data. *American Journal of Political Science*, *61*(2), 490–503. https://doi.org/10.1111/ajps.12274

Belinkov, Y. & Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7, 49–72. https://doi.org/10.1162/tacl_a_00254

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document Transformer. arXiv. https://arxiv.org/abs/2004.05150

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155. https://www.jmlr.org/papers/v3/bengio03a.html

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Benoit, K. (2020). Text as data: An overview. In L. Curini & R. Franzese (Eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations* (pp. 461–497). SAGE Publications. https://www.doi.org/10.4135/9781526486387.n29

Benoit, K. & Laver, M. (2006). *Party policy in modern democracies*. Routledge.

Benoit, K., Laver, M., & Mikhaylov, S. (2009). Treating words as data with error: Uncertainty in text statements of policy positions. *American Journal of Political Science*, *53*(2), 495–513. https://doi.org/10.1111/j.1540-5907.2009.00383.x

Bischl, B., Mersmann, O., Trautmann, H., & Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, *20*(2), 249–275. https://doi.org/10.1162/EVCO_a_00069

Bischl, B. & Molnar, C. (2018). *Introduction to machine learning* [Lecture Notes]. Ludwig-Maximilians-Universität München.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blaydes, L., Grimmer, J., & McQueen, A. (2018). Mirrors for princes and sultans: Advice on the art of governance in the medieval christian and islamic worlds. *The Journal of Politics*, *80*(4), 1150–1167. https://doi.org/10.1086/699246

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1145/2133806.2133826

Blei, D. M. & Lafferty, J. D. (2007). A Correlated Topic Model of science. *The Annals of Applied Statistics*, *1*(1), 17–35. https://doi.org/10.1214/07-AOAS114

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. https://doi.org/10.1162/tacl_a_00051

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., & et al. (2021). On the opportunities and risks of foundation models. arXiv. https://arxiv.org/abs/2108.07258

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 632–642). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1075

Bräuninger, T., Müller, J., & Stecker, C. (2016). Modeling preferences using roll call votes in parliamentary systems. *Political Analysis*, *24*(2), 189–210. https://doi.org/10.1093/pan/mpw006

Breiman, L. (2001a). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L. (2001b). Statistical modeling: The two cultures. *Statistical Science*, *16*(3), 199–215.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. arXiv. https://arxiv.org/abs/2005.14165

Brownlee, J. (2017). What is the difference between a parameter and a hyperparameter? *Machine Learning Mastery*. https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/

Brownlee, J. (2019). A gentle introduction to cross-entropy for machine learning. *Machine Learning Mastery*. https://machinelearningmastery.com/cross-entropy-for-machine-learning/

Brownlee, J. (2020a). Difference between algorithm and model in machine learning. *Machine Learning Mastery*. https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/

Brownlee, J. (2020b). Random oversampling and undersampling for imbalanced classification. *Machine Learning Mastery*. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Bussell, J. (2020). Shadowing as a tool for studying political elites. *Political Analysis*, *28*(4), 469–486. https://doi.org/10.1017/pan.2020.14

Busso, C., Bulut, M., Lee, C.-C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J. N., Lee, S., & Narayanan, S. S. (2008). IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources & Evaluation*, 42, 335. https://doi.org/10.1007/s10579-008-9076-6

Bustikova, L., Siroky, D. S., Alashri, S., & Alzahrani, S. (2020). Predicting partisan

responsiveness: A probabilistic text mining time-series approach. *Political Analysis*, *28*(1), 47–64. https://doi.org/10.1017/pan.2019.18

Cabrio, E. & Villata, S. (Eds.) (2020). *Proceedings of the 7th Workshop on Argument Mining*. Association for Computational Linguistics. https://aclanthology.org/2020.argmining-1.0

Cacioppo, J. T. & Berntson, G. G. (1994). Relationship between attitudes and evaluative space: A critical review, with emphasis on the separability of positive and negative substrates. *Psychological Bulletin*, *115*(3), 401–423. https://doi.org/10.1037/0033-2909.115.3.401

Cacioppo, J. T., Gardner, W. L., & Berntson, G. G. (1997). Beyond bipolar conceptualizations and measures: The case of attitudes and evaluative space. *Personality and Social Psychology Review*, *1*(1), 3–25. https://doi.org/10.1207/s15327957pspr0101_2

Camacho-Collados, J. & Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, *63*(1), 743–788. https://doi.org/10.1613/jair.1.11259

Card, D., Tan, C., & Smith, N. A. (2018). Neural models for documents with metadata. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2031–2040). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1189

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In P. E. Utgoff (Ed.), *Proceedings of the Tenth International Conference on Machine Learning* (pp. 41–48). Morgan Kaufmann.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28, 41–75. https://doi.org/10.1023/A:1007379606734

Ceron, A., Curini, L., & Iacus, S. M. (2015). Using sentiment analysis to monitor electoral campaigns: Method matters—Evidence from the United States and Italy. *Social Science Computer Review*, *33*(1), 3–20. https://doi.org/10.1177/0894439314521983

Ceron, A., Curini, L., Iacus, S. M., & Porro, G. (2014). Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. *New Media & Society*, *16*(2), 340–358. https://doi.org/10.1177/1461444813480466

Chang, C. & Masterson, M. (2020). Using word order in political text classification with long short-term memory models. *Political Analysis*, *28*(3), 395–411. https://doi.org/10.1017/pan.2019.46

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems*

22. Curran Associates Inc. https://proceedings.neurips.cc/paper/2009/file/f92586a25bb3145facd64ab20fd554ff-Paper.pdf

Cheang, B., Wei, B., Kogan, D., Qiu, H., & Ahmed, M. (2020). Language representation models for fine-grained sentiment classification. arXiv. https://arxiv.org/abs/2005.13619

Chen, D. & Manning, C. (2014). A fast and accurate dependency parser using neural networks. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 740–750). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1082

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1179

Chollet, F. (2021). *Deep learning with Python (2nd ed.)*. Manning Publications. https://www.manning.com/books/deep-learning-with-python-second-edition

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

Chomsky, N. (2017). The Galilean challenge: Architecture and evolution of language. *Journal of Physics: Conference Series*, 880, 012015. https://doi.org/10.1088/1742-6596/880/1/012015

Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10, 73–91. https://doi.org/10.1162/tacl_a_00448

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. In T. Linzen, G. Chrupała, Y. Belinkov, & D. Hupkes (Eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (pp. 276–286). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4828

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? Try ARC, the AI2 reasoning challenge. arXiv. https://arxiv.org/abs/1803.05457

Clark, T. S. & Lauderdale, B. (2010). Locating Supreme Court opinions in doctrine space. *American Journal of Political Science*, *54*(4), 871–890. https://doi.org/10.2307/20788775

Clinton, J., Jackman, S., & Rivers, D. (2004). The statistical analy-

sis of roll call data. *American Political Science Review*, *98*(2), 355–370. https://doi.org/10.1017/S0003055404001194

Collobert, R. & Weston, J. (2007). Fast semantic extraction using a novel neural network architecture. In A. Zaenen & A. van den Bosch (Eds.), *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* (pp. 560–567). Association for Computational Linguistics. https://aclanthology.org/P07-1071

Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 160–167). Association for Computing Machinery. https://doi.org/10.1145/1390156.1390177

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*(76), 2493–2537. http://jmlr.org/papers/v12/collobert11a.html

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440–8451). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747

Connor, R. J. & Mosimann, J. E. (1969). Concepts of independence for proportions with a generalization of the Dirichlet distribution. *Journal of the American Statistical Association*, *64*(325), 194–206.

Csereklyei, Z., Thurner, P. W., Langer, J., & Küchenhoff, H. (2017). Energy paths in the European Union: A model-based clustering approach. *Energy Economics*, 65, 442–457. https://doi.org/10.1016/j.eneco.2017.05.014

Däubler, T. & Benoit, K. (2017). Estimating better left-right positions through statistical scaling of manual content analysis. Manuscript. https://kenbenoit.net/pdfs/text_in_context_2017.pdf

Däubler, T. & Benoit, K. (2021). Scaling hand-coded political texts to learn more about left-right policy content. *Party Politics*, (pp. 1–11). https://doi.org/10.1177/13540688211026076

Denny, M. J. & Spirling, A. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, *26*(2), 168–189. https://doi.org/10.1017/pan.2017.44

Deshwal, A., Doppa, J. R., & Roth, D. (2019). Learning and inference for structured prediction: A unifying perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (pp.

6291–6299). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2019/878

Devlin, J. (2019). Multilingual BERT. *github.com/google-research*. https://github.com/google-research/bert/blob/master/multilingual.md

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Di Cocco, J. & Monechi, B. (2021). How populist are parties? Measuring degrees of populism in party manifestos using supervised machine learning. *Political Analysis*, (pp. 1–17). https://doi.org/10.1017/pan.2021.29

Diekmann, A. (2007). *Empirische Sozialforschung (11th ed.)*. Rowohlt.

Dietrich, B. J., Hayes, M., & O'Brien, D. Z. (2019). Pitch perfect: Vocal pitch and the emotional intensity of congressional speech. *American Political Science Review*, *113*(4), 941–962. https://doi.org/10.1017/S0003055419000467

D'Orazio, V., Landis, S. T., Palmer, G., & Schrodt, P. (2014). Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Political Analysis*, *22*(2), 224–242. https://doi.org/10.1093/pan/mpt030

Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv. https://arxiv.org/abs/1702.08608

Druckman, J. N., Kifer, M. J., & Parkin, M. (2009). Campaign communications in U.S. congressional elections. *American Political Science Review*, *103*(3), 343–366. https://doi.org/10.1017/S0003055409990037

Druckman, J. N. & Lupia, A. (2000). Preference formation. *Annual Review of Political Science*, 3, 1–24. https://doi.org/10.1146/annurev.polisci.3.1.1

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159. https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

Duong, L., Cohn, T., Bird, S., & Cook, P. (2015). Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 845–850). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-2139

Eagly, A. H. & Chaiken, S. (1993). *The psychology of attitudes*. Harcourt Brace Jovanovich College Publishers.

Eagly, A. H. & Chaiken, S. (2007). The advantages of an inclusive definition of attitude. *Social Cognition*, *25*(5), 582–602. https://doi.org/10.1521/soco.2007.25.5.582

Efron, B. & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall.

Egami, N., Fong, C. J., Grimmer, J., Roberts, M. E., & Stewart, B. M. (2018). How to make causal inferences using texts. arXiv. https://arxiv.org/abs/1802.02163

Eisenstein, J., O'Connor, B., Smith, N. A., & Xing, E. P. (2010). A latent variable model for geographic lexical variation. In H. Li & L. Màrquez (Eds.), *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1277–1287). Association for Computational Linguistics. https://aclanthology.org/D10-1124

Elff, M. (2013). A dynamic state-space model of coded political texts. *Political Analysis*, *21*(2), 217–232. https://doi.org/10.1093/pan/mps042

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1016/0364-0213(90)90002-E

Enamorado, T., López-Moctezuma, G., & Ratkovic, M. (2021). Scaling data from multiple sources. *Political Analysis*, *29*(2), 212–235. https://doi.org/10.1017/pan.2020.24

Ennser-Jedenastik, L. & Meyer, T. M. (2018). The impact of party cues on manual coding of political texts. *Political Science Research and Methods*, *6*(3), 625–633. https://doi.org/10.1017/psrm.2017.29

Erlich, A., Dantas, S. G., Bagozzi, B. E., Berliner, D., & Palmer-Rubin, B. (2021). Multi-label prediction for political text-as-data. *Political Analysis*, (pp. 1–18). https://doi.org/10.1017/pan.2021.15

Eshima, S., Imai, K., & Sasaki, T. (2021). Keyword assisted topic models. arXiv. https://arxiv.org/abs/2004.05964v2

Fabringar, L. R., MacDonald, T. K., & Wegener, D. T. (2019). The origins and structure of attitudes. In D. Albarracin & B. T. Johnson (Eds.), *The Handbook of Attitudes* (pp. 109–157). Routledge. https://doi.org/10.4324/9781315178103

Firat, A. (2016). Unified framework for quantification. arXiv. https://arxiv.org/abs/1606.00868

Firth, J. R. (1957). *Studies in linguistic analysis*. Publications of the Philological Society. Blackwell.

Fong, C. & Tyler, M. (2021). Machine learning predictions as regression covariates. *Political Analysis*, *29*(4), 467–484. https://doi.org/10.1017/pan.2020.38

Fowler, E. F., Franz, M. M., Martin, G. J., Peskowitz, Z., & Ridout, T. N. (2021). Political advertising online and offline. *American Political Science Review*, *115*(1), 130–149. https://doi.org/10.1017/S0003055420000696

Fu, T.-J., Li, L., Gan, Z., Lin, K., Wang, W. Y., Wang, L., & Liu, Z. (2021). VIOLET: End-to-end video-language Transformers with masked visual-token modeling. arXiv. https://arxiv.org/abs/2111.12681

Gabel, M. J. & Huber, J. D. (2000). Putting parties in their place: Inferring party left-right ideological positions from party manifestos data. *American Journal of Political Science*, *44*(1), 94–103. https://doi.org/10.2307/2669295

Galassi, A., Lippi, M., & Torroni, P. (2021). Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(10), 4291–4308. https://doi.org/10.1109/TNNLS.2020.3019893

Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 3816–3830). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.295

Gatt, A. & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61, 65–170. https://doi.org/10.1613/jair.5477

Gawronski, B. & Brannon, S. M. (2019). Attitudes and the implicit-explicit dualism. In D. Albarracin & B. T. Johnson (Eds.), *The Handbook of Attitudes* (pp. 158–196). Routledge. https://doi.org/10.4324/9781315178103

Ghahramani, Z. (2015). *The Machine Learning Summer School. Lecture: Bayesian inference* [Lecture Notes]. Max Planck Institute for Intelligent Systems. http://mlss.tuebingen.mpg.de/2015/slides/ghahramani/gp-neural-nets15.pdf

Glavaš, G., Nanni, F., & Ponzetto, S. P. (2017). Cross-lingual classification of topics in political texts. In D. Hovy, S. Volkova, D. Bamman, D. Jurgens, B. O'Connor, O. Tsur, & A. S. Doğruöz (Eds.), *Proceedings of the Second Workshop on NLP and Computational Social Science* (pp. 42–46). Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-2906

Glavaš, G., Nanni, F., & Ponzetto, S. P. (2019). Computational analysis of political texts: Bridging research efforts across communities. In P. Nakov & A. Palmer (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts* (pp. 18–23). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-4004

Göbel, S. & Munzert, S. (2018). Political advertising on the Wikipedia marketplace of information. *Social Science Computer Review*, *36*(2), 157–175. https://doi.org/10.1177/0894439317703579

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*(1), 345–420. https://jair.org/index.php/jair/article/download/11030/26198/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., & Parikh, D. (2017). Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, (pp. 6904–6913). https://openaccess.thecvf.com/content_cvpr_2017/papers/Goyal_Making_the_v_CVPR_2017_paper.pdf

Graves, A. (2008). *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technische Universität München. https://mediatum.ub.tum.de/doc/1289309/401810.pdf

Graves, A. (2014). Generating sequences with recurrent neural networks. arXiv. https://arxiv.org/abs/1308.0850

Griffiths, T. L. & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(suppl 1), 5228–5235. https://doi.org/10.1073/pnas.0307752101

Grimmer, J. (2010). A Bayesian hierarchical topic model for political texts: Measuring expressed agendas in Senate press releases. *Political Analysis*, *18*(1), 1–35. https://doi.org/10.1093/pan/mpp034

Grimmer, J. & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Grosse, R. (2020a). *CSC321 Neural networks and machine learning (UTM). Lecture 3: Linear classification* [Lecture Notes]. University of Toronto. https://www.cs.toronto.edu/~lczhang/321/notes/notes03.pdf

Grosse, R. (2020b). *CSC321 Neural networks and machine learning (UTM). Lecture 4: Training a classifier* [Lecture Notes]. University of Toronto. https://www.cs.toronto.edu/~lczhang/321/notes/notes04.pdf

Grosse, R. (2020c). *CSC321 Neural networks and machine learning (UTM). Lecture 9: Generalization* [Lecture Notes]. University of Toronto. https://www.cs.toronto.edu/~lczhang/321/notes/notes09.pdf

Habernal, I., Wachsmuth, H., Gurevych, I., & Stein, B. (2018). The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1930–1940). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1175

Haim, M. & Hoven, E. (2022). Hate speech's double damage: A semi-automated approach toward direct and indirect targets. *Journal of Quantitative Description: Digital Media*, 2. https://doi.org/10.51685/jqd.2022.009

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. SAGE Publications.

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques (3rd ed.)*. Elsevier.

Hansen, C. (2019). Activation functions explained - GELU, SELU, ELU, ReLU and more. *Machine Learning From Scratch*. https://mlfromscratch.com/activation-functions-explained/

Hare, C. & Poole, K. T. (2014). The polarization of contemporary American politics. *Polity*, *46*(3), 411–429. https://doi.org/10.1057/pol.2014.10

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction (2nd ed.)*. Springer. https://doi.org/10.1007/978-0-387-84858-7

Hayes, A. F. & Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, *1*(1), 77–89. https://doi.org/10.1080/19312450709336664

Henderson, M., Budzianowski, P., Casanueva, I., Coope, S., Gerz, D., Kumar, G., Mrkšić, N., Spithourakis, G., Su, P.-H., Vulić, I., & Wen, T.-H. (2019). A repository of conversational datasets. In Y.-N. Chen, T. Bedrax-Weiss, D. Hakkani-Tur, A. Kumar, M. Lewis, T.-M. Luong, P.-H. Su, & T.-H. Wen (Eds.), *Proceedings of the First Workshop on NLP for Conversational AI* (pp. 1–10). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4101

Henderson, M., Thomson, B., & Williams, J. D. (2014). The second dialog state tracking challenge. In K. Georgila, M. Stone, H. Hastie, & A. Nenkova (Eds.), *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)* (pp. 263–272). Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-4337

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020). Measuring massive multitask language understanding. arXiv. https://arxiv.org/abs/2009.03300

Hendrycks, D. & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). arXiv. https://arxiv.org/abs/1606.08415

Herrmann, M. & Döring, H. (2021). Party positions from Wikipedia classifications of party ideology. *Political Analysis*, (pp. 1–20). https://doi.org/10.1017/pan.2021.28

Herzog, A. & Benoit, K. (2015). The most unkindest cuts: Speaker selection and expressed government dissent during economic crisis. *The Journal of Politics*, *77*(4), 1157–1175. https://doi.org/10.1086/682670

Heumann, C., Schomaker, M., & Shalabh (2016). *Introduction to statistics and data analysis*. Springer. https://doi.org/10.1007/978-3-319-46162-5

Hewitt, J. & Manning, C. D. (2019). A structural probe for finding syntax in word representations. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4129–4138). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1419

Hinton, G., Srivastava, N., & Swerky, K. (2012). *Neural networks for machine learning. Lecture 6a: Overview of mini-batch gradient descent* [Lecture Notes]. Coursera. http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf

Hix, S., Noury, A., & Roland, G. (2006). Dimensions of politics in the European Parliament. *American Journal of Political Science*, *50*(2), 494–511. https://doi.org/10.1111/j.1540-5907.2006.00198.x

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, *81*(396), 945–960. https://doi.org/10.2307/2289064

Hopkins, D. J. & King, G. (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, *54*(1), 229–247. https://doi.org/10.1111/j.1540-5907.2009.00428.x

Howard, J. & Ruder, S. (2018). Universal language model fine-tuning for text classification. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (pp. 328–339). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1031

Hu, M., Peng, Y., Huang, Z., Li, D., & Lv, Y. (2019). Open-domain targeted sentiment analysis via span-based extraction and classification. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 537–546). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1051

Huang, J., Chang, Y., & Cheng, X. (Eds.) (2020a). *SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery. https://doi.org/10.1145/3397271

Huang, L., Perry, P. O., & Spirling, A. (2020b). A general model of author "style" with application to the UK House of Commons, 1935–2018. *Political Analysis*, *28*(3), 412–434. https://doi.org/10.1017/pan.2019.49

Hutchins, W. J. (2004). The Georgetown-IBM experiment demonstrated in January 1954. In R. E. Frederking & K. B. Taylor (Eds.), *Machine Translation: From Real Users to Research* (pp. 102–114). Springer. https://doi.org/10.1007/978-3-540-30194-3_12

İrsoy, O. & Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 720–728). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1080

Iyyer, M., Enns, P., Boyd-Graber, J., & Resnik, P. (2014). Political ideology detection using recursive neural networks. In K. Toutanova & H. Wu (Eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1113–1122). Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1105

Jackman, S. (2008). Measurement. In J. M. Box-Steffensmeier, H. E. Brady, & D. Collier (Eds.), *The Oxford Handbook of Political Methodology* (pp. 119–151). Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199286546.001.0001

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., Henaff, O. J., Botvinick, M., Zisserman, A., Vinyals, O., & Carreira, J. (2022). Perceiver IO: A general architecture for structured inputs & outputs. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=fILj7WpI-g

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., & Carreira, J. (2021). Perceiver: General perception with iterative attention. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 4651–4664). PMLR. https://proceedings.mlr.press/v139/jaegle21a.html

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning with applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7

Jawahar, G., Sagot, B., & Seddah, D. (2019). What does BERT learn about the structure of language? In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3651–3657). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1356

Jerzak, C. T., King, G., & Strezhnev, A. (2022). An improved method of auto-

mated nonparametric content analysis for social science. *Political Analysis*, (pp. 1–17). https://doi.org/10.1017/pan.2021.36

Jiang, J. (2008). A literature survey on domain adaptation of statistical classifiers. Manuscript. http://www.mysmu.edu/faculty/jingjiang/papers/da_survey.pdf

Jin, W. & Ho, H. H. (2009). A novel lexicalized HMM-based learning framework for web opinion mining. In A. Danyluk (Ed.), *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 465–472). Association for Computing Machinery. https://doi.org/10.1145/1553374.1553435

Johnson, J. (2017). *CS231n: Convolutional neural networks for visual recognition. Derivatives, backpropagation, and vectorization* [Lecture Notes]. http://cs231n.stanford.edu/handouts/derivatives.pdf

Joshi, P., Santy, S., Budhiraja, A., Bali, K., & Choudhury, M. (2020). The state and fate of linguistic diversity and inclusion in the NLP world. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 6282–6293). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.560

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., & Bennamoun, M. (2022). Hands-on Bayesian neural networks – A tutorial for deep learning users. arXiv. https://arxiv.org/abs/2007.06823

Jungherr, A., Posegga, O., & An, J. (2022). Populist supporters on reddit: A comparison of content and behavioral patterns within publics of supporters of Donald Trump and Hillary Clinton. *Social Science Computer Review*, *40*(3), 809–830. https://doi.org/10.1177/0894439321996130

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In K. Toutanova & H. Wu (Eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 655–665). Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1062

Katagiri, A. & Min, E. (2019). The credibility of public and private signals: A document-based approach. *American Political Science Review*, *113*(1), 156–172. https://doi.org/10.1017/S0003055418000643

Kaufman, A. R. & Klevs, A. (2021). Adaptive fuzzy string matching: How to merge datasets with only one (messy) identifying field. *Political Analysis*, (pp. 1–7). https://doi.org/10.1017/pan.2021.38

Kavlakoglu, E. (2020). AI vs. machine learning vs. deep learning vs. neural networks: What's the difference? *IBM Cloud Blog*. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Kellstedt, P. M. & Whitten, G. D. (2009). *The fundamentals of political science research*. Cambridge University Press. https://doi.org/10.1017/9781108131704

Kim, I. S. (2017). Political cleavages within industry: Firm-level lobbying for trade liberalization. *American Political Science Review*, *111*(1), 1–20. https://doi.org/10.1017/S0003055416000654

Kim, I. S. & Kunisky, D. (2021). Mapping political communities: A statistical analysis of lobbying networks in legislative politics. *Political Analysis*, *29*(3), 317–336. https://doi.org/10.1017/pan.2020.29

Kim, J., Griggs, E., Kim, I. S., & Oh, A. (2021). Learning bill similarity with annotated and augmented corpora of bills. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 10048–10064). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.787

Kim, Y. (2014). Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181

King, G. (1998). *Unifying political methodology: The likelihood theory of statistical inference*. The University of Michigan Press. https://doi.org/10.3998/mpub.23784

King, G., Lam, P., & Roberts, M. E. (2017). Computer-assisted keyword and document set discovery from unstructured text. *American Journal of Political Science*, *61*(4), 971–988. https://doi.org/10.1111/ajps.12291

King, G., Pan, J., & Roberts, M. E. (2013). How censorship in China allows government criticism but silences collective expression. *American Political Science Review*, *107*(2), 326–343. https://doi.org/10.1017/S0003055413000014

King, K., Keohane, R. O., & Verba, S. (1994). *Designing social inquiry: Scientific inference in qualitative research*. Princeton University Press.

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. http://arxiv.org/abs/1412.6980

Kingma, D. P. & Welling, M. (2014). Auto-encoding variational bayes. In Y. Bengio & Y. LeCun (Eds.), *International Conference on Learning Representations (ICLR 2014)*. https://arxiv.org/abs/1312.6114

Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Y. Bengio & Y. LeCun (Eds.), *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

Klüver, H. (2009). Measuring interest group influence using quantitative text analysis. *European Union Politics*, *10*(4), 535–549. https://doi.org/10.1177/1465116509346782

Klüver, H. & Bäck, H. (2019). Coalition agreements, issue attention, and cabinet governance. *Comparative Political Studies*, *52*(13–14), 1995–2031. https://doi.org/10.1177/0010414019830726

Knight, K., Badarau, B., Baranescu, L., Bonial, C., Bardocz, M., Griffitt, K., Hermjakob, U., Marcu, D., Palmer, M., O'Gorman, T., & Schneider, N. (2020). *Abstract meaning representation (AMR) Annotation release 3.0*. [Data set]. Linguistic Data Consortium. https://doi.org/10.35111/44cy-bp51

Kobayashi, G., Kuribayashi, T., Yokoi, S., & Inui, K. (2020). Attention is not only a weight: Analyzing Transformers with vector norms. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7057–7075). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.574

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., & Grefenstette, E. (2018). The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6, 317–328. https://doi.org/10.1162/tacl_a_00023

Koh, A. & Boey, D. K. S. (2021). Predicting policy domains and preferences with BERT and convolutional neural networks. Manuscript. https://hertie-data-science-lab.github.io/student-projects/posts/predicting-policy-domains-and-preferences-with-bert-and-convolutional-neural-networks/

Koo, T., Carreras, X., & Collins, M. (2008). Simple semi-supervised dependency parsing. In J. D. Moore, S. Teufel, J. Allan, & S. Furui (Eds.), *Proceedings of the Annual Meeting of the Association for Computational Linguistics 2008: Human Language Technologies* (pp. 595–603). Association for Computational Linguistics. https://aclanthology.org/P08-1068

Kouw, W. M. & Loog, M. (2019). A review of domain adaptation without target labels. arXiv. https://arxiv.org/abs/1901.05335

Kozareva, Z., Ravi, S., Vlachos, A., Agrawal, P., & Martins, A. (Eds.) (2021). *Proceedings of the 5th Workshop on Structured Prediction for NLP (SPNLP 2021)*. Association for Computational Linguistics. https://aclanthology.org/2021.spnlp-1.0

Krippendorff, K. (2013). *Content analysis: An introduction to its methodology (3rd ed.)*. SAGE Publications.

Krippendorff, K. & Craggs, R. (2016). The reliability of multi-valued coding of data. *Communication Methods and Measures*, *10*(4), 181–198. https://doi.org/10.1080/19312458.2016.1228863

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with

deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (pp. 1097–1105). Curran Associates Inc. https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Küçük, D. & Can, F. (2020). Stance detection: A survey. *ACM Computing Surveys*, *53*(1), 1–37. https://doi.org/10.1145/3369026

Kuhn, M. & Johnson, K. (2013). *Applied predictive modeling*. Springer. https://doi.org/10.1007/978-1-4614-6849-3

Kumar, E. (2011). *Natural language processing*. I. K. International Publishing House Pvt. Ltd.

Kumar, S., Jat, S., Saxena, K., & Talukdar, P. (2019). Zero-shot word sense disambiguation using sense definition embeddings. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 5670–5681). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1568

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C. E. Brodley & A. Danyluk (Eds.), *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 282–289). Morgan Kaufmann Publishers Inc. https://doi.org/10.5555/645530.655813

Lai, G., Xie, Q., Liu, H., Yang, Y., & Hovy, E. (2017). RACE: Large-scale reading comprehension dataset from examinations. arXiv. https://arxiv.org/abs/1704.04683

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In K. Knight, A. Nenkova, & O. Rambow (Eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 260–270). Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1030

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In A. Rush (Ed.), *8th International Conference on Learning Representations (ICLR 2020)*. https://openreview.net/pdf?id=H1eA7AEtvS

Lauderdale, B. E. & Herzog, A. (2016). Measuring political positions from legislative speech. *Political Analysis*, *24*(3), 374–394. https://doi.org/10.1093/pan/mpw017

Laver, M., Benoit, K., & Garry, J. (2003). Extracting policy positions from political texts using words as data. *American Political Science Review*, *97*(2), 311–331. https://doi.org/10.1017/S0003055403000698

Lawrence, J. & Reed, C. (2020). Argument mining: A survey. *Computational Linguistics*, *45*(4), 765–818. https://doi.org/10.1162/coli_a_00364

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. *Proceedings of Machine Learning Research*, *32*(2), 1188–1196. http://proceedings.mlr.press/v32/le14.html

LeCun, Y. & Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*. MIT Press.

Lei, T. (2021). When attention meets fast recurrence: Training language models with reduced compute. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7633–7648). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.602

Leitgeb, H. (2017). *The stability of belief. How rational belief coheres with probability*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198732631.001.0001

Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The Winograd schema challenge. In S. A. M. Gerhard Brewka, Thomas Eiter (Ed.), *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning* (pp. 552–561). AAAI Press. https://dl.acm.org/doi/10.5555/3031843.3031909

Lewis, D. D. (1997). *Reuters-21578 (Distribution 1.0)*. [Data set]. http://www.daviddlewis.com/resources/testcollections/reuters21578/

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7871–7880). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Li, F.-F., Krishna, R., & Xu, D. (2020a). *CS231n: Convolutional neural networks for visual recognition. Neural networks I* [Lecture Notes]. Stanford University. https://cs231n.github.io/neural-networks-1/

Li, F.-F., Krishna, R., & Xu, D. (2020b). *CS231n: Convolutional neural networks for visual recognition. Optimization I* [Lecture Notes]. Stanford University. https://cs231n.github.io/optimization-1/

Li, F.-F., Krishna, R., & Xu, D. (2020c). *CS231n: Convolutional neural networks for visual recognition. Optimization II* [Lecture Notes]. Stanford University. https://cs231n.github.io/optimization-2/

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., & Gonzalez, J. (2020d).

Train big, then compress: Rethinking model size for efficient training and inference of Transformers. *Proceedings of Machine Learning Research*, 119, 5958–5968. http://proceedings.mlr.press/v119/li20m.html

Little, R. J. A. & Rubin, D. B. (2002). *Statistical analysis with missing data (2nd ed.)*. Wiley. https://doi.org/10.1002/9781119013563

Liu, A. T., wen Yang, S., Chi, P.-H., chun Hsu, P., & yi Lee, H. (2020). Mockingjay: Unsupervised speech representation learning with deep bidirectional Transformer encoders. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*. IEEE. https://doi.org/10.1109/icassp40776.2020.9054458

Liu, B. (2015). *Sentiment analysis*. Cambridge University Press. https://doi.org/10.1017/CBO9781139084789

Liu, H., Simonyan, K., & Yang, Y. (2019a). DARTS: Differentiable architecture search. In T. Sainath (Ed.), *7th International Conference on Learning Representations, ICLR 2019*. https://openreview.net/forum?id=S1eYHoC5FX

Liu, J., Chen, J., & Ye, J. (2009). Large-scale sparse logistic regression. In J. Elder & F. S. Fogelman (Eds.), *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 547–556). Association for Computing Machinery. https://doi.org/10.1145/1557019.1557082

Liu, X. & Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 186–193). Association for Computing Machinery. https://doi.org/10.1145/1008992.1009026

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019b). RoBERTa: A robustly optimized BERT pretraining approach. arXiv. https://arxiv.org/abs/1907.11692

Lo, J., Proksch, S.-O., & Slapin, J. B. (2016). Ideological clarity in multiparty competition: A new measure and test using election manifestos. *British Journal of Political Science*, *46*(3), 591–610. https://doi.org/10.1017/S0007123414000192

Louis, A. (2020). A brief history of natural language processing – Part 1. *Medium.com*. https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbcb937ebce

Lowe, W. & Benoit, K. (2013). Validating estimates of latent traits from textual data using human judgment as a benchmark. *Political Analysis*, *21*(3), 298–313. https://doi.org/10.1093/pan/mpt002

Lucas, C., Nielsen, R. A., Roberts, M. E., Stewart, B. M., Storer, A., & Tingley, D. (2015). Computer-assisted text analysis for comparative politics. *Political Analysis*, *23*(2), 254–277. https://doi.org/10.1093/pan/mpu019

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1412–1421). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1166

MacCartney, B. (2014). *Understanding natural language understanding* [Presentation]. ACM SIGAI Bay Area Chapter Inaugural Meeting. https://nlp.stanford.edu/~wcmac/papers/20140716-UNLU.pdf

Magyar, Z. B. (2022). What makes party systems different? A principal component analysis of 17 advanced democracies 1970–2013. *Political Analysis*, *30*(2), 250–268. https://doi.org/10.1017/pan.2021.21

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press. https://nlp.stanford.edu/fsnlp/

Martin, G. J. & McCrain, J. (2019). Local news and national politics. *American Political Science Review*, *113*(2), 372–384. https://doi.org/10.1017/S0003055418000965

McCallum, A. & Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (pp. 188–191). https://aclanthology.org/W03-0430

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2018). Learned in translation: Contextualized word vectors. arXiv. https://arxiv.org/abs/1708.00107

Meidinger, M. & Aßenmacher, M. (2021). A new benchmark for NLP in social sciences: Evaluating the usefulness of pre-trained language models for classifying open-ended survey responses. In A. P. Rocha, L. Steels, & H. J. van den Herik (Eds.), *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021* (pp. 866–873). SciTePress. https://doi.org/10.5220/0010255108660873

Miao, Y., Yu, L., & Blunsom, P. (2016). Neural variational inference for text processing. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on International Conference on Machine Learning* (pp. 1727–1736). Journal of Machine Learning Research. https://dl.acm.org/doi/10.5555/3045390.3045573

Mikhaylov, S., Laver, M., & Benoit, K. R. (2012). Coder reliability and misclassification in the human coding of party manifestos. *Political Analysis*, *20*(1), 78–91. https://doi.org/10.1093/pan/mpr047

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv. https://arxiv.org/abs/1301.3781

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In T. Kobayashi, K. Hirose, & S. Nakamura (Eds.), *11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (pp. 1045–1048). International Speech Communication Association. https://doi.org/10.21437/Interspeech.2010-343

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3111–3119). Curran Associates Inc. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Mikolov, T., Yih, W.-t., & Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In L. Vanderwende, H. Daumé III, & K. Kirchhoff (Eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 746–751). Association for Computational Linguistics. https://www.aclweb.org/anthology/N13-1090

Miller, B., Linder, F., & Mebane, W. R. (2020). Active learning approaches for labeling text: Review and assessment of the performance of active learning approaches. *Political Analysis*, *28*(4), 532–551. https://doi.org/10.1017/pan.2020.4

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39–41. https://doi.org/10.1145/219717.219748

Mitchell, M., Aguilar, J., Wilson, T., & Van Durme, B. (2013). Open domain targeted sentiment. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1643–1654). Association for Computational Linguistics. https://aclanthology.org/D13-1171

Mitts, T. (2019). From isolation to radicalization: Anti-Muslim hostility and support for ISIS in the West. *American Political Science Review*, *113*(1), 173–194. https://doi.org/10.1017/S0003055418000618

Moens, M.-F., Boiy, E., Palau, R. M., & Reed, C. (2007). Automatic detection of arguments in legal texts. In A. Gardner & R. Winkels (Eds.), *Proceedings of the 11th International Conference on Artificial Intelligence and Law* (pp. 225–230). Association for Computing Machinery. https://doi.org/10.1145/1276318.1276362

Mohammad, S. M., Sobhani, P., & Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Transactions on Internet Technology*, *17*(3), 1–22. https://doi.org/10.1145/3003433

Molnar, C. (2014). *Statistical modeling: The two cultures* [Presentation]. Ludwig-

Maximilians-Universität München. https://github.com/christophM/presentation-two-cultures/blob/master/presentation.pdf

Molnar, C. (2022). *Interpretable machine learning*. https://christophm.github.io/interpretable-ml-book/

Montgomery, D. C. (2012). *Design and analysis of experiments (8th ed.)*. Wiley.

Moore, W. H. & Siegel, D. A. (2013). *A mathematics course for political and social research*. Princeton University Press.

Moosbrugger, H. (2012). Item-Response-Theorie (IRT). In H. Moosbrugger & A. Kelava (Eds.), *Testtheorie und Fragebogenkonstruktion (2nd ed.)*. Springer.

Mosbach, M., Andriushchenko, M., & Klakow, D. (2021). On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*. https://openreview.net/forum?id=nzpLWnVAyah

Mozer, R., Miratrix, L., Kaufman, A. R., & Anastasopoulos, J. L. (2020). Matching with text data: An experimental evaluation of methods for matching documents and of measuring match quality. *Political Analysis*, *28*(4), 445–468. https://doi.org/10.1017/pan.2020.1

Muchlinski, D., Yang, X., Birch, S., Macdonald, C., & Ounis, I. (2021). We need to go deeper: Measuring electoral violence using convolutional neural networks and social media. *Political Science Research and Methods*, *9*(1), 122–139. https://doi.org/10.1017/psrm.2020.32

Munzert, S., Rubba, C., Meißner, P., & Nyhuis, D. (2015). *Automated data collection with R*. John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118834732

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve Restricted Boltzmann machines. In J. Fürnkranz & T. Joachims (Eds.), *Proceedings of the 27th International Conference on International Conference on Machine Learning* (pp. 807–814). Omnipress. https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

Nakagawa, T., Inui, K., & Kurohashi, S. (2010). Dependency tree-based sentiment classification using CRFs with hidden variables. In R. Kaplan, J. Burstein, M. Harper, & G. Penn (Eds.), *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 786–794). Association for Computational Linguistics. https://aclanthology.org/N10-1120

Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., & Stoyanov, V. (2016). SemEval-2016 task 4: Sentiment analysis in Twitter. In S. Manandhar & D. Yuret (Eds.), *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 1–18). Association for Computational Linguistics. https://doi.org/10.18653/v1/S16-1001

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., & Wilson, T. (2013). SemEval-2013 task 2: Sentiment analysis in Twitter. In S. Manandhar & D. Yuret (Eds.), *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (pp. 312–320). Association for Computational Linguistics. https://aclanthology.org/S13-2052

Nallapati, R., Zhou, B., dos Santos, C., Gülçehre, Ç., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In S. Riezler & Y. Goldberg (Eds.), *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning* (pp. 280–290). Association for Computational Linguistics. https://aclanthology.org/K16-1028

Nanni, F., Zirn, C., Glavaš, G., Eichorst, J., & Ponzetto, S. P. (2016). TopFish: Topic-based analysis of political position in US electoral campaigns. In D. Širinić, J. Šnajder, Z. Fazekas, & S. Bevan (Eds.), *PolText 2016: The International Conference on the Advances in Computational Analysis of Political Text* (pp. 61–67). University of Zagreb. https://madoc.bib.uni-mannheim.de/41550/

Neelakantan, A., Shankar, J., Passos, A., & McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1059–1069). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1113

Nelson, L. K., Burk, D., Knudsen, M., & McCall, L. (2021). The future of coding: A comparison of hand-coding and three types of computer-assisted text analysis methods. *Sociological Methods & Research*, *50*(1), 202–237. https://doi.org/10.1177/0049124118769114

Nozza, D., Bianchi, F., & Hovy, D. (2020). What the [MASK]? Making sense of language-specific BERT models. arXiv. https://arxiv.org/abs/2003.02912

Olsson, F., Sahlgren, M., Ben Abdesslem, F., Ekgren, A., & Eck, K. (2020). Text categorization for conflict event annotation. In A. Hürriyetoğlu, E. Yörük, V. Zavarella, & H. Tanev (Eds.), *Proceedings of the Workshop on Automated Extraction of Socio-Political Events from News 2020* (pp. 19–25). European Language Resources Association (ELRA). https://aclanthology.org/2020.aespen-1.5

Osnabrügge, M., Ash, E., & Morelli, M. (2021). Cross-domain topic classification for political texts. *Political Analysis*, (pp. 1–22). https://doi.org/10.1017/pan.2021.37

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pang, B. & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In K. Knight (Ed.), *Proceedings of the*

43rd Annual Meeting of the Association for Computational Linguistics (pp. 115–124). Association for Computational Linguistics. https://doi.org/10.3115/1219840.1219855

Pang, B. & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2), 1–135. https://doi.org/10.1561/1500000011

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 79–86). Association for Computational Linguistics. https://doi.org/10.3115/1118693.1118704

Park, B., Greene, K., & Colaresi, M. (2020). Human rights are (increasingly) plural: Learning the changing taxonomy of human rights from large-scale text reveals information effects. *American Political Science Review*, *114*(3), 888–910. https://doi.org/10.1017/S0003055420000258

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). Towards the unification and robustness of perturbation and gradient based explanations. *Proceedings of Machine Learning Research*, *28*(3), 1310–1318. https://proceedings.mlr.press/v28/pascanu13.pdf

Patwa, P., Aguilar, G., Kar, S., Pandey, S., PYKL, S., Gambäck, B., Chakraborty, T., Solorio, T., & Das, A. (2020). SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, & E. Shutova (Eds.), *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 774–790). International Committee for Computational Linguistics. https://doi.org/10.18653/v1/2020.semeval-1.100

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Perry, P. O. & Benoit, K. (2017). Scaling text with the Class Affinity Model. arXiv. https://arxiv.org/abs/1710.08963

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018a). Deep contextualized word representations. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 2227–2237). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1202

Peters, M., Neumann, M., Zettlemoyer, L., & Yih, W.-t. (2018b). Dissecting contextual word embeddings: Architecture and representation. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1499–1509). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1179

Pilehvar, M. T. & Camacho-Collados, J. (2020). *Embeddings in natural language processing: Theory and advances in vector representations of meaning*. Morgan & Claypool Publishers. https://doi.org/10.2200/S01057ED1V01Y202009HLT047

Pilny, A., McAninch, K., Slone, A., & Moore, K. (2019). Using supervised machine learning in automated content analysis: An example using relational uncertainty. *Communication Methods and Measures*, *13*(4), 287–304. https://doi.org/10.1080/19312458.2019.1650166

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., & Eryiğit, G. (2016). SemEval-2016 task 5: Aspect based sentiment analysis. In S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, & T. Zesch (Eds.), *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)* (pp. 19–30). Association for Computational Linguistics. https://doi.org/10.18653/v1/S16-1002

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015). SemEval-2015 task 12: Aspect based sentiment analysis. In P. Nakov, T. Zesch, D. Cer, & D. Jurgens (Eds.), *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 486–495). Association for Computational Linguistics. https://doi.org/10.18653/v1/S15-2082

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 task 4: Aspect based sentiment analysis. In P. Nakov & T. Zesch (Eds.), *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (pp. 27–35). Association for Computational Linguistics. https://doi.org/10.3115/v1/S14-2004

Poole, K. T. & Rosenthal, H. (1985). A spatial model for legislative roll call analysis. *American Journal of Political Science*, *29*(2), 357–384. https://doi.org/10.2307/2111172

Poole, K. T. & Rosenthal, H. (1991). Patterns of congressional voting. *American Journal of Political Science*, *35*(1), 228–278. https://doi.org/10.2307/2111445

Porter, E. & Velez, Y. R. (2021). Placebo selection in survey experiments: An agnostic approach. *Political Analysis*, (pp. 1–14). https://doi.org/10.1017/pan.2021.16

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., & Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In S. Pradhan, A. Moschitti, & N. Xue (Eds.), *Joint Conference on EMNLP and CoNLL - Shared Task* (pp. 1–40). Association for Computational Linguistics. https://aclanthology.org/W12-4501

Princeton University (2010). *About WordNet*. https://wordnet.princeton.edu/

Proksch, S.-O. & Slapin, J. B. (2009). How to avoid pitfalls in statistical analysis of political texts: The case of Germany. *German Politics*, *18*(3), 323–344. https://doi.org/10.1080/09644000903055799

Proksch, S.-O. & Slapin, J. B. (2010). Position taking in European Parliament speeches. *British Journal of Political Science*, *40*(3), 587–611. https://doi.org/10.1017/S0007123409990299

Proksch, S.-O. & Slapin, J. B. (2012). Institutional foundations of legislative speech. *American Journal of Political Science*, *56*(3), 520–537. https://doi.org/10.1111/j.1540-5907.2011.00565.x

Quinn, K. M., Monroe, B. L., Colaresi, M., Crespin, M. H., & Radev, D. R. (2010). How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science*, *54*(1), 209–228. https://doi.org/10.1111/j.1540-5907.2009.00427.x

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 8748–8763). PMLR. https://proceedings.mlr.press/v139/radford21a.html

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. Manuscript. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. Manuscript. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. http://jmlr.org/papers/v21/20-074.html

Raganato, A., Camacho-Collados, J., & Navigli, R. (2017). Word sense disambiguation: A unified evaluation framework and empirical comparison. In M. Lapata, P. Blunsom, & A. Koller (Eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers* (pp. 99–110). Association for Computational Linguistics. https://aclanthology.org/E17-1010

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. arXiv. https://arxiv.org/abs/1806.03822

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. arXiv. https://arxiv.org/abs/1606.05250

Ramey, A. J., Klingler, J. D., & Hollibaugh, G. E. (2019). Measuring elite personality using speech. *Political Science Research and Methods*, *7*(1), 163–184. https://doi.org/10.1017/psrm.2016.12

Reddy, S., Chen, D., & Manning, C. D. (2019). CoQA: A conversational question answering challenge. arXiv. https://arxiv.org/abs/1808.07042

Rehbein, I., Lapesa, G., Glavaš, G., & Ponzetto, S. (Eds.) (2021a). *Proceedings of the 1st Workshop on Computational Linguistics for Political Text Analysis (CPSS-2021)*. German Society for Computational Linguistics & Language Technology. https://gscl.org/media/pages/arbeitskreise/cpss/cpss-2021/workshop-proceedings/352683648-1631172151/cpss2021-proceedings.pdf

Rehbein, I., Ponzetto, S. P., Adendorf, A., Bahnsen, O., Stoetzer, L., & Stuckenschmidt, H. (2021b). Come hither or go away? Recognising pre-electoral coalition signals in the news. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7798–7810). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.615

Rehbein, I., Ruppenhofer, J., & Bernauer, J. (2021c). Who is we? Disambiguating the referents of first person plural pronouns in parliamentary debates. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 147–158). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.13

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., & Kim, B. (2019). Visualizing and measuring the geometry of BERT. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf

Reimers, N. & Gurevych, I. (2018). Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. arXiv. http://arxiv.org/abs/1803.09578

Reimers, N. & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Rheault, L., Beelen, K., Cochrane, C., & Hirst, G. (2016). Measuring emotion in parliamentary debates with automated textual analysis. *PLoS ONE*, *11*(12), e0168843. https://doi.org/10.1371/journal.pone.0168843

Rheault, L. & Cochrane, C. (2020). Word embeddings for the analysis of ideological placement in parliamentary corpora. *Political Analysis*, *28*(1), 112–133. https://doi.org/10.1017/pan.2019.26

Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4902–4912). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.442

Riezler, S. & Maxwell, J. T. (2005). On some pitfalls in automatic evaluation and significance testing for MT. In J. Goldstein, A. Lavie, C.-Y. Lin, & C. Voss (Eds.), *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (pp. 57–64). Association for Computational Linguistics. https://aclanthology.org/W05-0908

Roberts, M. E., Stewart, B. M., & Airoldi, E. M. (2016). A model of text for experimentation in the social sciences. *Journal of the American Statistical Association*, *111*(515), 988–1003. https://doi.org/10.1080/01621459.2016.1141684

Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Luis, J. L., Gadarian, S. K., Albertson, B., & Rand, D. G. (2014). Structural Topic Models for open-ended survey responses. *American Journal of Political Science*, *58*(4), 1064–1082. https://doi.org/10.1111/ajps.12103

Rodman, E. (2020). A timely intervention: Tracking the changing meanings of political concepts with word vectors. *Political Analysis*, *28*(1), 87–111. https://doi.org/10.1017/pan.2019.23

Rodriguez, P. L. & Spirling, A. (2022). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. *The Journal of Politics*, *84*(1), 101–115. https://doi.org/10.1086/715162

Rothe, S. & Schütze, H. (2015). AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1793–1803). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-1173

Ruder, S. (2019a). *Neural transfer learning for natural language processing*. PhD thesis, National University of Ireland, Galway. https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf

Ruder, S. (2019b). Unsupervised cross-lingual representation learning. *Sebastian Ruder*. https://ruder.io/unsupervised-cross-lingual-learning/index.html

Ruder, S. (2020). Why you should do NLP beyond English. *Sebastian Ruder*. https://ruder.io/nlp-beyond-english/

Ruder, S. (2021a). ML and NLP research highlights of 2020. *Sebastian Ruder*. https://ruder.io/research-highlights-2020/

Ruder, S. (2021b). Recent advances in language model fine-tuning. *Sebastian Ruder*. https://ruder.io/recent-advances-lm-fine-tuning/

Ruder, S. (2022). ML and NLP research highlights of 2021. *Sebastian Ruder*. http://ruder.io/ml-highlights-2021/

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, S., & Sedlmair, M. (2018). More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures*, *12*(2–3), 140–157. https://doi.org/10.1080/19312458.2018.1455817

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 379–389). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1044

Saeidi, M., Bartolo, M., Lewis, P., Singh, S., Rocktäschel, T., Sheldon, M., Bouchard, G., & Riedel, S. (2018). Interpretation of natural language rules in conversational machine reading. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2087–2097). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1233

Salton, G. (1971). *The SMART retrieval system — Experiments in automatic document processing*. Prentice-Hall, Inc.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Sanders, J., Lisi, G., & Schonhardt-Bailey, C. (2018). Themes and topics in parliamentary oversight hearings: A new direction in textual data analysis. *Statistics, Politics and Policy*, *8*(2), 153–194. https://doi.org/10.1515/spp-2017-0012

Sap, M., Gabriel, S., Qin, L., Jurafsky, D., Smith, N. A., & Choi, Y. (2020). Social bias frames: Reasoning about social and power implications of language. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 5477–5490). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.486

Sarawagi, S. (2007). Information extraction. *Foundations and Trends in Databases*, *1*(3), 261–377.

Sarkar, D. (2016). *Text analytics with Python*. Apress. https://doi.org/10.1007/978-1-4842-2388-8

Schick, T. & Schütze, H. (2021). Exploiting cloze-questions for few-shot text classification and natural language inference. In P. Merlo, J. Tiedemann, & R. Tsarfaty (Eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (pp. 255–269). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.20

Schnell, R., Hill, P. B., & Esser, E. (2018). *Methoden der empirischen Sozialforschung (11th ed.)*. De Gruyter Oldenbourg.

Schulze, P. & Wiegrebe, S. (2020). Twitter in the Parliament — A text-based analysis of German political entities. Manuscript.

Schulze, P., Wiegrebe, S., Thurner, P. W., Heumann, C., Aßenmacher, M., & Wankmüller, S. (2021). Exploring topic-metadata relationships with the STM: A Bayesian approach. arXiv. https://arxiv.org/abs/2104.02496

Schuster, M. & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123. https://aclanthology.org/J98-1004

Schwarz, D., Traber, D., & Benoit, K. (2017). Estimating intra-party preferences: Comparing speeches to votes. *Political Science Research and Methods*, *5*(2), 379–396. https://doi.org/10.1017/psrm.2015.77

scikit-learn Developers (2021). Metrics and scoring: quantifying the quality of predictions. *scikit-learn*. https://scikit-learn.org/stable/modules/model_evaluation.html

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Sebők, M. & Kacsuk, Z. (2021). The multiclass classification of newspaper articles with machine learning: The hybrid binary snowball approach. *Political Analysis*, *29*(2), 236–249. https://doi.org/10.1017/pan.2020.27

Shepsle, K. A. & Bonchek, M. S. (2010). *Analyzing politics*. W. W. Norton.

Shmueli, B. (2019). Multi-class metrics made simple, part ii: The F1-score. *Towards Data Science*. https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

Slapin, J. B. & Kirkland, J. H. (2020). The sound of rebellion: Voting dissent and legislative speech in the UK House of Commons. *Legislative Studies Quarterly*, *45*(2), 153–176. https://doi.org/10.1111/lsq.12251

Slapin, J. B. & Proksch, S.-O. (2008). A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, *52*(3), 705–722. https://doi.org/10.1111/j.1540-5907.2008.00338.x

Smith, N. A. (2011). *Linguistic structure prediction*. Morgan & Claypool Publishers.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In R. Barzilay & M. Johnson (Eds.), *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 151–161). Association for Computational Linguistics. https://aclanthology.org/D11-1014

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1631–1642). Association for Computational Linguistics. https://aclanthology.org/D13-1170

Song, H., Tolochko, P., Eberl, J.-M., Eisele, O., Greussing, E., Heidenreich, T., Lind, F., Galyga, S., & Boomgaarden, H. G. (2020). In validations we trust? The impact of imperfect human annotations as a gold standard on the quality of validation of automated content analysis. *Political Communication*, *37*(4), 550–572. https://doi.org/10.1080/10584609.2020.1723752

Spirling, A. & Rodriguez, P. L. (2020). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. Manuscript. http://arthurspirling.org/documents/embed.pdf

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Stein, B. & Wachsmuth, H. (Eds.) (2019). *Proceedings of the 6th Workshop on Argument Mining*. Association for Computational Linguistics. https://aclanthology.org/W19-4500

Stimson, J. A., Mackuen, M. B., & Erikson, R. S. (1995). Dynamic representation. *American Political Science Review*, *89*(03), 543–565. https://doi.org/10.2307/2082973

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3104–3112). MIT Press. https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2021). Long Range Arena: A benchmark for efficient Transformers.

In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*. https://openreview.net/forum?id=qVyeW-grC2k

Tay, Y., Tran, V. Q., Ruder, S., Gupta, J., Chung, H. W., Bahri, D., Qin, Z., Baumgartner, S., Yu, C., & Metzler, D. (2022). Charformer: Fast character Transformers via gradient-based subword tokenization. In K. Hofmann & A. Rush (Eds.), *10th International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=JtBRnrlOEFN

Tenney, I., Das, D., & Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4593–4601). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1452

Theocharis, Y., Barberá, P., Fazekas, Z., Popa, S. A., & Parnet, O. (2016). A bad workman blames his tweets: The consequences of citizens' uncivil Twitter use when interacting with party candidates. *Journal of Communication*, *66*(6), 1007–1031. https://doi.org/10.1111/jcom.12259

Thomas, G. (2018). *Mathematics for machine learning* [Notes]. University of California, Berkeley. https://gwthomas.github.io/docs/math4ml.pdf

Tjong Kim Sang, E. F. & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (pp. 142–147). https://aclanthology.org/W03-0419

Torres, M. & Cantu, F. (2022). Learning to see: Convolutional neural networks for the analysis of social science data. *Political Analysis*, *30*(1), 113–131. https://doi.org/10.1017/pan.2021.9

Toutenburg, H., Heumann, C., & Nittner, T. (2004). Statistische Methoden bei unvollständigen Daten. Open Access LMU. https://doi.org/10.5282/ubm/epub.1750

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In P. Isabelle (Ed.), *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417–424). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073153

Turney, P. D. & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37, 141–188. https://doi.org/10.1613/jair.2934

Ulmer, D., Bassignana, E., Müller-Eberstein, M., Varab, D., Zhang, M., Hardmeier, C., & Plank, B. (2022). Experimental standards for deep learning research: A natural language processing perspective. In S. Chan, R. Agarwal, X. Bouthillier, C. Gulcehre, & J. Dodge (Eds.), *ICLR Workshop on ML Evaluation Standards*. https://arxiv.org/abs/2204.06251

Vapnik, V. (1991). Principles of risk minimization for learning theory. In J. Moody, S. Hanson, & R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems 4* (pp. 831–838). Morgan-Kaufmann. https://proceedings.neurips.cc/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Volkens, A., Burst, T., Krause, W., Lehmann, P., Matthieß, T., Regel, S., Weßels, B., & Zehnter, L. (2021a). *The Manifesto Data Collection. Manifesto Project (MRG / CMP / MARPOR). Version 2021a.* [Data set]. Wissenschaftszentrum Berlin für Sozialforschung (WZB). https://doi.org/10.25522/manifesto.mpds.2021a

Volkens, A., Burst, T., Krause, W., Lehmann, P., Matthieß, T., Regel, S., Weßels, B., & Zehnter, L. (2021b). *The Manifesto Project Dataset - Codebook. Manifesto Project (MRG / CMP / MARPOR). Version 2021a.* Wissenschaftszentrum Berlin für Sozialforschung (WZB). https://manifesto-project.wzb.eu/down/data/2021a/codebooks/codebook_MPDataset_MPDS2021a.pdf

Walker, C., Strassel, S., Medero, J., & Maeda, K. (2006). *ACE 2005 Multilingual training corpus.* [Data set]. Linguistic Data Consortium. https://doi.org/10.35111/mwxc-vh88

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 3266–3280). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf

Wang, X., Liu, Y., Sun, C., Wang, B., & Wang, X. (2015). Predicting polarities of tweets by composing word embeddings with long short-term memory. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1343–1353). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-1130

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2022). Finetuned language models are zero-shot learners. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022).* https://openreview.net/forum?id=gEZrGCozdqR

Wiseman, S., Shieber, S., & Rush, A. (2017). Challenges in data-to-document generation. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 Conference on Em-*

pirical Methods in Natural Language Processing (EMNLP) (pp. 2253–2263). Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1239

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). Hugging Face's Transformers: State-of-the-art natural language processing. arXiv. https://arxiv.org/abs/1910.03771v5

Wood, W. (1982). Retrieval of attitude-relevant information from memory: Effects on susceptibility to persuasion and on intrinsic motivation. *Journal of Personality and Social Psychology*, *42*(5), 798–810.

Wu, P. Y. & Mebane, W. R. (2021). MARMOT: A deep learning framework for constructing multimodal representations for vision-and-language tasks. arXiv. https://arxiv.org/abs/2109.11526v1

Wu, S. & Dredze, M. (2019). Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 833–844). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1077

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10, 291–306. https://doi.org/10.1162/tacl_a_00461

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, & Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 483–498). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.41

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, (pp. 5753–5763). Curran Associates, Inc. http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

Yin, W., Hay, J., & Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Pro-*

cessing (EMNLP-IJCNLP) (pp. 3914–3923). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1404

Ying, L., Montgomery, J. M., & Stewart, B. M. (2021). Topics, concepts, and measurement: A crowdsourced procedure for validating topics as measures. *Political Analysis*, (pp. 1–20). https://doi.org/10.1017/pan.2021.33

Zellers, R., Bisk, Y., Schwartz, R., & Choi, Y. (2018). SWAG: A large-scale adversarial dataset for grounded commonsense inference. arXiv. https://arxiv.org/abs/1808.05326

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4791–4800). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1472

Zhang, H. & Pan, J. (2019). CASM: A deep-learning approach for identifying collective action events with text and image data from social media. *Sociological Methodology*, *49*(1), 1–57. https://doi.org/10.1177/0081175019860244

Zhang, M., Zhang, Y., & Vo, D.-T. (2015a). Neural networks for open domain targeted sentiment. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 612–621). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1073

Zhang, M.-L. & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

Zhang, X., Zhao, J., & LeCun, Y. (2015b). Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Proceedings of the 28th International Conference on Neural Information Processing Systems* (pp. 649–657). MIT Press. https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

Zhao, H., Phung, D., Huynh, V., Jin, Y., Du, L., & Buntine, W. (2021). Topic modelling meets deep neural networks: A survey. In Z.-H. Zhou (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (pp. 4713–4720). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2021/638

Zhou, L., Xu, C., & Corso, J. J. (2017). Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press. https://ojs.aaai.org/index.php/AAAI/article/download/12342/12201

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching

movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)* (pp. 19–27). IEEE. https://doi.org/10.1109/ICCV.2015.11

Chapter 2

Introduction to Neural Transfer Learning with Transformers for Social Science Text Analysis

Article.

The following article version is the version at the time this dissertation has been handed in. An updated version of the following article has been published in the journal *Sociological Methods & Research* and is available online at https://doi.org/10.1177/00491241221134527. The full citation information for the published article is as follows:

Wankmüller, S. (2022). Introduction to neural transfer learning with Transformers for social science text analysis. *Sociological Methods & Research*, (p. 1–77). https://doi.org/10.1177/00491241221134527

A version of this article also is available at https://arxiv.org/abs/2102.02111

Source Code. https://doi.org/10.6084/m9.figshare.14394173

Introduction to Neural Transfer Learning with Transformers for Social Science Text Analysis

Abstract. Transformer-based models for transfer learning have the potential to achieve high prediction accuracies on text-based supervised learning tasks with relatively few training data instances. These models thus are likely to benefit social scientists that seek to have as accurate as possible text-based measures and probably have only limited resources for annotating training data. To enable social scientists to leverage these potential benefits for their research, this paper explains how these methods work, why they might be advantageous, and what their limitations are. Additionally, three Transformer-based models for transfer learning, BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and the Longformer (Beltagy et al., 2020), are compared to conventional machine learning algorithms on three applications. Across all evaluated tasks, textual styles, and training data set sizes, the conventional models are consistently outperformed by transfer learning with Transformers, thereby demonstrating the benefits these models can bring to text-based social science research.

Keywords. Natural language processing, deep learning, neural networks, transfer learning, Transformer, BERT

Acknowledgements. I am very grateful to Paul W. Thurner, Christian Heumann, Matthias Aßenmacher, the participants of the colloquium at the chair of Paul W. Thurner, and three anonymous reviewers for their highly valuable guidance and helpful comments on this work.

2.1 Introduction: Why Neural Transfer Learning with Transformers?

Social scientists have at their disposal a wide spectrum of text analysis tools (Grimmer & Stewart, 2013; Benoit, 2020). And different tools are suitable to achieve different goals. Social scientists use unsupervised learning methods to detect latent topic structures (e.g. Roberts et al., 2016) and to estimate positions of texts on uncovered continuous latent dimensions (e.g. Slapin & Proksch, 2008). Whilst unsupervised learning is essential for exploring, discovering, and mapping yet unknown (and possibly theoretically informative) underlying properties of texts, supervised learning methods are important in situations in which a researcher has a clearly defined concept and seeks to measure the concept by means of using textual data in an automated process (Ahlquist & Breunig, 2012, p. 94-95; Grimmer & Stewart, 2013, p. 268, 270). Supervised learning techniques have been employed to measure a vast range of application-specific (and often complex, latent, and multidimensional) concepts from texts, such as e.g. tonality (Rudkowsky et al., 2018; Bar-

berá et al., 2021; Fowler et al., 2021), inequality (Nelson et al., 2021), populism (Di Cocco & Monechi, 2021), attitudes (Ceron et al., 2014; Mitts, 2019), policy topics (Osnabrügge et al., 2021; Sebők & Kacsuk, 2021), and events (D'Orazio et al., 2014; Zhang & Pan, 2019; Muchlinski et al., 2021). In such supervised learning settings, the training data encode how the concept (e.g. attitude, inequality, event) is to be operationalized and the text analysis method is the measurement method that is deployed to assign the textual units to the values of the variable.

If a researcher is applying a supervised learning method on text data for the purpose of measuring an a priori-specified concept, her aim—as in any measurement process—will be to have a valid measure that captures the concept it is devised to measure. And consequently—because when working with text data humans are usually seen as the "the ultimate arbiter of the 'validity' of any research exercise" (Benoit, 2020, p. 470)—the aim for the researcher is to have a supervised learning technique that as closely as possible can imitate human codings (Grimmer & Stewart, 2013, p. 270, 279).[1] After having trained a model on human annotated training data, the researcher thus will hope that the trained model as accurately as possible predicts human codings on data that have not been used in training (Grimmer & Stewart, 2013, p. 271, 279). If this is the case and hence the model can be said to generalize well, this indicates that the model's predictions will provide a valid measure of the concept under study (Grimmer & Stewart, 2013, p. 271, 279).

This focus on prediction performance is a major deviation from the usual social science focus on making causal inferences. In a causal inference setting, modeling is theory-based and interpretable models are used to identify the effects of single independent variables. But in order to test hypotheses about causal relations between concepts, the concepts have to be translated into measurable variables that constitute valid measures of the concepts under study. And if for the process of measurement a supervised learning method is used, then the goal is to as closely as possible replicate human coding as this indicates validity (Grimmer & Stewart, 2013, p. 271, 279). So here, for the very purpose of measurement, the aim is not causal inference but precise prediction. Therefore, the focus is less on interpretable models that allow identifying the effects of single features.[2] Rather, the focus is on models that as closely as possible can approximate the often very complex functions that map from input text data to the respective human coded outputs (Breiman, 2001, p. 199).

In the field of natural language processing (NLP), the usage of deep learning models (as compared to conventional machine learning algorithms) has enabled researchers to learn

[1] Benoit (2020, p. 470) points out that research indicates that humans are not very reliable coders of text data (see also e.g. Mikhaylov et al., 2012; Ennser-Jedenastik & Meyer, 2018). This, in turn, raises the question of how valid human judgments can be (Song et al., 2020, p. 553). Nevertheless, in this study—and in concordance with the literature (Benoit, 2020, p. 470; Nelson et al., 2021, p. 232)—the comparison of human codings to the predictions of a supervised learning method is considered the best available procedure for validation.

[2] In studies that apply supervised learning approaches for purposes other than the one discussed here interpretability can be essential (see e.g. Slapin & Kirkland, 2020).

better generalizing mappings from textual inputs to task-specific outputs and hence has enabled researchers to more accurately perform a wide spectrum of tasks such as text classification, machine translation, or reading comprehension (Goldberg, 2016, p. 347-348; Ruder, 2020). Despite the fact that deep learning techniques tend to exhibit higher prediction accuracies in text-based supervised learning tasks compared to traditional machine learning algorithms (Socher et al., 2013; Iyyer et al., 2014; Budhwar et al., 2018; Ruder, 2020), they are not yet a standard tool for social science researchers that use supervised learning for text analysis. Although there are exceptions (e.g. Rudkowsky et al., 2018; Zhang & Pan, 2019; Amsalem et al., 2020; Chang & Masterson, 2020; Muchlinski et al., 2021; Wu & Mebane, 2021), for the implementation of supervised learning tasks, social scientists typically resort to bag-of-words-based representations of texts that serve as an input to conventional machine learning models such as support vector machines (SVMs), naive Bayes, random forests, boosting algorithms, or regression with regularization (see e.g. Diermeier et al., 2011; Colleoni et al., 2014; D'Orazio et al., 2014; Ceron et al., 2015; Theocharis et al., 2016; Welbers et al., 2017; Kwon et al., 2018; Greene et al., 2019; Katagiri & Min, 2019; Mitts, 2019; Pilny et al., 2019; Ramey et al., 2019; Rona-Tas et al., 2019; Anastasopoulos & Bertelli, 2020; Miller et al., 2020; Park et al., 2020; Barberá et al., 2021; Di Cocco & Monechi, 2021; Fowler et al., 2021; Osnabrügge et al., 2021; Sebők & Kacsuk, 2021).[3]

One among several likely reasons why deep learning methods so far have not been widely used for text-based supervised learning tasks by social scientists might be that training deep learning models is resource intensive. Deep learning models have considerably more parameters to be learned in training than classic machine learning models. Consequently, deep learning models are computationally highly intensive and require substantially larger numbers of training examples. Goodfellow et al. (2016, p. 20) stated that "As of 2016, a rough rule of thumb is that a supervised deep learning algorithm will generally achieve acceptable performance with around 5,000 labeled examples per category".[4] For research

[3]This is not to say that social scientists would not have started to leverage the foundations of deep learning approaches in NLP: During the last years, the use of real-valued vector representations of terms, known as word embeddings, enabled social scientists to explore new research questions or to study old research questions by new means (e.g. Rheault et al., 2016; Han et al., 2018; Kozlowski et al., 2019; Rheault & Cochrane, 2020; Rodman, 2020; Watanabe, 2021). Moreover, there is a small but increasing number of publications in social science journals that apply deep neural networks to texts (e.g. Rudkowsky et al., 2018; Zhang & Pan, 2019; Amsalem et al., 2020; Chang & Masterson, 2020; Muchlinski et al., 2021; Wu & Mebane, 2021). Yet applications of deep neural networks (let alone deep neural networks plus transfer learning) are not widely used by social scientists. And thus, implementations of deep neural networks and modern NLP techniques on texts that are relevant for social science research up til now are mostly conducted by research teams that are not primarily social science trained (see e.g. Iyyer et al., 2014; Zarrella & Marsh, 2016; Glavaš et al., 2017; Budhwar et al., 2018; Meidinger & Aßenmacher, 2021) and/or are published via platforms and venues (e.g. important NLP conferences such as the EMNLP, ACL, or NAACL) that social scientists typically do not closely monitor (e.g. Kim et al., 2021; Rehbein et al., 2021a,b).

[4]Yet how much training data instances are really needed depends on the width and depth of the deep neural network, the task, and training data quality. Thus, precise numbers on the amounts of

questions relating to domains in which it is difficult to access or label large enough numbers of training data instances, deep learning becomes infeasible or prohibitively costly.

What kind of NLP tasks are there? In the field of NLP, a large spectrum of diverse tasks are addressed. There are NLP tasks that operate at the linguistic level (e.g. part-of-speech (POS) tagging, syntactic parsing) (Smith, 2011, p. 4-11), and there are tasks that operate at the semantic level and focus on natural language understanding (e.g. information extraction, sentiment analysis, or question answering) (MacCartney, 2014). Furthermore, there are natural language generation tasks (e.g. machine translation, text summarization) (Gatt & Krahmer, 2018), and there are multimodal tasks in which the inputs to be processed can be of different modalities (e.g. text plus image, audio, or video). Additionally, these tasks can be approached in different formats. Sentiment analysis, for example, can be conducted as a document classification task (Pang et al., 2002), a sequence tagging task (Mitchell et al., 2013), or a span extraction task (Hu et al., 2019). Especially with regard to natural language understanding, however, many NLP tasks can be framed as binary or multi-class classification tasks in which the model's task is to assign one out of two or one out of several class labels to each text input (see e.g. Wang et al., 2019). This matches well with text-based research in social science where the measurement of an a priori-defined concept via supervised learning is very frequently implemented as a text classification task.[a]

[a]This is not to say that all supervised learning in social science is classification. Especially in political science, supervised techniques that estimate values for documents on latent continuous dimensions have been developed (Laver et al., 2003; Perry & Benoit, 2017). For a new technique see Wankmüller & Heumann (2021).

Recent developments within NLP on transfer learning alleviate this problem. Transfer learning is a set of learning procedures in which knowledge that has been learned from training on a source task in a source domain is used to improve learning on the target task in the target domain (where the target task is the task of interest that a researcher actually seeks to conduct) (Pan & Yang, 2010, p. 1347). In sequential transfer learning—which is one common type of transfer learning—the aim when training on a source task is to acquire a highly general, close to universal language representation model (Ruder, 2019a, p. 64). The pretrained general-purpose representation model then can be used as an input to a target task of interest (Ruder, 2019a, p. 63-64). This practice of using a pretrained language model as an initialization for training on a target task has been shown to improve the prediction performances on a large variety of NLP target tasks (Ruder, 2020; Bommasani et al., 2021, p. 22-23). Moreover, adapting a pretrained language model

parameters and required training examples cannot be specified. To nevertheless put the sizes in relation, note that an SVM with a linear kernel that learns to construct a hyperplane in a 3,000-dimensional feature space which separates instances into two categories based on 1,000 support vectors has around 3 million parameters. The Transformer-based models presented in this article, in contrast, have well above 100 million parameters.

to a target task requires fewer target training examples than when not using transfer learning and training the model from scratch on the target task (Howard & Ruder, 2018, p. 334).

In addition to the efficiency and performance gains from research on transfer learning, the introduction of the attention mechanism (Bahdanau et al., 2015) and the self-attention mechanism (Vaswani et al., 2017) has significantly improved the ability of deep learning NLP models to capture contextual information from texts. (Self-)attention mechanisms learn a token representation by capturing information from other tokens, and thereby encode textual dependencies and context-dependent meanings. (Self-)attention mechanisms constitute the core building blocks of the Transformer—a type of deep learning model that has been presented by Vaswani et al. in 2017. During the last years, several Transformer-based models that are used in a transfer learning setting have been introduced (e.g. Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019). These models substantively outperform previous state-of-the-art models across a large variety of NLP tasks (Ruder, 2020; Bommasani et al., 2021, p. 22-23).

Due to the likely increases in prediction accuracy, as well as the efficient and less resourceful adaptation phase, transfer learning with deep (e.g. Transformer-based) language representation models seems promising to social science researchers. It seems especially promising to researchers that seek to have as accurate as possible text-based measures but lack the resources to annotate large amounts of data or are interested in specific domains in which only small corpora and few training instances exist. In order to equip social scientists to use the potential of transfer learning with Transformer-based models for their research, this paper provides an introduction to transfer learning and the Transformer. Hence, the contribution of this paper is to present learning techniques that might enable social scientists to obtain more valid text-based measures for concepts they seek to measure.

The following Section 2.2 compares conventional machine learning to deep learning by focusing on the question of how textual features (e.g. characters, terms, symbols) and larger textual units (e.g. sentences, paragraphs, tweets, comments, speeches, ... here named: documents) tend to be represented in conventional vs. deep learning approaches. The subsequent Section 2.3 on transfer learning provides an answer to the question of what transfer learning is and explains in more detail in what ways transfer learning might be beneficial. The then following Section 2.4 introduces the attention mechanism and the Transformer and elaborates on how the Transformer has advanced the study of text. Afterward, an overview of Transformer-based models for transfer learning is provided (Section 2.5). Here, a special focus will be given to the seminal Transformer-based language representation model BERT (standing for Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). Additionally, the changes in NLP and artificial intelligence (AI) research, that these models have caused, are outlined and problematic aspects are discussed. Finally, three Transformer-based models for transfer learning, BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and the Longformer (Beltagy et al., 2020), are compared to traditional learning algorithms based on three classification tasks using data

from speeches in the UK parliament (Duthie & Budzynska, 2018), tweets regarding the legalization of abortion (Mohammad et al., 2017), and comments from Wikipedia Talk pages (Jigsaw/Conversation AI, 2018) (Section 2.6). The final Section 2.7 concludes with a discussion on task-specific factors and research goals for which neural transfer learning with Transformers is highly beneficial vs. rather limited.

Throughout the paper, it is assumed that readers know core elements of neural network architectures, and are also familiar with recurrent neural networks (RNNs) as well as with optimization via stochastic gradient descent with backpropagation. For readers that feel not sufficiently acquainted with these deep learning concepts see Appendix 2.A. Also note that a document is an ordered sequence of tokens and here is denoted as $d_i = (a_1, \ldots, a_t, \ldots, a_T)$. A token a_t is an instance of a type, which is the set of all tokens that are made up of the same string of characters (Manning et al., 2008, p. 22). A type that is used for analysis is named term or feature and here is given as z_u. The set of features that are used in an analysis is $\{z_1, \ldots, z_u \ldots, z_U\}$.

2.2 Conventional Machine Learning vs. Deep Learning

2.2.1 Conventional Machine Learning

Given raw input data $D = (d_1, \ldots, d_i, \ldots, d_N)$ (e.g. a corpus comprising N raw text files) and a corresponding output variable $\boldsymbol{y} = [y_1, \ldots, y_i, \ldots, y_N]^\top$ (e.g. class labels), the aim in supervised machine learning is to find the parameters $\boldsymbol{\theta}$ of a function f that captures the general systematic relation between D and $\boldsymbol{y}$ such that the trained model will generalize well and generate accurate predictions for new, yet unseen data D_{test} (James et al., 2013, p. 30; Chollet, 2021, ch. 1.1.3).

When applying a machine learning algorithm in order to learn a function that as accurately as possible maps from text data inputs to provided outputs, the algorithm, however, will not take as an input raw text documents. The raw text units first have to be converted into a format that is suitable for data analysis (Benoit, 2020, p. 463-464). This is achieved by transforming each raw data unit d_i into an abstracted representation of d_i (Benoit, 2020, p. 463-464). Learning in supervised machine learning hence essentially is a two-step process (Goodfellow et al., 2016, p. 10): The first step is to create or learn representations of the data, and the second step is to learn mappings from these representations of the data to the output. For a single document d_i, the first step of being transformed into a representation can be described as $\mathsf{f}_l(d_i, \hat{\boldsymbol{\theta}}_l)$ and the entire process as

$$\hat{y}_i = \mathsf{f}(d_i, \hat{\boldsymbol{\theta}}) = \mathsf{f}_o(\mathsf{f}_l(d_i, \hat{\boldsymbol{\theta}}_l), \hat{\boldsymbol{\theta}}_o) \tag{2.1}$$

where the subscript l indicates the mapping from raw data to a representation and the subscript o indicates the mapping from the representation to the output. Conventional

machine learning algorithms cover the second step: They learn a function mapping data representations to the output. This in turn implies that the first step falls to the researcher who has to (manually) generate representations of the data herself.

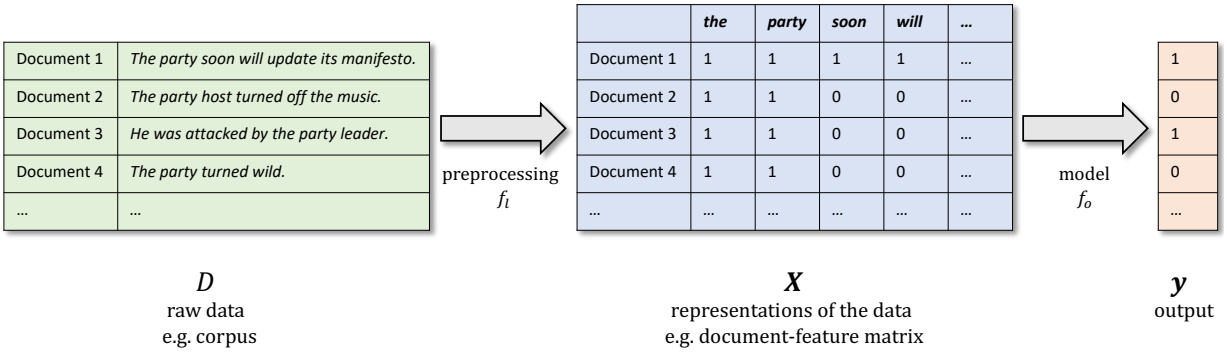


Figure 2.1: **Learning as a Two-Step Process.** In text-based applications of conventional machine learning approaches, the raw data D first are (manually) preprocessed such that each example is represented as a feature vector in the document-feature matrix $\boldsymbol{X}$. Second, these representations of the data are fed as inputs to a traditional machine learning algorithm that learns a mapping between data representations $\boldsymbol{X}$ and outputs $\boldsymbol{y}$.

When working with texts, the raw data D are typically a corpus of text documents. A very common approach in the social sciences is to transform the raw text files via multiple preprocessing procedures into a document-feature matrix $\boldsymbol{X} = [\boldsymbol{x}_1|\ldots|\boldsymbol{x}_i|\ldots|\boldsymbol{x}_N]^\top$ (see Figure 2.1) (Benoit, 2020, p. 464). In a document-feature matrix, each document is represented as a feature vector $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iu}, \ldots, x_{iU}]$ (Turney & Pantel, 2010, p. 147). Element x_{iu} in this vector gives the value of the ith document on the uth textual feature—and typically is the (weighted) number of times that the uth feature occurs in the ith document (Turney & Pantel, 2010, p. 143, 147). To conduct the second learning step, the researcher then commonly applies a conventional machine learning algorithm on the document-feature matrix to learn the relation between the document-feature representation of the data $\boldsymbol{X}$ and the provided response values $\boldsymbol{y}$.

There are three difficulties with this approach. The first is that it may be hard for the researcher to a priori know which features are useful for the task at hand (Goodfellow et al., 2016, p. 3-5). The performance of a supervised learning algorithm will depend on the representation of the data in the document-feature matrix (Goodfellow et al., 2016, p. 3-4). In a classification task, features that capture observed linguistic variation that helps in assigning the texts into the correct categories are more informative and will lead to a better classification performance than features that capture variation that is not helpful in distinguishing between the classes (Goodfellow et al., 2016, p. 3-5). Yet determining which sets of possibly highly abstract and complex features are informative (and which are not) is highly difficult (Goodfellow et al., 2016, p. 3-5). A researcher can choose from a multitude of possible preprocessing steps such as stemming, lowercasing, removing

stopwords, adding POS tags, or applying a sentiment lexicon.[5] Social scientists may be able to use some of their domain knowledge in deciding upon a few specific preprocessing decisions (e.g. whether it is likely that excluding a predefined list of stopwords will be beneficial because it reduces dimensionality or will harm performance because the stopword list includes terms that are important). Domain knowledge, however, is most unlikely to guide researchers regarding all possible permutations of preprocessing steps. Simply trying out each possible preprocessing permutation in order to select the best performing one for a supervised task is not possible given the massive number of permutations and limited researcher resources.

Second, the document-feature matrix defines a representational space in which each feature constitutes one separate and independent dimension of the space (Goldberg, 2016, p. 349-350). Accordingly, if there are U features, $\{z_1, \ldots, z_u, \ldots, z_U\}$, then each feature z_u defines one dimension of the representational space. This implies that each feature is represented to be as distant (and thus as dissimilar) to one feature as to each other feature (Goldberg, 2016, p. 351). The terms *'excellent'* and *'outstanding'* are treated as (dis)similar to each other as the terms *'excellent'* and *'terrible'*. The representational setting of the document-feature matrix furthermore implies that the number of features (here denoted with U) defines the dimensionality of the representational space (Goldberg, 2016, p. 349). As—even after feature exclusion and feature normalization—the number of features in any text-based analysis typically tends to be high, the document representation vectors $\boldsymbol{x}_i$ tend to be high-dimensional and sparse. (This is, $\boldsymbol{x}_i$ is likely to be a vector with a large number of elements, most of which will be zero.) By defining such a high-dimensional and sparse feature space, a document-feature matrix brings about the curse of dimensionality: There are much more combinations of feature values than can be covered by the training data, therefore making it difficult to generalize to regions of the space for which no or only few training data are observed (Bengio et al., 2003, p. 1137-1138; Goodfellow et al., 2016, p. 351).

The third problem is that in a document-feature matrix each document is represented as a bag-of-words (Turney & Pantel, 2010, p. 147). Bag-of-words-based representations disregard word order and syntactic or semantic dependencies between words in a sequence (Turney & Pantel, 2010, p. 147).[6] Yet text is contextual and sequential by nature. Word order carries meaning. And the context, in which a word is embedded in, is essential in determining the meaning of a word. When represented as a bag-of-words, the sentence

[5] For a more detailed list of possible steps see Turney & Pantel (2010, p. 153 ff.) and Denny & Spirling (2018, p. 170-172). Note that not only the set of selected preprocessing steps but also the order in which they are implemented define the way in which the texts at hand are represented and thus affect the research findings (Denny & Spirling, 2018).

[6] By counting the occurrence of word sequences of length N, N-gram models extend unigram-based bag-of-words models and allow for capturing information from small contexts around words. However, by including N-grams as features, the dimensionality of the feature space increases, thereby increasing the problem of high dimensionality and sparsity. Moreover, texts often exhibit dependencies between words that are positioned much farther apart than what could be captured with N-gram models (Chang & Masterson, 2020, p. 395).

'The opposition party leader attacked the prime minister.' cannot be distinguished from the sentence *'The prime minister attacked the opposition party leader.'*. Moreover, the fact that the word *'party'* here refers to a political party rather than a festive social gathering only becomes clear from the context.

2.2.2 Deep Learning and Embeddings

These stated problems are overcome by deep neural networks and the real-valued vector representations, known as embeddings, that typically accompany deep neural networks (Goldberg, 2016; Goodfellow et al., 2016). In contrast to conventional machine learning algorithms, deep learning models can be considered to conduct both learning steps: They learn representations of the data *and* a function mapping data representations to the output. In deep learning models, an abstract representation of the data is learned by applying the data to a stack of several simple (typically nonlinear) functions (Goodfellow et al., 2016, p. 5, 164-165). Each function takes as an input the representation of the data created by (the sequence of) previous functions and generates a new representation:

$$\mathsf{f}(d_i, \hat{\boldsymbol{\theta}}) = \mathsf{f}_o(\ldots \mathsf{f}_{l_3}(\mathsf{f}_{l_2}(\mathsf{f}_{l_1}(d_i, \hat{\boldsymbol{\theta}}_{l_1}), \hat{\boldsymbol{\theta}}_{l_2}), \hat{\boldsymbol{\theta}}_{l_3}) \ldots, \hat{\boldsymbol{\theta}}_o) \tag{2.2}$$

Deep learning models thus are characterized by providing a layered representation of the data in which each layer of representation is based on previous data representations (Goodfellow et al., 2016, p. 5, 8). Hereby, complex and abstract representations are learned from less abstract, more simple ones (Goodfellow et al., 2016, p. 5, 8).

Nevertheless, when applying deep neural networks to text-based applications, deep neural networks do not take as an input the raw text documents. They still have to be fed with a data format they can read. Neural networks usually operate on real-valued vector representations of entities, named embeddings (Goldberg, 2016, p. 349-351). Frequently, the embedded entities are unique vocabulary terms (Pilehvar & Camacho-Collados, 2020, p. 5). (In this case, embeddings are referred to as word embeddings.) Yet embeddings also can be learned for smaller textual units (e.g. characters (Akbik et al., 2018) and subwords (Bojanowski et al., 2017)), larger textual units such as sentences or documents (Le & Mikolov, 2014; Reimers & Gurevych, 2019), and even for entities of a different nature, e.g. word senses (Rothe & Schütze, 2015) or the nodes in a network (Kipf & Welling, 2017).

When working with text data and having a set of U textual features (e.g. U vocabulary terms in a corpus), which are given by $\{z_1, \ldots, z_u, \ldots, z_U\}$, then each feature z_u can be represented as an embedding—a K-dimensional real-valued vector $\boldsymbol{z}_u \in \mathbb{R}^K$. Whereas in a document-feature matrix representation z_u is a dimension of a U-dimensional feature space, now z_u is represented as a dense vector $\boldsymbol{z}_u$ that is embedded in a K-dimensional continuous space (where typically $K << U$) (Goldberg, 2016, p. 350-351). The positioning of the embedding vectors within this K-dimensional space reflects the information that

the embeddings encode about the features. For example, if the embeddings encode the feature's semantics, then features that are semantically similar are likely to have close embedding vectors and thus are likely to be positioned close in space (Pilehvar & Camacho-Collados, 2020, p. 4-5, 39). (The terms *'excellent'* and *'outstanding'* then are likely to be close together and far from *'terrible'*.) Learning real-valued vector representations for textual features and documents implies that one obtains relatively low-dimensional and dense (rather than high-dimensional and sparse) representations (Goldberg, 2016, p. 349-351). This, in turn, much facilitates generalization via the employment of local smoothness assumptions (Bengio et al., 2003, p. 1137-1140).

In text-based applications, the feature representation vectors can be collectively kept in an embedding matrix $\boldsymbol{E}$, which is a $U \times K$ matrix that stores for each of the U unique features its K-dimensional embedding $\boldsymbol{z}_u$ (Goldberg, 2016, p. 360). Therefore, if a researcher wants to feed a text document, $d_i = (a_1, \ldots, a_t, \ldots, a_T)$, to a neural network, then for each token a_t, the respective feature embedding $\boldsymbol{z}_{[a_t]}$ is retrieved from the embedding matrix $\boldsymbol{E}$ (Goldberg, 2016, p. 360). In the end, the document $(a_1, \ldots, a_t, \ldots, a_T)$ is mapped to a sequence of embeddings $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_t]}, \ldots, \boldsymbol{z}_{[a_T]})$ which is the input representation entering the network (Ruder, 2019a, p. 33). The values of the elements of the embedding vector $\boldsymbol{z}_u$ of each feature are treated as usual parameters and are learned jointly with the other model parameters in the optimization process (Goldberg, 2016, p. 349, 361). A researcher that has a corpus of raw text documents at his disposal thus merely has to extract features $\{z_1, \ldots, z_u, \ldots, z_U\}$ for which vector representations will be learned (Goldberg, 2016, p. 349-353). In practice, this typically involves tokenization and sometimes normalization (e.g. lowercasing).[7] Other than that, no text preprocessing steps are required. The representation $\boldsymbol{z}_u$ for each extracted feature is learned when training the model. It does not have to be manually prefabricated by the researcher.

Nevertheless, it is common practice to initialize the representation vectors $\boldsymbol{z}_u$ with pretrained embeddings (Goldberg, 2016, p. 365). Continuous bag-of-words (CBOW) (Mikolov et al., 2013a), Skip-gram (Mikolov et al., 2013a,b), and Global Vectors (GloVe) (Pennington et al., 2014), are early seminal models that learn (pretrained) word embeddings. In these models, the embedding for a target term z_u is learned on the basis of words that occur in a context window surrounding instances of term z_u (Pennington et al., 2014, p. 1533-1535). In CBOW, for example, the self-supervised learning task is to predict a word given its context words (Mikolov et al., 2013a, p. 4-5). In Skip-gram, surrounding context words are predicted given a target word (Mikolov et al., 2013a, p. 4-5). And GloVe seeks to find a representation for term z_u and context term z_j such that the dot product of their representation vectors, $\boldsymbol{z}_u^\top \tilde{\boldsymbol{z}}_j$, has a minimal squared difference to the logged number of times that z_j occurs in a context window around z_u (Pennington et al., 2014, p. 1535). By utilizing the contexts of a term to learn a representation for this term, these models implement the

[7] For example, a researcher may decide to tokenize each document into words and then lowercase all words such that each document $(a_1, \ldots, a_t, \ldots, a_T)$ is given as a sequence of lowercased words. The set of features $\{z_1, \ldots, z_u, \ldots, z_U\}$ consequently comprises all unique lowercased word tokens in the corpus.

distributional hypothesis (Firth, 1957) according to which the meaning of a term can be inferred from its context (Goldberg, 2016, p. 365; Spirling & Rodriguez, 2020, p. 4). Similar terms are expected to be observed in similar contexts and, consequently, semantically or syntactically similar terms are expected to be positioned close in the embedding space (Goldberg, 2016, p. 365; Pilehvar & Camacho-Collados, 2020, p. 27).

Representations learned by these early word embedding models such as CBOW and GloVe, however, have two shortcomings. First, these models learn for each feature z_u a single vector representation $\boldsymbol{z}_u \in \mathbb{R}^K$ that encodes one information (Ruder, 2019a, p. 74). For models to deduce complex meanings from sequences of tokens, however, several different information types that build on top of each other are likely to be required (e.g. morphological, syntactic, and semantic information) (Peters et al., 2018b; Tenney et al., 2019a). In NLP, therefore, deep neural networks are now being used to learn deep (i.e. multi-layered) representations (Peters et al., 2018a, p. 2233-2234; Ruder, 2019a, p. 74). In deep neural networks, each layer learns one vector representation for a feature (Peters et al., 2018a, p. 2228). Hence, a single feature is represented by several vectors—one vector from each layer. Although it cannot be specified a priori which information is encoded in which hidden layer in a specific model trained on a specific task, research suggests that information encoded in lower layers is less complex and more general whereas information encoded in higher layers is more complex and more task-specific (Yosinski et al., 2014; Tenney et al., 2019a). The representations learned by a deep neural language model thus may, for example, encode morphological information about core textual elements at lower layers, syntactic aspects at middle layers, and semantic information in higher layers (Peters et al., 2018b; Jawahar et al., 2019; Tenney et al., 2019a).

While previously often only the first embedding layer $\boldsymbol{E}$ of a deep neural network had been initialized with pretrained word embeddings (e.g. from Skip-gram or GloVe), the standard procedure in NLP now is to pretrain an entire deep neural network in order to learn model parameters on the basis of which representation vectors' elements can be computed through the layers (Pilehvar & Camacho-Collados, 2020, p. 74-75; Ruder, 2019a, p. 74).Then, the model (including its pretrained parameters) is used as the starting point for training on the target task of interest (Ruder, 2019a, p. 64, 77). In general, this procedure, in which pretrained models are transferred from a source pretraining task to the target task of interest is called sequential transfer learning (Ruder, 2019a, p. 45) and will be introduced in more detail in Section 2.3 below.

The second issue with the early word embedding models is that by representing each feature z_u with a single vector $\boldsymbol{z}_u$, distinct meanings of one feature are fused into one representation vector (Pilehvar & Camacho-Collados, 2020, p. 60). This is known as the meaning conflation deficiency (Pilehvar & Camacho-Collados, 2020, p. 60). For example, the term *'class'* can denote a group of people with a similar status but also a course taken at an educational institution (Princeton University, 2010). A single vector is likely to blend these two meanings (having the effect that the vector will be located somewhere between the two different meanings in space) (Schütze, 1998, p. 102). In recent years, this issue has

been addressed in NLP by learning contextualized representations (Pilehvar & Camacho-Collados, 2020, p. 74). Contextualized representations account for the observation that the (exact) meaning of a token arises from its context (Pilehvar & Camacho-Collados, 2020, p. 82). A contextualized representation is a representation of a token a_t (not a feature z_u) and is a function of the tokens that precede and/or proceed token a_t (Pilehvar & Camacho-Collados, 2020, p. 82). Hence, two identical tokens that occur in different contexts, will have a different representation. As contextualized representations capture information from surrounding tokens, they also allow encoding information on syntactic or semantic dependencies between tokens (Pilehvar & Camacho-Collados, 2020, p. 74). The representation for the token *'it'* in the sentence *'The party leader made a suggestion and it was immediately adopted.'* hence can encode its reference to *'suggestion'* (Alammar, 2018a). Deep and contextualized representations are learned by deep RNNs (Elman, 1990) (and derived architectures such as deep long short-term memory (LSTM) models (Hochreiter & Schmidhuber, 1997)) and the Transformer (Vaswani et al., 2017). Currently, especially Transformer-based models are widely used to learn deep contextualized representations.

To wrap up and to sum up: Because they are composed of a stack of nonlinear functions that map from one vector representation to the next, deep learning models tend to have a high capacity (Goodfellow et al., 2016, p. 5, 168). This is, they can approximate a large variety of complex functions (Goodfellow et al., 2016, p. 110). On less complex data structures, large deep learning models may risk overfitting and conventional machine learning approaches with lower expressivity may be more suitable. The ability to express complicated functions, the ability to automatically learn multi-layered representations, and the ability to encode information on dependencies between tokens and to encode context-dependent meanings of tokens, however, seem important when working with text data: In most areas of NLP, bag-of-words-based representations coupled with conventional machine learning does not constitute the state-of-the-art for some time now (Goldberg, 2016). Moreover, models that learn deep and contextualized representations tend to generalize better across a wide spectrum of specific target tasks compared to the one-layer representations from early word embedding architectures (see e.g. McCann et al., 2018). Consequently, over the last two decades, the field of NLP moved from sparse, high-dimensional representations of single textual features and documents to dense, relatively low-dimensional, deep, and contextualized representations. Today, models that can learn deep contextualized representations and that can be transferred (and then put to use) across learning tasks and domains are at the heart of many modern NLP approaches (Bommasani et al., 2021). How and why models are transferred across tasks and domains is described in the next section.

2.3 Transfer Learning

The classic approach in supervised learning is to have a training data set containing a large number of annotated instances, $(\boldsymbol{x}_i, y_i)_{i=1}^{N}$, that are provided to a model that learns a function relating the $\boldsymbol{x}_i$ to the y_i (Ruder, 2019a, p. 2). If the train and test data instances have been drawn from the same distribution over the feature space, the trained model can be expected to make accurate predictions for the test data, i.e. to generalize well (Ruder, 2019a, p. 42). Given another task (i.e. another set of labels to learn and thus another function f to approximate) or another domain (e.g. another set of documents with a different thematic focus and thus another distribution over the feature space), the standard supervised learning procedure would be to sample and create a new training data set for this new task and domain (Ruder, 2019a, p. 42). This is, for each new task and domain, a new model is trained from the start (Ruder, 2019a, p. 42). There is no transferal of already existing, potentially relevant, and useful information from other domains or tasks to the task at hand (Ruder, 2019a, p. 2). Trained supervised learning models thus are not good at generalizing to data exhibiting characteristics different from the data they have been trained on (Ruder, 2019a, p. 2). Moreover, the (manual) labeling of thousands to millions of training instances for each new task makes supervised learning highly resource intensive and prohibitively costly to be applied for all potentially useful and interesting tasks (Ruder, 2019a, p. 2-3). In situations in which the number of annotated training examples is restricted or the researcher lacks the resources to label a sufficiently large number of training instances classic supervised learning fails (Ruder, 2019a, p. 2-3). This is where transfer learning comes in. Transfer learning refers to a set of learning procedures in which knowledge, that has been obtained by training on a source task in a source domain, is transferred to the learning process of the target task in a task domain, where either the target task is not the same task as the source task or the target domain is not the same as the source domain (Pan & Yang, 2010, p. 1347; Ruder, 2019a, p. 42-43).

2.3.1 A Taxonomy of Transfer Learning

Ruder (2019a, p. 44-46) provides a taxonomy of transfer learning scenarios in NLP: In transductive transfer learning, source and target domains differ, and annotated training examples are typically only available for the source domain (Ruder, 2019a, p. 46). Here, knowledge is transferred across domains (domain adaptation); or—if source and target documents are from different domains in the sense that they are from different languages—knowledge is transferred across languages (cross-lingual learning) (Ruder, 2019a, p. 46). In inductive transfer learning, source and target tasks differ, but the researcher has at least some labeled training samples of the target task (Ruder, 2019a, p. 46). In this setting, tasks can be learned simultaneously (multitask learning) or sequentially (sequential transfer learning) (Ruder, 2019a, p. 46).

2.3.2 Sequential Transfer Learning

In this article, the focus is on sequential transfer learning, which is a frequently employed type of transfer learning. In sequential transfer learning, the source task differs from the target task, and training is conducted in a sequential manner (Ruder, 2019a, p. 63). Two stages are distinguished: First, a model is pretrained on a source task (pretraining phase) (Ruder, 2019a, p. 64). Subsequently, the knowledge gained in the pretraining phase is transferred to the learning process on the target task (adaptation phase) (Ruder, 2019a, p. 64). In NLP, the *knowledge* that is transferred are typically the parameter values learned during training the source model (Ruder, 2019a, p. 43). The model parameters define how token representations are computed from inputs and define how token representations are transformed into updated versions of token representations in deeper layers.

The common procedure in sequential transfer learning in NLP is to select a source task that is likely to learn a model that constitutes a widely applicable language representation tool and thus is likely to provide an effective input for a large spectrum of specific target tasks (Ruder, 2019a, p. 64). Because many training instances are required to learn such a general model, training a source model in the sequential transfer learning setting is highly expensive (Ruder, 2019a, p. 64). Yet adapting a once pretrained model to a target task is often fast and cheap as transfer learning procedures require only a small proportion of the annotated target data required by standard supervised learning procedures in order to achieve the same level of performance (Howard & Ruder, 2018, p. 334). In Howard & Ruder (2018, p. 334), for example, training the deep learning model ULMFiT from scratch on the target task requires 5 to 20 times more labeled training examples to reach the same error rate than when adapting a pretrained ULMFiT model to the target task. Thus, the common sequential transfer learning procedure in applied research projects is to take a model that has already been pretrained and then to adapt it to the target task of interest.

When a model whose parameter values have been learned by training on a suitable task and data set is used as a pretrained input to the training process on a target task, this is likely to increase the prediction performance on the target task—even if only few target training instances are used (Howard & Ruder, 2018, p. 334-335; Ruder, 2019a, p. 65).[8] Note that the smaller the target task training data set size, the more salient the pretrained model parameters become. When decreasing the number of target task training set instances, the prediction performance of deep neural networks that are trained from scratch on the target task declines (Howard & Ruder, 2018, p. 334). For models that are used in a transfer learning setting and are pretrained on a source task before being trained on the target task, prediction performance levels also decline; yet performance levels decrease more slowly and more slightly (Howard & Ruder, 2018, p. 334). Hence, for medium-sized or small training data sets, the prediction performance increase achieved by transfer learning is likely to be

[8]Suitable here means that the task and data set are conducive to learning a well-generalizing language representation model. (See Section 2.3.3 below.)

larger than for very large training data sets (Howard & Ruder, 2018, p. 334-335).

2.3.3 Pretraining

In order to learn a general, all-purpose language representation model, that is relevant for a wide spectrum of tasks within an entire discipline, two things are required: (1) a pretraining data set that contains a large number of training samples and is representative of the feature distribution studied across the discipline and (2) a suitable pretraining task (Ruder, 2018; Ruder, 2019a, p. 65).

The most fundamental pretraining approaches in NLP are self-supervised (Ruder, 2019a, p. 68). Among these, a very common pretraining task is language modeling (Bengio et al., 2003). A language model assigns a probability to the next token given the sequence of previous tokens (Bengio et al., 2003, p. 1138). As the probability for a sequence of T tokens, $P(a_1, \ldots, a_t, \ldots, a_T)$, can be computed as

$$P(a_1, \ldots, a_t, \ldots, a_T) = \prod_{t=1}^{T} P(a_t | a_1, \ldots, a_{t-1}) \tag{2.3}$$

or as

$$P(a_1, \ldots, a_t, \ldots, a_T) = \prod_{t=T}^{1} P(a_t | a_T, \ldots, a_{t+1}) \tag{2.4}$$

language modeling involves predicting the conditional probability of token a_t given all its preceding tokens, $P(a_t|a_1, \ldots, a_{t-1})$, or implicates predicting the conditional probability of token a_t given all its succeeding tokens, $P(a_t|a_T, \ldots, a_{t+1})$ (Bengio et al., 2003, p. 1138; Peters et al., 2018a, p. 2229). A forward language model models the probability in Equation 2.3, a backward language model computes the probability in Equation 2.4 (Peters et al., 2018a, p. 2228-2229). When being trained on a forward and/or backward language modeling task in pretraining, a model learns general structures and aspects of language, such as long-range dependencies, compositional structures, semantics, and sentiment, that are relevant for a wide spectrum of possible target tasks (Howard & Ruder, 2018; Peters et al., 2018b; Ruder, 2018). Hence, language modeling can be considered a well-suited pretraining task (Howard & Ruder, 2018, p. 329-330).[9]

[9]The text corpora that are employed for pretraining vary widely regarding the number of tokens they contain as well as their accessibility (Aßenmacher & Heumann, 2020, p. 3-4). (A detailed and systematic overview of these data sets is provided by Aßenmacher & Heumann (2020).) Most models are trained on a combination of different corpora. Several models (e.g. Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020; Liu et al., 2019) use the English Wikipedia and the BooksCorpus Dataset (Zhu et al., 2015). Many models (e.g. Liu et al., 2019; Radford et al., 2019; Yang et al., 2019; Brown et al., 2020) additionally also use pretraining corpora made up of web documents obtained from crawling the web.

2.3.4 Adaptation: Feature Extraction vs. Fine-Tuning

There are two basic ways how to implement the adaptation phase in transfer learning: feature extraction vs. fine-tuning (Ruder, 2019a, p. 77). In a feature extraction approach, the parameters learned in the pretraining phase are frozen and not altered during adaptation (Ruder, 2019a, p. 77). In fine-tuning, on the other hand, the pretrained parameters are updated in the adaptation phase (Ruder, 2019a, p. 77).

An example of a feature extraction approach is ELMo (Peters et al., 2018a). After pretraining, ELMo is applied without further adaptations on each target task sequence to produce for each token in each sequence three layers of representation vectors (Peters et al., 2018a, p. 2229-2230). For each token, the representation vectors then are extracted to serve as the input for a new target task-specific model that learns a linear combination of the three layers of representation vectors (Peters et al., 2018a, p. 2229-2230). Here, only the weights of the linear model but not the parameters extracted from the pretrained model are trained (Peters et al., 2018a, p. 2229-2230).

In fine-tuning—which now is the standard adaptation procedure in sequential transfer learning (Ruder, 2021)—typically the same model architecture used in pretraining is also used for adaptation (Peters et al., 2019, p. 8). Merely a task-specific output layer is added to the model (Peters et al., 2019, p. 8). The parameters learned in the pretraining phase serve as initializations for the model in the adaptation phase (Ruder, 2019a, p. 77). When training the model on the target task, the gradients are allowed to backpropagate to the pretrained parameters and thus induce changes on these pretrained parameters (Ruder, 2019a, p. 77). In contrast to the feature extraction approach, the pretrained parameters hence are allowed to be fine-tuned to capture task-specific adjustments (Ruder, 2019a, p. 77). When fine-tuning BERT on a target task, for example, a target task-specific output layer is put on top of the pretraining architecture (Devlin et al., 2019, p. 4173). Then the entire architecture is trained, meaning that *all* parameters are updated (Devlin et al., 2019, p. 4173).[10]

Note that both ELMo and BERT learn deep and contextualized representations. In ELMo there are three layers of representations and in BERT there are 12 or 24 layers depending on the selected model size (Peters et al., 2018a, p. 2230; Devlin et al., 2019, p. 4173). The representations are contextualized because the neural network architectures, that ELMo and BERT are based on, learn representations for tokens that encode information from the textual context a token is embedded in. ELMo is based on a deep bidirectional LSTM

[10] A central parameter in fine-tuning is the learning rate η with which the gradients are updated during training on the target task (see Equation 2.22 in Appendix 2.A). Too much fine-tuning (i.e. a too high learning rate) can lead to catastrophic forgetting—a situation in which the parameters learned during pretraining are overwritten and therefore forgotten when fine-tuning the model (Kirkpatrick et al., 2017; Howard & Ruder, 2018, p. 330-332). A too careful fine-tuning scheme (i.e. a too low learning rate), in contrast, may lead to a very slow convergence process (Howard & Ruder, 2018, p. 330-332). In general, it is recommended that the learning rate should be lower than the learning rate used in pretraining such that the parameters learned during pretraining are not altered too much (Ruder, 2019a, p. 78).

and BERT is based on elements from the Transformer architecture (which is introduced in Section 2.4) (Peters et al., 2018a, p. 2228-2230; Devlin et al., 2019, p. 4173).

2.3.5 Cross-Lingual Learning

One reason for having only a limited amount of target task training data (or limited resources for labeling target task training data) could be that the target task texts are in a language other than English.[11] In this case, transfer learning offers two solutions. One solution is to implement sequential transfer learning with a model that has been pretrained on a monolingual corpus in the target language. Examples of non-English pretrained language representation models are, for example, the French CamemBERT (Martin et al., 2020), the Vietnamese PhoBERT (Nguyen & Tuan Nguyen, 2020), or German (dbmdz 2021) and Chinese BERT models (Devlin, 2019). An overview of language-specific pretrained models is provided by the website https://bertlang.unibocconi.it/ which is introduced in Nozza et al. (2020).

If, however, no monolingual pretrained model exists for the target language and/or no labeled target task training data in the language of interest are available, then another type of transfer learning—namely cross-lingual learning—provides a possible solution. In cross-lingual learning, source and target domains differ in the sense that source and target documents come from different languages (Ruder, 2019a, p. 45). Moreover, labeled training data are usually only available for the source language but not the target language (Ruder, 2019b).

Cross-lingual learning involves the learning of cross-lingual representations that allow the transferal of information across languages (Ruder, 2019b). This can be achieved by pretraining a model on text data from multiple languages (see e.g. the pretraining processes of the multilingual models mBERT (Devlin, 2019), XLM-R (Conneau et al., 2020), and mT5 (Xue et al., 2021)).

If only unlabeled but no labeled data in the target language are available, one way to conduct cross-lingual learning is as follows: (1) Cross-lingual representations are learned. (2) The labeled training examples in the source language are used to learn task-specific parameters that map from the cross-lingual representations to the task-specific outputs. (3) The pretrained model (containing cross-lingual representations plus task-specific parameters) is directly applied—without any further adaptation step—on data in the target language to make predictions for target language data (Ruder, 2019b).[12] So far, research suggests that the prediction performance of pretrained monolingual models on downstream target tasks tends to exceed the performance of multilingual models (Rust et al., 2021). But if

[11] I am grateful to one of the reviewers for pointing this out to me.

[12] Although this last predictive step does not match the definition of zero-shot learning given in Section 2.3.6 below, it is often called zero-shot cross-lingual transfer (see e.g. Ruder, 2019b; Wu & Dredze, 2019; Nozza et al., 2020).

the multilingual pretraining corpus contains substantial amounts of text in the target task language and if target language-adapted tokenizers are used, the performance differences between monolingual and multilingual models can become small (Rust et al., 2021). For more information on cross-lingual learning see Ruder (2019b).

2.3.6 Zero-Shot Learning

A further strand of research within NLP aims at the development and pretraining of models that are able to make accurate predictions for a wide range of different target tasks without having been explicitly fine-tuned on those target tasks (Radford et al., 2019; Yin et al., 2019; Brown et al., 2020). The aim is to have a model that performs well on a task it has not conducted before (Davison, 2020b). This general idea is often referred to as zero-shot learning (but the precise definition of the term varies across research papers) (Davison, 2020b). Here, following the *Definition-Wild* of Yin et al. (2019, p. 3915) zero-shot learning is considered a setting in which a model makes predictions for target task texts without having seen task-specific pairs $(\boldsymbol{x}_i, y_i)$ and without having seen the space of task-specific labels (e.g. $\mathcal{Y} = \{positive, negative\}$) during training.[13] One work in this context that has generated attention far beyond the boundaries of the field of NLP is the GPT-3 model (Brown et al., 2020). (For a note on GPT-3 see Appendix 2.B.) Zero-shot learning partly can achieve surprisingly high prediction performances on target tasks. Thus far, however, performance levels tend to be lower compared to state-of-the-art fine-tuned models (see e.g. the zero-shot GPT-3 in Brown et al., 2020).

2.4 (Self-)Attention and the Transformer

As the family of RNNs is made to process sequential input data, they seem the natural models of choice for processing sequences of textual tokens. The problem of recurrent architectures, however, is that they model dependencies by sequentially propagating through the positions of the input sequence. Thus, the longer the distance between the tokens, the less well the dependency tends to be learned (Goodfellow et al., 2016, p. 396-399). A solution to this problem is provided by the attention mechanism, which first has been introduced for Neural Machine Translation (NMT) by Bahdanau et al. (2015) and was refined by Luong et al. (2015). The attention mechanism allows to model dependencies between tokens irrespective of the distance between them (Vaswani et al., 2017, p. 5999). The Transformer is a deep learning architecture that is based on attention mechanisms (Vaswani et al., 2017, p. 5999). This section first explains the attention mechanism and then introduces the Transformer.

[13] In in-context-learning, no adaptation to the target task in the form of gradient updates is performed, but the model is presented with task examples and instructions in natural language (Brown et al., 2020, p. 4).

2.4.1 The Attention Mechanism

The common task encountered in NMT is to translate a sequence of T tokens in language A, $(a_1, \ldots, a_t, \ldots, a_T)$, to a sequence of S tokens in language O, $(o_1, \ldots, o_s, \ldots, o_S)$ (Sutskever et al., 2014, p. 3106). The classic architecture to solve this task is an encoder-decoder structure (see Figure 2.2) (Sutskever et al., 2014, p. 3106). In general, an encoder transforms input data into a representation and a decoder conducts the reverse operation: The decoder produces data output from an encoded representation. In the early NMT articles, the encoder maps the input tokens $(a_1, \ldots, a_t, \ldots, a_T)$ into a single context vector $\boldsymbol{c}$ of fixed dimensionality that is then provided to the decoder that generates the sequence of translated output tokens $(o_1, \ldots, o_s, \ldots, o_S)$ from $\boldsymbol{c}$ (Sutskever et al., 2014, p. 3106).

Another characteristic of early NMT articles is that encoder and decoder are recurrent models (Sutskever et al., 2014, p. 3106) (on recurrent models see Appendix 2.A.3). Hence, the encoder processes each input embedding $\boldsymbol{z}_{[a_t]}$ step by step. The hidden state at time step t, $\boldsymbol{h}_t$, is a nonlinear function (here denoted by σ) of the previous hidden state, $\boldsymbol{h}_{t-1}$, and input embedding $\boldsymbol{z}_{[a_t]}$ (Cho et al., 2014, p. 1725):

$$\boldsymbol{h}_t = \sigma(\boldsymbol{h}_{t-1}, \boldsymbol{z}_{[a_t]}) \tag{2.5}$$

The last encoder hidden state, $\boldsymbol{h}_T$, corresponds to context vector $\boldsymbol{c}$ that then is passed on to the decoder which—given the information encoded in $\boldsymbol{c}$—produces a variable-length sequence output $(o_1, \ldots, o_s, \ldots, o_S)$ (Cho et al., 2014, p. 1725). The decoder also operates in a recurrent manner: Based on the current decoder hidden state $\boldsymbol{h}_s$, one output token o_s is predicted at one time step (Cho et al., 2014, p. 1725).[14] In contrast to the encoder, the hidden state of the decoder at time step s, $\boldsymbol{h}_s$, is not only a function of the previous hidden state $\boldsymbol{h}_{s-1}$ but also the embedding of the previous output token $\boldsymbol{z}_{[o_{s-1}]}$, and context vector $\boldsymbol{c}$ (see also Figure 2.2) (Cho et al., 2014, p. 1725):

$$\boldsymbol{h}_s = \sigma(\boldsymbol{h}_{s-1}, \boldsymbol{z}_{[o_{s-1}]}, \boldsymbol{c}) \tag{2.6}$$

A problem with this traditional encoder-decoder structure is that all the information about the input sequence—regardless of the length of the input sequence—is captured in a single context vector $\boldsymbol{c}$ (Bahdanau et al., 2015, p. 1).

The attention mechanism resolves this problem. In the attention mechanism, at each time step, the decoder can attend to, and thus derive information from, all encoder-produced input hidden states when computing its hidden state $\boldsymbol{h}_s$ (see Figure 2.3). More precisely, the decoder hidden state at time point s, $\boldsymbol{h}_s$, is a function of the initial decoder hidden state $\tilde{\boldsymbol{h}}_s$, the previous output token $\boldsymbol{z}_{[o_{s-1}]}$, and an output token-specific context vector $\boldsymbol{c}_s$

[14] More precisely, in Cho et al. (2014, p. 1725), the decoder's prediction for the next output token is a function of the current decoder hidden state $\boldsymbol{h}_s$, the embedding of the previous output token $\boldsymbol{z}_{[o_{s-1}]}$, and context vector $\boldsymbol{c}$. The decoder produces a probability distribution over the vocabulary signifying the next predicted output token: $P(o_s|o_1, \ldots, o_{s-1}, \boldsymbol{c}) = \sigma_o(\boldsymbol{h}_s, \boldsymbol{z}_{[o_{s-1}]}, \boldsymbol{c})$ (Cho et al., 2014, p. 1725).

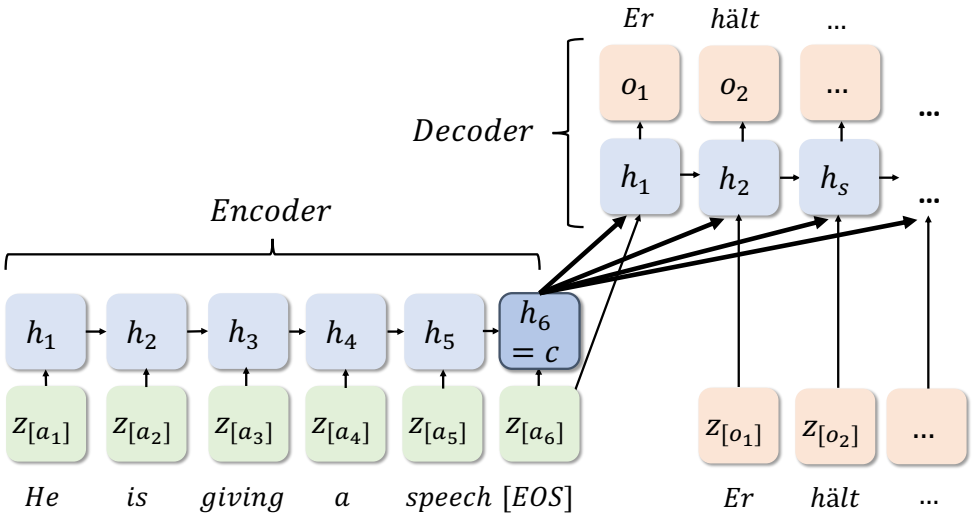


Figure 2.2: **Encoder-Decoder Architecture.** Encoder-decoder structure in neural machine translation. In this example, the six token input sentence $(He, is, giving, a, speech, [EOS])$ is translated to German: *(Er, hält, eine, Rede, [EOS])*. The end-of-sentence symbol $[EOS]$ is used to signal to the model the end of a sentence. The recurrent encoder processes one input embedding $\boldsymbol{z}_{[a_t]}$ at a time and updates the input hidden state $\boldsymbol{h}_t$ at each time step. The last encoder hidden state $\boldsymbol{h}_6$ serves as context vector $\boldsymbol{c}$ that captures all the information from the input sequence. The decoder generates one translated output token at a time. Each output hidden state $\boldsymbol{h}_s$ is a function of the preceding hidden state $\boldsymbol{h}_{s-1}$, the preceding predicted output token embedding $\boldsymbol{z}_{[o_{s-1}]}$, and context vector $\boldsymbol{c}$.

(Luong et al., 2015, p. 1414).[15]

$$\boldsymbol{h}_s = \sigma(\tilde{\boldsymbol{h}}_s, \boldsymbol{z}_{[o_{s-1}]}, \boldsymbol{c}_s) \tag{2.7}$$

Note that now at each time step there is a context vector $\boldsymbol{c}_s$ that is specific to the sth output token (Bahdanau et al., 2015, p. 3). The attention mechanism rests in the computation of $\boldsymbol{c}_s$, which is a weighted sum over the input hidden states $(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_t, \ldots, \boldsymbol{h}_T)$ (Bahdanau et al., 2015, p. 3):

$$\boldsymbol{c}_s = \sum_{t=1}^{T} \alpha_{s,t} \boldsymbol{h}_t \tag{2.8}$$

The weight $\alpha_{s,t}$ is computed as

$$\alpha_{s,t} = \frac{exp(score(\tilde{\boldsymbol{h}}_s, \boldsymbol{h}_t))}{\sum_{t^*=1}^{T^*} exp(score(\tilde{\boldsymbol{h}}_s, \boldsymbol{h}_{t^*}))} \tag{2.9}$$

[15]Note that Equation 2.7 blends the specifications of Luong et al. (2015, p. 1414) and Bahdanau et al. (2015, p. 3). Luong et al. (2015, p. 1414) do not include $\boldsymbol{z}_{[o_{s-1}]}$. Luong et al. (2015) also do not explicitly state how they compute $\tilde{\boldsymbol{h}}_s$. Bahdanau et al. (2015, p. 3) use $\boldsymbol{h}_{s-1}$ instead of $\tilde{\boldsymbol{h}}_s$ to represent the state of the decoder at s (or rather at the moment just before producing the sth output token).

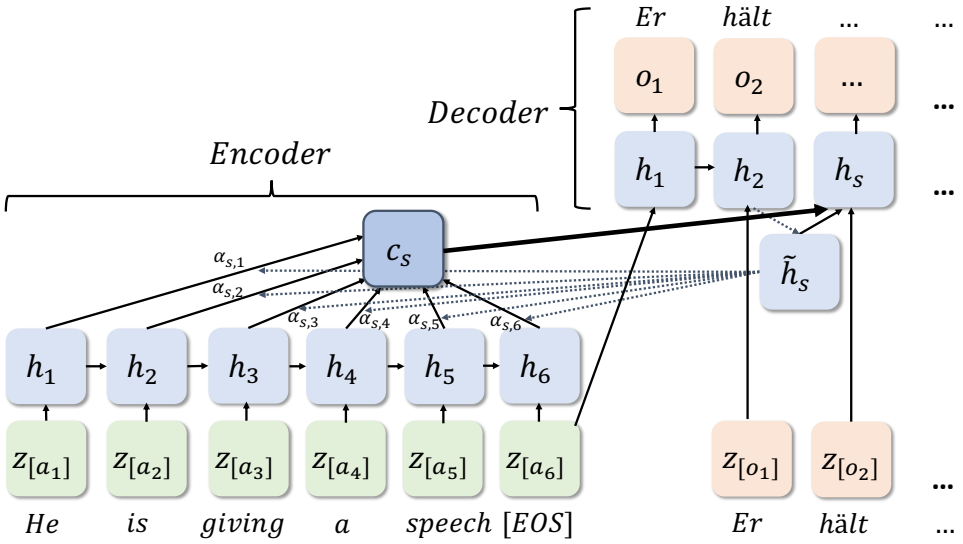


Figure 2.3: **Attention in an Encoder-Decoder Architecture.** Visualization of the attention mechanism in an encoder-decoder structure at time step s. In the attention mechanism, at each time step, i.e. for each output token, there is a token-specific context vector $\boldsymbol{c}_s$. $\boldsymbol{c}_s$ is computed as the weighted sum over all input hidden states $(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_6)$. The weights are $(\alpha_{s,1}, \ldots, \alpha_{s,6})$. $\alpha_{s,1}$ results from a scoring function that captures the similarity between the sth output token, as represented by the initial output hidden state $\tilde{\boldsymbol{h}}_s$, and input token hidden state $\boldsymbol{h}_1$.

where *score* is a scoring function assessing the compatibility between output token representation $\tilde{\boldsymbol{h}}_s$ and input token representation $\boldsymbol{h}_t$ (Luong et al., 2015, p. 1414). *score* could be, for example, the dot product of $\tilde{\boldsymbol{h}}_s$ and $\boldsymbol{h}_t$ (Luong et al., 2015, p. 1414). The attention weight $\alpha_{s,t}$ is a measure of the degree of alignment of the tth input token, represented by $\boldsymbol{h}_t$, with the sth output token, represented as $\tilde{\boldsymbol{h}}_s$ (Bahdanau et al., 2015, p. 3-4). Input hidden states that do not match with output token representation $\tilde{\boldsymbol{h}}_s$ receive a small weight such that their contribution vanishes, whereas input hidden states that are relevant to output token $\tilde{\boldsymbol{h}}_s$ receive high weights, thereby increasing their contribution (Alammar, 2018c). Hence, $\boldsymbol{c}_s$ considers all input hidden states and especially attends to those input hidden states that match with the current output token. As context vector $\boldsymbol{c}_s$ is constructed for each output token based on a weighted sum of *all* input hidden states, the attention architecture allows for modeling dependencies between tokens irrespective of their distance (Vaswani et al., 2017, p. 5999).

2.4.2 The Transformer

The original articles on attention use recurrent architectures in the encoder and decoder. The sequential nature of recurrent models implies that within each training example sequence each token has to be processed one after another—a computationally not efficient

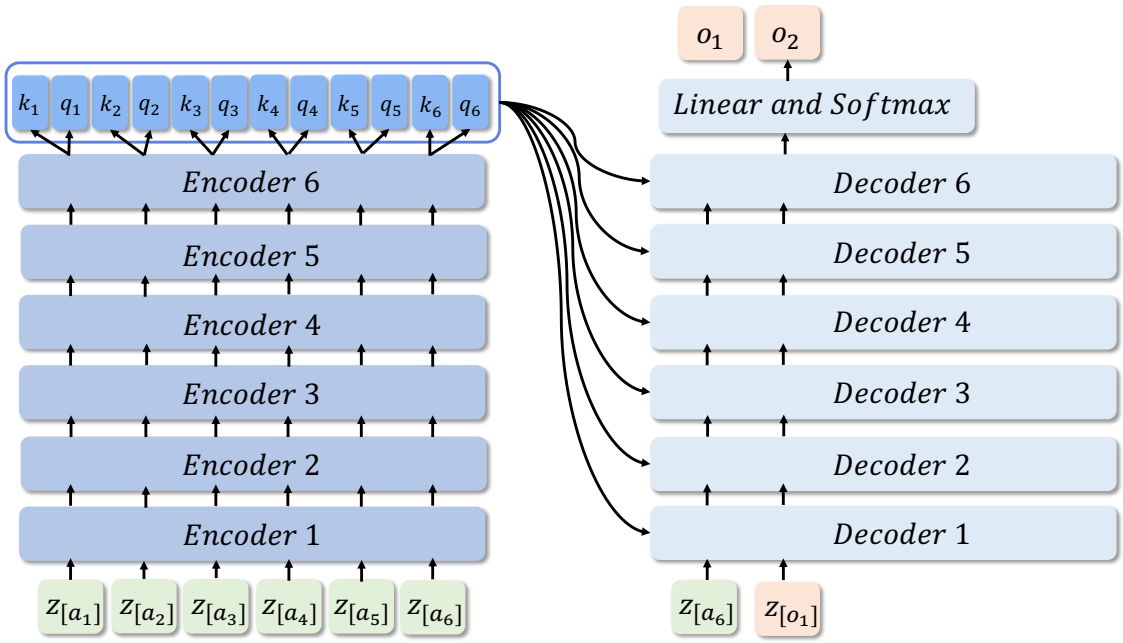


Figure 2.4: **Transformer Architecture.** In the original article by Vaswani et al. (2017), the Transformer is made up of a stack of six encoders proceeded by a stack of six decoders. In contrast to recurrent architectures where each input token is handled one after another, a Transformer encoder processes the entire set of input token representations in parallel (Vaswani et al., 2017, p. 5999). Here, the input embeddings are $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_6]})$. The sixth encoder passes the key and query vectors of the input tokens, $(\boldsymbol{k}_1, \boldsymbol{q}_1, \ldots, \boldsymbol{k}_6, \boldsymbol{q}_6)$, to each of the decoders. These key and query vectors from the last encoder are processed in each decoder's encoder-decoder attention layer (Vaswani et al., 2017, p. 6002). The Transformer decoders operate in an autoregressive manner, meaning that the stack of decoders processes as an additional input the sequence of previous output tokens (Vaswani et al., 2017, p. 6002). In the visualization here, output tokens are denoted with $(o_1, o_2, \ldots)$ and the decoder predicts output token o_2 given the previous tokens (a_6, o_1) (where a_6 is an end-of-sentence symbol). To predict the tth output token, the hidden state of the last decoder is processed through a linear layer and a softmax layer to produce a probability distribution over the terms in the vocabulary (Vaswani et al., 2017, p. 6002).

strategy (Vaswani et al., 2017, p. 5999). To overcome this inefficiency and to enable parallel processing within training sequences, Vaswani et al. (2017) introduced the Transformer architecture that is built from attention mechanisms. The Transformer consists of a sequence of six encoders followed by a stack of six decoders (see Figure 2.4) (Vaswani et al., 2017, p. 6000).[16] Each encoder consists of two components: a multi-head self-attention layer (to be explained below) and a feedforward neural network (Vaswani et al., 2017, p. 6000). Each decoder also has a multi-head self-attention layer followed by a multi-head encoder-decoder attention layer and a feedforward neural network (Vaswani et al., 2017, p. 6000). Instead of processing each token of each training example one after another, the Transformer encoder takes as an input the whole set of T embeddings for one training example and processes this set of embeddings, $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_t]}, \ldots, \boldsymbol{z}_{[a_T]})$, in parallel (Alammar, 2018b). The T embeddings entering the first encoder are position-aware embeddings (see bottom of Figure 2.5 that provides a visualization of the first Transformer encoder) (Vaswani et al., 2017, p. 6002-6003). A position-aware embedding is the sum of a pure embedding vector and a positional encoding vector (Vaswani et al., 2017, p. 6003). The positional encoding vector contains information on the position of the tth token within the input sequence, thereby making the model aware of token positions (Vaswani et al., 2017, p. 6002-6003).

The first element in a Transformer encoder is the multi-head self-attention layer. In the self-attention layer, the provided input sequence $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_t]}, \ldots, \boldsymbol{z}_{[a_T]})$ attends to itself. Instead of improving the representation of an output token by attending to tokens in the input sequence, the idea of self-attention is to improve the representation of a token a_t by attending to the tokens in the same sequence in which a_t is embedded in (Alammar, 2018b). For example, if *'The company is issuing a statement as it is bankrupt.'* were a sentence to be processed, then the embedding for the token *'it'* that enters the Transformer would not contain any information regarding which other token in the sentence *'it'* is referring to. Is it the company or the statement? In the self-attention mechanism, the representation for *'it'* is updated by attending to—and incorporating information from—other tokens in this sentence (Alammar, 2018b). It, therefore, is to be expected that after passing through the self-attention layers, the representation of *'it'* absorbed some of the representation for *'company'* and so encodes information on the dependency between *'it'* and *'company'* (Alammar, 2018b).

The first operation within a self-attention layer is that each input embedding $\boldsymbol{z}_{[a_t]}$ is transformed into three separate vectors, called key $\boldsymbol{k}_t$, query $\boldsymbol{q}_t$, and value $\boldsymbol{v}_t$ (see Figure 2.5). The key, query, and value vectors are three different projections of the input embedding $\boldsymbol{z}_{[a_t]}$ (Alammar, 2018b). They are generated by matrix multiplication of $\boldsymbol{z}_{[a_t]}$ with three

[16] Note that the number of encoders and decoders, as well as the dimensionality of the input embeddings and the key, query and value vectors (introduced in the following), are Transformer hyperparameters that are simply set by the authors to specific values. Other suitable values could be used instead.

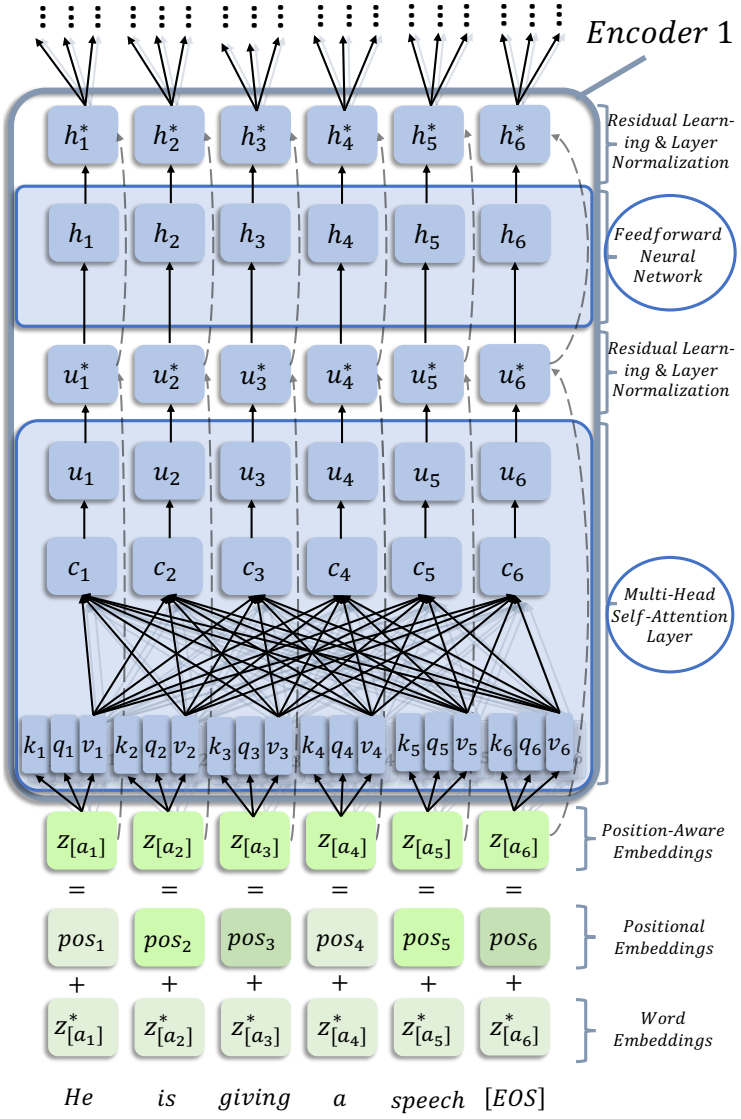


Figure 2.5: **Transformer Encoder Architecture.** This visualization details the processes in the first Transformer encoder. The encoder comprises a multi-head self-attention layer and a feedforward neural network, each followed by residual learning and layer normalization. The first encoder takes as an input position-aware embeddings, $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_6]})$, that are transformed into eight sets of key, query and value vectors. One set is $(\boldsymbol{k}_1, \boldsymbol{q}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{k}_6, \boldsymbol{q}_6, \boldsymbol{v}_6)$. These are processed in the multi-head self-attention layer to produce eight sets of context vectors (one set being $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_6)$). The sets then are concatenated and transformed linearly to become the updated representations $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_6)$. After residual learning and layer normalization, $(\boldsymbol{u}_1^*, \ldots, \boldsymbol{u}_6^*)$ enter the feedforward neural network, whose output—after residual learning and layer normalization—are the updated representations produced by the first Transformer encoder: $(\boldsymbol{h}_1^*, \ldots, \boldsymbol{h}_6^*)$. The representations $(\boldsymbol{h}_1^*, \ldots, \boldsymbol{h}_6^*)$ constitute the input to the next encoder, where they are first transformed to sets of key, query and value vectors.

different weight matrices, $\boldsymbol{W}_k$, $\boldsymbol{W}_q$, and $\boldsymbol{W}_v$ (Vaswani et al., 2017, p. 6002):[17]

$$\boldsymbol{k}_t = \boldsymbol{z}_{[a_t]}\boldsymbol{W}_k \qquad \boldsymbol{q}_t = \boldsymbol{z}_{[a_t]}\boldsymbol{W}_q \qquad \boldsymbol{v}_t = \boldsymbol{z}_{[a_t]}\boldsymbol{W}_v \tag{2.10}$$

Then, for each token a_t, an updated representation (named context vector $\boldsymbol{c}_t$) is computed as a weighted sum over the value vectors of *all* tokens that are in the *same* sequence as token a_t (Vaswani et al., 2017, p. 6000-6002):

$$\boldsymbol{c}_t = \sum_{t^*=1}^{T^*} \alpha_{t,t^*}\boldsymbol{v}_{t^*} \tag{2.11}$$

The attention weight α_{t,t^*} is a function of the similarity between token a_t, represented by $\boldsymbol{q}_t$, and token a_{t^*}, that is represented as $\boldsymbol{k}_{t^*}$:

$$\alpha_{t,t^*} = \frac{exp(score(\boldsymbol{q}_t, \boldsymbol{k}_{t^*}))}{\sum_{t^*=1}^{T^*} exp(score(\boldsymbol{q}_t, \boldsymbol{k}_{t^*}))} \tag{2.12}$$

where *score* is $(\boldsymbol{q}_t\boldsymbol{k}_{t^*}^\top)/\sqrt{|\boldsymbol{k}_{t^*}|}$ (Vaswani et al., 2017, p. 6001). α_{t,t^*} indicates the contribution of token a_{t^*} for the representation of token a_t.[18] Thus, attention vector $\boldsymbol{c}_t$ is calculated as in a basic attention mechanism (see Equations 2.8 and 2.9)—except that the attention now is with respect to the value vectors of the tokens that are part of the same sequence as a_t (see also Figure 2.6).

The self-attention mechanism outlined so far is conducted eight times in parallel (Vaswani et al., 2017, p. 6001-6002). Hence, for each token a_t, eight different sets of query, key and value vectors are generated and there will be not one but eight attention vectors $\{\boldsymbol{c}_{t,1}, \ldots, \boldsymbol{c}_{t,8}\}$ (Vaswani et al., 2017, p. 6001-6002). In doing so, each attention vector can attend to different tokens in each of the eight different representation spaces (Vaswani et al., 2017, p. 6002). For example, in one representation space the attention vector for token a_t may learn syntactic structures and in another representation space the attention vector may attend to semantic connections (Vaswani et al., 2017, p. 6004; Clark et al., 2019). In the example sentence from above, the first attention vector for the token *'it'*, $\boldsymbol{c}_{8,1}$, may have a high attention weight for *'company'*, whereas another attention vector, say $\boldsymbol{c}_{8,3}$, may more strongly attend to *'bankrupt'* (Alammar, 2018b). Because the self-attention mechanism is implemented eight times in parallel and generates eight attention vectors (or heads), the procedure is called multi-head self-attention (Vaswani et al., 2017, p. 6001). The eight attention vectors subsequently are concatenated into a single vector,

[17] Note that in order to follow the notation in Vaswani et al. (2017), vectors (which are indicated by bold letters) are treated as row vectors in the following.

[18] Kobayashi et al. (2020, p. 7057) emphasize that because the attention mechanism is composed of a "*weighted* sum of *linearly transformed vectors*", the raw attention weight α_{t,t^*} is not an all-encompassing indicator of the contribution that the t^*th input token has on the representation of the tth token. To capture attention patterns more comprehensively, they propose norm-based analysis in which not only the attention weight sizes but also the magnitudes of the linearly transformed input vectors are taken into account (Kobayashi et al., 2020).

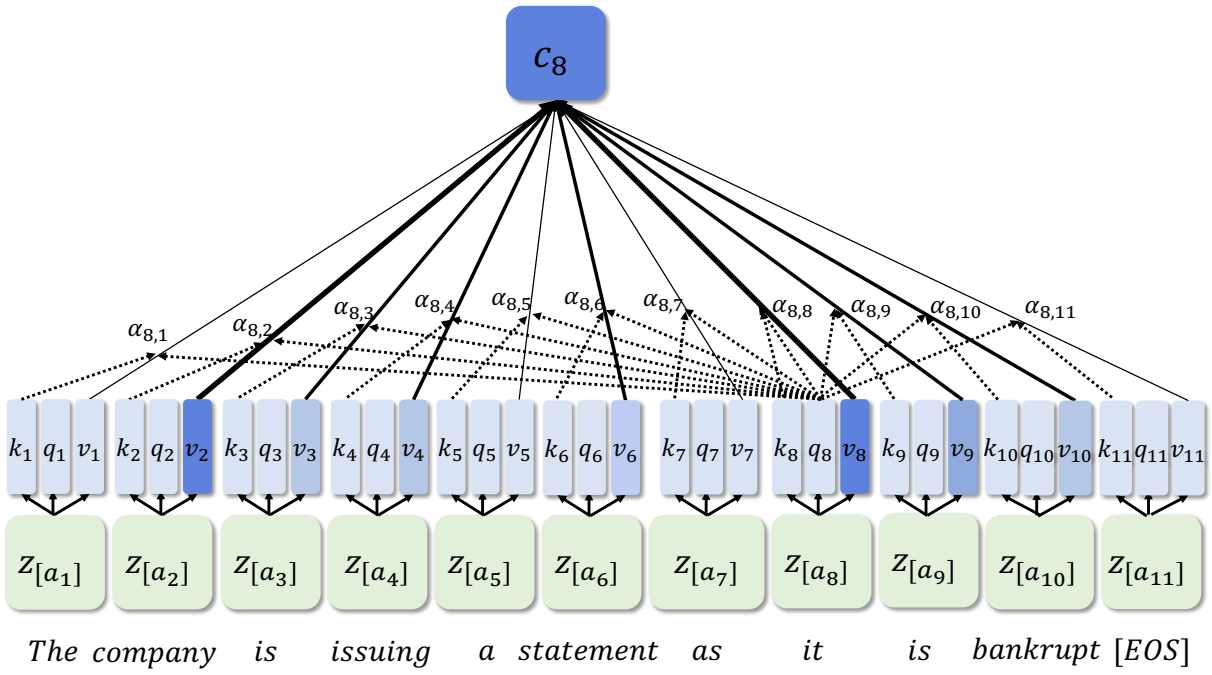


Figure 2.6: **Attention Mechanism in the Transformer.** Illustration of the attention mechanism in the first Transformer encoder for the 8th token (*'it'*) in the example sentence *'The company is issuing a statement as it is bankrupt.'*. The arrows pointing from the value vectors $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{11})$ to context vector $\boldsymbol{c}_8$ are the weights $(\alpha_{8,1}, \ldots, \alpha_{8,t^*}, \ldots, \alpha_{8,11})$. A single weight α_{8,t^*} indicates the contribution of token t^* to the representation of token 8, $\boldsymbol{c}_8$. The larger α_{8,t^*} is assumed to be in this example, the thicker the arrow and the darker the corresponding value vector. The dotted lines symbolize the computation of the weights $(\alpha_{8,1}, \ldots, \alpha_{8,t^*}, \ldots, \alpha_{8,11})$.

$\boldsymbol{c}_t = [\boldsymbol{c}_{t,1}; \ldots; \boldsymbol{c}_{t,8}]$, and multiplied with a corresponding weight matrix $\boldsymbol{W}_0$ to produce vector $\boldsymbol{u}_t$ (Vaswani et al., 2017, p. 6002): $\boldsymbol{u}_t = \boldsymbol{c}_t \boldsymbol{W}_0$. Afterward, $\boldsymbol{u}_t$ is added to $\boldsymbol{z}_{[a_t]}$, thereby allowing for residual learning (He et al., 2015).[19] Then, layer normalization as suggested in Ba et al. (2016) is conducted (Vaswani et al., 2017, p. 6000).[20]

$$\boldsymbol{u}_t^* = LayerNorm(\boldsymbol{u}_t + \boldsymbol{z}_{[a_t]}) \tag{2.13}$$

$\boldsymbol{u}_t^*$ then enters the feedforward neural network with a Rectified Linear Unit (ReLU) activation function (Vaswani et al., 2017, p. 6002)

$$\boldsymbol{h}_t = max(0, \boldsymbol{u}_t^* \boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{2.14}$$

[19] In residual learning, instead of leaning a new representation in each layer, merely the residual change is learned (He et al., 2015). Here $\boldsymbol{u}_t$ can be conceived of as the residual on the original representation $\boldsymbol{z}_{[a_t]}$. Residual learning has been shown to facilitate the optimization of very deep neural networks (He et al., 2015).

[20] In layer normalization, for each training instance, the values of the hidden units within a layer are standardized by using the mean and standard deviation of the layer's hidden units (Ba et al., 2016). Layer normalization reduces training time and enhances generalization performance due to its regularizing effects (Ba et al., 2016).

followed by a residual connection with layer normalization (Vaswani et al., 2017, p. 6000):

$$\boldsymbol{h}_t^* = LayerNorm(\boldsymbol{h}_t + \boldsymbol{u}_t^*) \tag{2.15}$$

$\boldsymbol{h}_t^*$ finally is the representation of token a_t produced by the encoder. It constitutes an updated representation of input embedding $\boldsymbol{z}_{[a_t]}$. Due to the self-attention mechanism, $\boldsymbol{h}_t^*$ is a function of the other tokens in the same sequence and thus captures context-dependent information. Hence, $\boldsymbol{h}_t^*$ is a contextualized representation of token a_t. The same token in another sequence would obtain another token representation vector.

The entire sequence of representations, $(\boldsymbol{h}_1^*, \ldots, \boldsymbol{h}_t^*, \ldots, \boldsymbol{h}_T^*)$, that is produced as the encoder output, serves as the input for the next encoder that generates eight sets of query, key, and value vectors from each representation $\boldsymbol{h}_t^*$ to implement multi-head self attention and to finally produce an updated set of representations, $(\boldsymbol{h}_1^*, \ldots, \boldsymbol{h}_t^*, \ldots, \boldsymbol{h}_T^*)^*$, that are passed to the next encoder and so on. The last encoder from the stack of encoders passes the key and value vectors from its produced sequence of updated representations to each encoder-decoder multi-head attention layer in each decoder (see Figure 2.4) (Vaswani et al., 2017, p. 6002).

Except for the encoder-decoder attention layer in which the decoder pays attention to the encoder input, the architecture of each decoder is largely the same as those of the encoders (Vaswani et al., 2017, p. 6000). Note, however, that the stack of decoders operates in an autoregressive manner (Vaswani et al., 2017, p. 5999). This is, when making the prediction for the next output token o_s, the decoders have access to and process the sequence of previous output tokens, $(a_T, o_s, \ldots, o_{s-1})$, as additional inputs (see Figure 2.4) (Vaswani et al., 2017, p. 5999). In order to ensure that the decoders are autoregressive, self-attention in each decoder is masked, meaning that the attention vector for output token o_s can only attend to output tokens preceding token o_s (Vaswani et al., 2017, p. 6000). To predict an output token, the hidden state of the last decoder is handed to a linear and softmax layer to produce a probability distribution over the vocabulary (Vaswani et al., 2017, p. 6002).

2.5 Transfer Learning with Transformer-Based Models

Taken together, the Transformer architecture in combination with transfer learning literally transformed the field of NLP (Bommasani et al., 2021, p. 5). After the introduction of the Transformer by Vaswani et al. (2017), several models for transfer learning that included elements of the Transformer were developed (e.g. Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Clark et al., 2020; Lan et al., 2020; Raffel et al., 2020). These models and their derivatives significantly outperformed previous state-of-the-art models.

An important step within these developments was the introduction of BERT (Devlin et al., 2019). By establishing new state-of-the-art performance levels for eleven NLP tasks, BERT

demonstrated the power of transfer learning (Bommasani et al., 2021, p. 5). The introduction of BERT finally paved the way to a new transfer learning-based mode of learning in which it is common to use an already pretrained language model and adapt it to a specific target task as needed (Alammar, 2018a; Bommasani et al., 2021, p. 5). Simultaneously with and independently of BERT, a wide spectrum of Transformer-based models for transfer learning have been developed. This section first introduces BERT and then provides an overview of further models.

2.5.1 BERT

BERT consists of a stack of Transformer encoders and comes in two different model sizes (Devlin et al., 2019, p. 4173): $\text{BERT}_{\text{BASE}}$ consists of 12 stacked Transformer encoders. In each encoder, there are 12 attention heads in the multi-head self-attention layer. The dimensionality of the input embeddings and the updated hidden vector representations is 768. $\text{BERT}_{\text{LARGE}}$ has 24 Transformer encoders with 16 attention heads and a hidden vector size of 1024.[21] As in the original Transformer, the first BERT encoder takes as an input a sequence of embedded tokens, $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_t]}, \ldots, \boldsymbol{z}_{[a_T]})$, processes the embeddings in parallel through the self-attention layer and the feedforward neural network to generate a set of updated token representations, $(\boldsymbol{h}_1^*, \ldots, \boldsymbol{h}_t^*, \ldots, \boldsymbol{h}_T^*)$, that are then passed to the next encoder that also generates updated representations to be passed to the next encoder and so on until the representations finally enter output layers for prediction (Alammar, 2018a).

The authors inventing BERT sought to tackle a disadvantage of the classic language modeling pretraining task (see Equations 2.3 and 2.4), namely that it is strictly unidirectional (Devlin et al., 2019, p. 4171). A forward language model predicts the probability for the next token a_t given the so far predicted tokens, $P(a_t|a_1, \ldots, a_{t-1})$. Here, the model can only access information from the preceding tokens $(a_1, \ldots, a_{t-1})$ but not from the following tokens $(a_{t+1}, \ldots, a_T)$.[22] The same is true for a backward language model in which the next token is predicted given all its following tokens, $P(a_t|a_T, \ldots, a_{t+1})$. A backward language model can only operate on, and capture information from, succeeding tokens (Yang et al., 2019, p. 5753). Assuming that a representation of token a_t from a bidirectional model that simultaneously can attend to preceding and succeeding tokens may constitute a better representation of token a_t than a representation stemming from a unidirectional language model, the authors of BERT invented an adapted variant of the traditional language modeling pretraining task, named masked language modeling, to learn deep contextualized

[21] In the feedforward neural networks, Devlin et al. (2019, p. 4183) employ the Gaussian Error Linear Unit (GELU) (Hendrycks & Gimpel, 2016) instead of the ReLU activation function used in the original Transformer. This change in the activation function has also been used for the OpenAI GPT (Radford et al., 2018). $\text{BERT}_{\text{BASE}}$ has 110 million parameters. $\text{BERT}_{\text{LARGE}}$ has 340 million parameters.

[22] In a self-attention mechanism this means that the context vector for token a_t can merely attend to, and hence can only incorporate information from, the representations of preceding but not from succeeding tokens (Devlin et al., 2019, p. 4171).

representations that are bidirectional (Devlin et al., 2019, p. 4171-4172).[23]

To conduct the masked language modeling task in the pretraining process of BERT, in each input sequence, 15% of the input embeddings are selected at random (Devlin et al., 2019, p. 4174, 4183). The selected tokens are indexed as $(1, \ldots, q, \ldots, Q)$ here. 80% of the Q selected tokens will be replaced by the *'[MASK]'* token (Devlin et al., 2019, p. 4174). 10% of the selected tokens are supplanted with another random token, and 10% of selected tokens remain unchanged (Devlin et al., 2019, p. 4174). The task then is to correctly predict all Q tokens sampled for the task based on their respective input token representation (for an illustration see Figure 2.7) (Devlin et al., 2019, p. 4173-4174). In doing so, self-attention is possible with regard to all—instead of only preceding or only succeeding—tokens in the same sequence, and thus the learned representations for all tokens in the sequence can capture encoded information from bidirectional contexts (Devlin et al., 2019, p. 4174, 4182).

In addition to the masked language modeling task, BERT is also pretrained on a next sentence prediction task in which the model has to predict whether the second of two text segments it is presented with succeeds the first (Devlin et al., 2019, p. 4172, 4174). The second pretraining task is hypothesized to serve the purpose of making BERT also a well-generalizing pretrained model for NLP target tasks that require an understanding of the association between two text segments (e.g. question answering or natural language inference) (Devlin et al., 2019, p. 4172, 4174).

To accommodate for the pretraining tasks and to prepare for a wide spectrum of downstream target tasks, the input format accepted by BERT consists of the following elements (see Figure 2.7) (Devlin et al., 2018, p. 3-5; Devlin et al., 2019, p. 4174-4175, 4182-4183):

- Each sequence of tokens $(a_1, \ldots, a_t, \ldots, a_T)$ is set to start with the classification token *'[CLS]'*. After fine-tuning, the *'[CLS]'* token functions as an aggregate representation of the entire sequence and is used as an input for single sequence classification target tasks such as sentence sentiment analysis.
- The separation token *'[SEP]'* is used to separate different segments.
- Each token a_t is represented by the sum of its input embedding with a positional embedding and a segment embedding.[24]

[23]The concatenation of representations learned by a forward language model with the representations of a backward language model does not generate representations that genuinely draw from left and right contexts (Devlin et al., 2019, p. 4172). The reason is that the forward and backward representations are learned separately and each representation captures information only from a unidirectional context (Yang et al., 2019, p. 5753).

[24]BERT employs the WordPiece tokenizer and uses a vocabulary of 30,000 features (Wu et al., 2016). WordPiece (Schuster & Nakajima, 2012) is a variant of the Byte-Pair Encoding (BPE) subword tokenization algorithm. (For more information on subword tokenization algorithms see Appendix 2.C.) The segment embeddings allow the model to distinguish segments. All tokens belonging to the same segment have the same segment embedding.

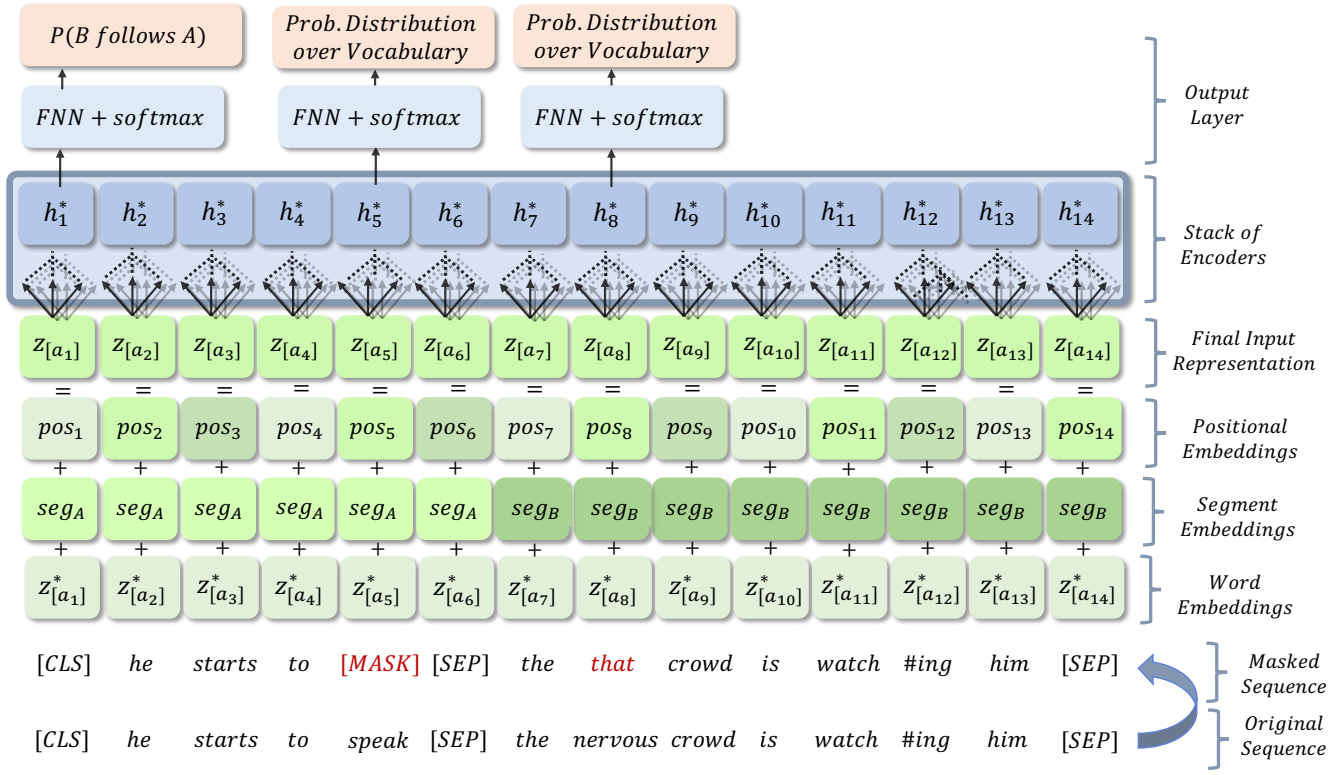


Figure 2.7: **Pretraining BERT.** Architecture of BERT in pretraining. Assume that in the lowercased example sequence consisting of the segment pair *'he starts to speak. the nervous crowd is watch-ing him.'* the tokens *'speak'* and *'nervous'* were sampled to be masked. *'speak'* is replaced by the *'[MASK]'* token and *'nervous'* is replaced by the random token *'that'*. The model's task is to predict the tokens *'speak'* and *'nervous'* from the representation vectors it learns at the positions of the input embeddings of *'[MASK]'* and *'that'*. $P(BfollowsA)$ is the next sentence prediction task. FNN stands for feedforward neural network.

- In practical software-based implementations, BERT-like models typically require all input sequences to have the same length (Hugging Face, 2020a). To meet this requirement, the text sequences are tailored to the same length by padding or truncation (Hugging Face, 2020a). Truncation is typically employed if text sequences exceed the maximum accepted sequence length. Truncation implies that excess tokens are removed. In padding, a padding token (*'[PAD]'*) is repeatedly added to a sequence until the desired length is reached (McCormick & Ryan, 2019). Note that due to memory restrictions, the maximum sequence length that BERT can process is limited to 512 tokens.

BERT is pretrained with the masked language modeling and the next sentence prediction task. As pretraining corpora the BooksCorpus (Zhu et al., 2015) and the English Wikipedia are used (Devlin et al., 2019, p. 4175). Taken together the pretraining corpus consists of 3.3 billion tokens (Devlin et al., 2019, p. 4175). (For details on pretraining BERT see

Figure 2.8: **Fine-Tuning BERT.** Architecture of BERT during fine-tuning on a single sequence classification task.

Appendix 2.D.)

Token representations that are produced from a pretrained BERT model afterward can be extracted and taken as an input for a target task-specific architecture as in a classic feature extraction approach (Devlin et al., 2019, p. 4179). The more common way to use BERT, however, is to fine-tune BERT on the target task. Here, merely the output layer from pretraining is exchanged with an output layer tailored for the target task (Devlin et al., 2019, p. 4173, 4184). Other than that, the same model architecture is used in pretraining and fine-tuning (compare Figures 2.7 and 2.8) (Devlin et al., 2019, p. 4173, 4184). If the target task is to classify single input sequences into a set of predefined categories (see Figure 2.8), the hidden state vector generated by the last Transformer encoder for the *[CLS]* token, $\boldsymbol{h}_1^*$, enters the following output layer to generate output vector $\boldsymbol{y}$ (Hugging Face, 2018):

$$\boldsymbol{y} = softmax(\tanh(\boldsymbol{h}_1^* \boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2) \tag{2.16}$$

$\boldsymbol{y}$'s dimensionality corresponds to C—the number of categories in the target classification task. The cth element of vector $\boldsymbol{y}$ gives the predicted probability of the input sequence belonging to the cth class. Note that during fine-tuning not only weight matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ in Equation 2.16 but *all* parameters of BERT are updated (Devlin et al., 2018,

p. 6).[25]

2.5.2 More Transformer-Based Pretrained Models

A helpful way to describe and categorize the various Transformer-based models for transfer learning is to differentiate them according to their pretraining objective and their model architecture (Hugging Face, 2020b). The major groups of models in this categorization scheme are autoencoding models, autoregressive models, and sequence-to-sequence models (Hugging Face, 2020b).

2.5.2.1 Autoencoding Models

In their pretraining task, autoencoding models are presented with input sequences that are altered at some positions (Yang et al., 2019, p. 5753-5755). The task is to correctly predict the uncorrupted sequence (Yang et al., 2019, p. 5753-5755). The models' architecture is typically composed of the encoders of the Transformer which implies that autoencoding models can access the entire set of input sequence tokens and can learn bidirectional token representations (Hugging Face, 2020b). Autoencoding models tend to be especially high performing in sequence or token classification target tasks (Hugging Face, 2020b). BERT with its masked language modeling pretraining task is a typical autoencoding model (Yang et al., 2019, p. 5753).

Among the various extensions of BERT that have been developed since its introduction, RoBERTa (Liu et al., 2019) is widely known. RoBERTa makes changes in the pretraining and hyperparameter settings of BERT. For example, RoBERTa is only pretrained on the masked language modeling and not the next sentence prediction task (Liu et al., 2019, p. 4-6). Masking is performed dynamically each time before a sequence is presented to the model instead of being conducted once in data preprocessing (Liu et al., 2019, p. 4, 6). Moreover, RoBERTa is pretrained on more data and more heterogeneous data (e.g. also on web corpora) (Liu et al., 2019, p. 5-6). On ALBERT (Lan et al., 2020) and ELECTRA (Clark et al., 2020), two further well-known autoencoding models, see Appendix 2.E.

One major disadvantage of pretrained models that are based on the self-attention mechanism in the Transformer is that currently available hardware does not allow Transformer-based models to process long text sequences (Beltagy et al., 2020, p. 1). The reason is that the memory and time required increase quadratically with sequence length (Beltagy et al., 2020, p. 1). Long text sequences thereby quickly exceed memory limits of presently existing graphics processing units (GPUs) (Beltagy et al., 2020, p. 1). Transformer-based

[25] Based on their experiences with adapting BERT on various target tasks, the authors recommend to use for fine-tuning a mini-batch size of 16 or 32 sequences and a global Adam learning rate of 5e-5, 3e-5, or 2e-5 (Devlin et al., 2019, p. 4183-4184). They also suggest to set the number of epochs to 2, 3 or 4 (Devlin et al., 2019, p. 4184).

pretrained models therefore typically induce a maximum sequence length. For BERT and related models this maximum length usually is 512 tokens. Simple workarounds for processing sequences longer than 512 tokens (e.g. truncating texts or processing them in chunks) lead to information loss and potential errors (Beltagy et al., 2020, p. 2-3). To solve this problem, various works present procedures for altering the Transformer architecture such that longer text documents can be processed (Child et al., 2019; Dai et al., 2019; Beltagy et al., 2020; Kitaev et al., 2020; Wang et al., 2020; Zaheer et al., 2020).

Here, one of these models, the Longformer (Beltagy et al., 2020), is presented in more detail. The Longformer introduces a new variant of the attention mechanism such that time and memory complexity does not scale quadratically but linearly with sequence length and thus longer texts can be processed (Beltagy et al., 2020, p. 3). The attention mechanism in the Longformer is composed of a sliding window as well as global attention mechanisms for specific preselected tokens (Beltagy et al., 2020, p. 3-4). In the sliding window, each input token a_t—instead of attending to all tokens in the sequence—attends only to a fixed number of tokens to the left and right of a_t (Beltagy et al., 2020, p. 3). In order to learn representations better adapted to specific NLP tasks, the authors use global attention for specific tokens on specific tasks (e.g. for the *'[CLS]'* token in sequence classification tasks) (Beltagy et al., 2020, p. 3-4). These preselected tokens directly attend to all tokens in the sequence and enter the computation of the attention vectors of all other tokens (Beltagy et al., 2020, p. 3-4). The position embeddings of the Longformer allow processing text sequences of up to 4,096 tokens (Beltagy et al., 2020, p. 6). This Longformer-specific attention mechanism can be used as a plug-in replacement of the original attention mechanism in any Transformer-based model (Beltagy et al., 2020, p. 6). Beltagy et al. (2020, p. 2) insert the Longformer attention mechanism into the RoBERTa architecture. The Longformer then is pretrained by continuing to pretrain RoBERTa with the Longformer attention mechanism on the masked language modeling task (Beltagy et al., 2020, p. 2).

2.5.2.2 Autoregressive Models

Autoregressive models are pretrained on the classic language modeling task (see Equations 2.3 and 2.4) (Yang et al., 2019, p. 5753-5755). They learn a forward language model in which they are trained to predict the next token given all the preceding tokens in the sequence, $P(a_t|a_1, \ldots, a_{t-1})$, and/or a backward language model in which the next token is predicted given all its succeeding tokens, $P(a_t|a_T, \ldots, a_{t+1})$ (Yang et al., 2019, p. 5753). Hence, autoregressive models are not capable of learning genuine bidirectional representations that draw from left and right contexts (Yang et al., 2019, p. 5753). In correspondence with this pretraining objective, their architecture is typically based only on the decoders of the Transformer (without encoder-decoder attention) (Hugging Face, 2020b). To ensure that an autoregressive model only consumes previously predicted output tokens, the self-attention layer of its decoders are masked such that the model only can attend to the preceding but not the proceeding tokens (Hugging Face, 2020b).

Due to their decoder-based architecture and the characteristics of their pretraining task, autoregressive models are typically very good at target tasks in which they have to generate text (Hugging Face, 2020b). Autoregressive models, however, can be successfully fine-tuned to a large variety of downstream tasks (Hugging Face, 2020b). An elementary autoregressive model, which is based on the Transformer decoder, is the GPT (Radford et al., 2018). Its successors GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020; Hugging Face, 2020b) are well-known and also play a role in the context of zero-shot learning (see Appendix 2.B). In a recent, ingenious article, Porter & Velez (2021) apply the GPT-2 to generate placebos texts in social science survey experiments. Another model using the autoregressive language modeling framework in pretraining is the XLNet (Yang et al., 2019). (On the XLNet see Appendix 2.F.)

2.5.2.3 Sequence-to-Sequence Models

The architecture of sequence-to-sequence models contains Transformer encoders and decoders (Hugging Face, 2020b). They tend to be pretrained on sequence-to-sequence tasks (e.g. translation) and, consequently, are especially suited for sequence-to-sequence-like downstream tasks such as translating or summarizing input sequences (Hugging Face, 2020b). The Transformer itself is a sequence-to-sequence model for translation tasks. BART (Lewis et al., 2020) and the T5 (Raffel et al., 2020) are further well-known sequence-to-sequence models applicable to a large variety of target tasks (Hugging Face, 2020b). (On the T5 and BART see Appendix 2.G.)

2.5.3 Foundation Models: Concept, Limitations, and Issues

Transfer learning, as well as the Transformer architecture, is at the heart of fundamental changes within the entire field of NLP and beyond: In a recent article written by a large group of researchers and students at the Stanford Institute for Human-Centered Artificial Intelligence, it is stated that

> "[...] there was a sociological inflection point around the introduction of BERT. Before 2019, self-supervised learning with language models was essentially a *subarea* in NLP, which progressed in parallel to other developments in NLP. After 2019, self-supervised learning with language models became more of a *substrate* of NLP [...]" (Bommasani et al., 2021, p. 5)

Bommasani et al. (2021) call deep neural networks that are pretrained in a self-supervised fashion on large amounts of data and then can be adapted to a wide spectrum of target tasks *foundation models*. The mode of learning, that they refer to, in which one pretrained model serves as the *foundation* for various different tasks, not only has taken hold in NLP but across the field of AI research (Bommasani et al., 2021). Transfer learning with (Transformer-based) neural networks is not only applied to text data but also, for example,

to images (Dosovitskiy et al., 2021; Goyal et al., 2021), videos (Sun et al., 2019a), audio data (Baevski et al., 2020), protein sequences (Rives et al., 2021), and data in tabular form (Yin et al., 2020). Moreover, an increasing number of models are multimodal (Lu et al., 2019; Bapna et al., 2021; Fu et al., 2021; Radford et al., 2021; Ramesh et al., 2021, e.g.).

AI research previously had moved from classic machine learning (in which a function between representations of data and outputs are learned but representations still have to be engineered) to the era of deep learning (in which deep neural networks learn representations of data but still typically one model is trained for one specific task) (Bommasani et al., 2021, p. 3-4). Now, AI research seems to move toward highly general all-purpose models (Bommasani et al., 2021, p. 3-6). Summarizing current developments, the emerging mode of learning is characterized by the implementation of a deep neural network that

- is frequently based on the Transformer architecture. (Due to its self-attention mechanisms, the Transformer is more flexible and general than convolutional or recurrent neural networks (Bommasani et al., 2021, p. 75-76). The Transformer, however, is not a defining feature of foundation models and at some point may be superseded by new neural network architectures (Jaegle et al., 2021, 2022).)
- has been pretrained—typically in self-supervised learning mode—on massive amounts of data from various sources and domains. (Pretraining does not have to be constrained to one pretraining task conducted on one type of data in one language. Rather, increasingly general models are developed by pretraining on multiple tasks (Wei et al., 2022; Aribandi et al., 2022), pretraining on data in multiple languages (Conneau et al., 2020; Babu et al., 2021), or pretraining on data from multiple modes (Luo et al., 2020; Bapna et al., 2021; Radford et al., 2021).)
- can process—and learn representations for—these data inputs (probably across domains, languages, and modes) (Tenney et al., 2019a; Radford et al., 2021),
- after adaptation can be applied to a wide spectrum of tasks, for example, various language understanding tasks (e.g. sentiment analysis, question answering) (Devlin et al., 2019; Liu et al., 2019), different language understanding plus language generation tasks (Lewis et al., 2020; Wei et al., 2022), or tasks related to the understanding and/or generation of data in multiple modes (Luo et al., 2020; Fu et al., 2021; Ramesh et al., 2021).[26]

The application of deep neural networks and—in particular—the use of (Transformer-based) neural networks with transfer learning has triggered significant performance enhancements across NLP. Still, there are substantive limitations and problematic issues—that are also relevant to social scientists that seek to apply these models for their research purposes. In the following, several of these aspects will be outlined. For an elaborate discussion see Bommasani et al. (2021).

[26]Note that the step of adaptation to a specific target task is skipped in zero-shot learning.

- One major concern is that the amounts of resources required in pretraining (especially in terms of data and compute) are so massively large that academic institutions and the scientific research community struggle (or are not able) to pretrain the largest foundation models (but see https://bigscience.huggingface.co/) (Bommasani et al., 2021, p. 11). Moreover, the data used in pretraining and the model source code are not always publicly available (Aßenmacher & Heumann, 2020, p. 4; Riedl, 2020). This raises deep concerns regarding accessibility and traceability.
- Another problematic aspect is that these models reflect the representational biases (e.g. stereotypes, underrepresentations) encoded in the data they have been pretrained on (Bommasani et al., 2021, p. 129-131). As soon as a model is adapted to some target task, these biases materialize with serious negative consequences (Bommasani et al., 2021, p. 130).
- A further problem is the fixed (typically relatively small) maximum sequence length that Transformer-based models can process. Whatever the given current computational restrictions, efficient modifications of the self-attention mechanism, as for example presented by the Longformer (Beltagy et al., 2020), allow for longer sequences to be processed than with the original Transformer and thereby constitute important steps toward alleviating this major drawback. (For an evaluative overview of efficient Transformer-based models see Tay et al. (2021).)
- Further research is required regarding the theoretical underpinnings of foundation models. Exactly why, and in which circumstances, pretraining on a pretraining data distribution helps in reducing the loss on various target task data distributions so far is not well understood (Bommasani et al., 2021, p. 117-121).

Besides these issues related to large pretrained representation models, the mere application of deep neural networks is likely to pose further difficulties for social science researchers. One issue is interpretability: If a researcher applies a learning method to measure an a priori-defined concept from text, the ability to as closely as possible imitate human codings on yet unseen test data is arguably the most important goal because this ability indicates the measure's validity. In this very context, a model's prediction performance thus is considered more important than a model's interpretability (see argumentation in Section 2.1). Yet interpretability (i.e. the human-understandable and accurate representation of a model's decision process) can be highly important (Miller, 2019; Jacovi & Goldberg, 2020). Interpretability makes a model's predictions more transparent and more human-retraceable. If a researcher can understand why a model made predictions in a particular way and if a researcher can estimate the effect of input features for a model's predictions, this can enhance the researcher's trust in the model she applies (Doshi-Velez & Kim, 2017, p. 2; Molnar, 2022, ch. 3.1).

For some conventional machine learning methods (such as linear regression, logistic regression, or decision trees) it is straightforward to assess the effect or importance of specific features on a model's predictions (Molnar, 2022, ch. 5). Furthermore, there is a large spec-

trum of model-agnostic interpretation tools that can be applied to any machine learning model (for an overview see Molnar (2022, ch. 6)). But because deep neural networks do not operate on a priori-defined, human-engineered features but have a layered architecture that enables them to learn complex representations of textual inputs, interpretation of deep neural networks is less straightforward and neural network-specific interpretation methods tend to be used (Belinkov & Glass, 2019, p. 49; Molnar, 2022, ch. 10).

For social scientists that apply deep neural networks the open-source library Captum (https://captum.ai/) is likely to be a useful interpretability tool. Captum implements several attribution algorithms that allow researchers to examine how predicted outputs relate to input features (Kokhlikyan et al., 2020, p. 3). Captum furthermore provides tools for analyzing attention patterns (see the tutorial at https://captum.ai/tutorials/Bert_SQUAD_Interpret2). (For an overview of common methods to make neural networks interpretable see Appendix 2.H.)

Another issue is reproducibility: As for conventional models, reproducibility issues with deep neural networks typically arise from random elements that are used during optimization and/or when sampling data (e.g. in cross-validation, or batch allocation). In both cases, sources of randomness usually can be controlled. Yet in practice, this often proves to be more difficult for deep neural networks than for conventional models. Note, furthermore, that full reproducibility *across* different computing platforms and environments cannot be ensured (Freidank, 2020; Torch Contributors, 2021).

2.6 Applications

Researchers who wish to apply sequential transfer learning can pretrain a model on a suitable source task by themselves and then finetune the pretrained model to their target task of interest. (Researchers that seek to pretrain a model by themselves may find reading the paper by Aßenmacher et al. (2021) useful as it provides a consistent comparison of different pretraining objectives and settings.) Because pretraining tends to be very expensive, however, the much more convenient, cost-effective, and common approach for applied researchers is to make use of an already pretrained model and then to merely adapt the pretrained model to the target task. Hence, to fully leverage the power of neural transfer learning, researchers require access to already pretrained models that they can fine-tune on their specific tasks. Such access is provided by Hugging Face's Transformers (Wolf et al., 2020) which is an open-source library that contains thousands of pretrained NLP models ready to download and use: https://huggingface.co/. The Hugging Face library contains pretrained versions of the models discussed here and a great many models more. Most of the available pretrained models in the Hugging Face library have been pretrained on English texts, yet there are numerous monolingual models pretrained in other languages. Moreover, the library also comprises several models for cross-lingual learning that have been pretrained on text in several languages. The pretrained models can be accessed via

the respective Transformers Python package that also provides compatibility with PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2016). For basic guidance on deep learning and transfer learning in practice see Appendix 2.I.

In the applications presented in the following neural transfer learning is conducted in Python 3 (van Rossum & Drake, 2009) making use of PyTorch (Paszke et al., 2019) and Hugging Face's Transformers (Wolf et al., 2020). The code is executed in Google Colab.[27] Whenever a GPU is used, an NVIDIA Tesla T4 is employed. The source code for this study is openly available in figshare at https://doi.org/10.6084/m9.figshare.14394173. Especially the shared Colab Notebooks serve as templates that other researchers can easily adapt for their NLP tasks.[28]

2.6.1 Models, Data Sets, and Tasks

The aim of this applied section is to explore the use of transfer learning with Transformer-based models for text analyses in social science contexts. To do so, the prediction performances of BERT, RoBERTa, and the Longformer are compared to the performances of two conventional machine learning algorithms: Support vector machines (SVMs) (Boser et al., 1992; Cortes & Vapnik, 1995) and the gradient tree boosting algorithm XGBoost (Chen & Guestrin, 2016). SVMs have been widely used in social science text applications (e.g. Diermeier et al., 2011; D'Orazio et al., 2014; Ramey et al., 2019; Miller et al., 2020; Sebők & Kacsuk, 2021). As a tree-based (boosting) method XGBoost represents a type of algorithm also commonly utilized (e.g. Katagiri & Min, 2019; Anastasopoulos & Bertelli, 2020; Park et al., 2020). The comparisons are conducted on the basis of three different data sets of varying sizes and textual styles:

1. The Ethos Dataset (Duthie & Budzynska, 2018) is a corpus of 3,644 sentences from debates in the UK parliament (train: 2,440; test: 1,204). Duthie & Budzynska (2018) gathered 90 debate transcripts from the period Margaret Thatcher served as Prime Minister (1979-1990). In each debate, they recorded for each spoken sentence whether the sentence refers to the ethos (i.e. the character) of another politician or party, and if so whether the other's ethos is supported or attacked (Duthie & Budzynska, 2018, p. 4042). The task associated with this data set thus is to as precisely as possible measure the concept of ethos from text. Though not conducted here, this measurement of ethos from text could be the

[27] https://colab.research.google.com/notebooks/intro.ipynb

[28] More specifically, bag-of-words and word vector-based text preprocessing is implemented in R (R Core Team, 2020) using the packages quanteda (Benoit et al., 2018), stringr (Wickham, 2019), text2vec (Selivanov et al., 2020), and rstudioapi (Ushey et al., 2020). Training and evaluating the pretrained Transformer models and the conventional machine learning algorithms is conducted in Python 3 (van Rossum & Drake, 2009) employing the modules and packages gdown (Kentaro, 2020), imbalanced-learn (Lemaître et al., 2017), matplotlib (Hunter, 2007), NumPy (Oliphant, 2006), pandas (McKinney, 2010), seaborn (Michael Waskom and Team, 2020), scikit-learn (Pedregosa et al., 2011), PyTorch (Paszke et al., 2019), watermark (Raschka, 2020), Hugging Face's Transformers (Wolf et al., 2020), and the XGBoost Python package (Chen & Guestrin, 2016).

first step in a larger analysis of the network of ethotic expressions between politicians. With 82.5% of the sentences being non-ethotic, 12.9% attacking and 4.6% supporting another's ethos, the data are quite imbalanced.

2. The Legalization of Abortion Dataset comprises 933 tweets (train: 653; test: 280). The data set is a subset of the Stance Dataset (Mohammad et al., 2017) that was used for detecting the attitude toward five different targets from tweets. Mohammad et al. (2017) collected the tweets via hashtags and let CrowdFlower workers annotate the tweets regarding whether the tweeter is in favor, against, or neutral toward the target of interest (Mohammad et al., 2017, p. 4-7). The Legalization of Abortion Dataset used here contains those tweets that refer to the target 'legalization of abortion'. The task associated with this data set thus is to measure attitudes toward a policy issue from text. 58.3% of the tweets express an opposing and 17.9% a favorable position toward legalization of abortion whilst 23.8% express a neutral or no position.

3. The Wikipedia Toxic Comment Dataset (Jigsaw/Conversation AI, 2018) contains 159,571 comments from Wikipedia Talk pages that were annotated by human raters for their toxicity. On Wikipedia Talk pages contributors discuss changes to Wikipedia pages and articles.[29] Toxic comments are comments that are obscene, threatening, insulting, express hatred toward social groups, and identities, "are rude, disrespectful, or otherwise likely to make people leave the discussion" (Dixon, 2017). Whereas the tasks associated with the Ethos and the Legalization of Abortion Datasets are multi-class classification tasks, the task here is a simple binary classification task in which the aim is to separate toxic from non-toxic comments. Tasks in which the aim is to separate documents in which a concept (here: toxicity) occurs from documents in which the concept does not occur are common in text-based social science applications. Such tasks often constitute a first step in a text analysis in which documents that refer to concepts or entities that are of interest to the analysis have to be singled out from a large heterogeneous corpus (King et al., 2017, p. 971). Often, such tasks are imbalanced classification problems (Manning et al., 2008, p. 155). And also here, only 9.6% of comments in the data are labeled as being toxic.

In this work, the Wikipedia Toxic Comment Dataset is used to assess in how far the algorithms' performances vary with training set size. To do so, five training data sets of sizes 10,000, 5,000, 2,000, 1,000, and 500 and a test set comprising 1,000 comments are sampled uniformly at random from the 159,571 comments in the Wikipedia Toxic Comment Dataset. To account for the uncertainty induced by operating on samples of training sets, five iterations are performed. This is, the sampling is repeated five times, such that in the end there are five sets comprising five training data sets of varying sizes.[30]

[29] https://en.wikipedia.org/wiki/Help:Talk_pages

[30] More precisely: To get five differently sized training data sets evaluated on the same test set, the following steps are conducted:

1. A set of 11,000 comments is sampled uniformly at random from the 159,571 comments in the Wikipedia Toxic Comment Dataset.

2. A random sample of 1,000 comments is drawn from the set of 11,000 comments to become the test

The three applications—Ethos, Abortion, and Toxic—are selected so that the methods are applied on text data that, on the one hand, represent types of texts that are often used within social science and, on the other hand, vary regarding core characteristics (for a comparison see also Figures 2.9a to 2.9i). Across the three applications, textual style ranges from the formal, rule-based, courteous language of parliamentary speeches over the short, statement-like nature of tweets to informal, interrelating (and at times disrespectful) comments from online discussions (to get an impression see the most frequent trigrams in Figures 2.9c, 2.9f, 2.9i). The tasks associated with the applications vary with regard to the number of class labels (binary vs. three-class classification) and the distribution over these labels (see Figures 2.9a, 2.9d, 2.9g). The data sets are furthermore characterized by different document lengths (see Figures 2.9b, 2.9e, 2.9h) and vary with regard to their sizes (and hence the number of data available for training). The fewer training data, the more class labels, and the more imbalanced the distribution over class labels, the more difficult the task is likely to be. Especially with regard to imbalanced classification problems when there are few training instances in the minority class, it can be difficult to have enough training data to train an adequately performing deep neural network from scratch. As transfer learning reduces the number of required training data instances, transfer learning is likely to facilitate the training of neural networks in such situations of imbalance.

2.6.2 Text Preprocessing for the Conventional Models

Two types of preprocessing procedures are employed on the raw texts to provide data representation inputs for the conventional models SVM and XGBoost:

1. Basic BOW: The texts are tokenized into unigrams. Punctuation, numbers, and symbols are removed in the Ethos application but kept in the other applications. Afterward, the tokens are lowercased and stemmed. Then, tokens occurring in less than a tiny share of documents (e.g. 0.1% in the Ethos application) and more than a large share of documents (e.g. 33% in the Ethos application) are excluded. Finally, the elements in the document-feature matrix are weighted such that the mere presence (1) vs. absence (0) of each feature within each document is recorded.

2. GloVe Representation: For each unigram that occurs at least 3 (Ethos, Abortion) or 5 (Toxic) times in the respective corpus, the 300-dimensional pretrained GloVe word vector is identified (Pennington et al., 2014).[31] Each document then is represented by the mean

data set. The remaining 10,000 comments constitute the first training data set.

3. From the training set of 10,000 comments, a subset of 5,000 comments is randomly drawn to become the second training set. From this subset again a smaller training subset of 2,000 texts is sampled from which a subset of 1,000 and then 500 comments are drawn.

4. To account for the induced uncertainty, steps (a) to (c) are repeated five times.

[31] GloVe embeddings are pretrained based on a web data corpus from CommonCrawl comprising 42 billion tokens (Pennington et al., 2014, p. 1538).

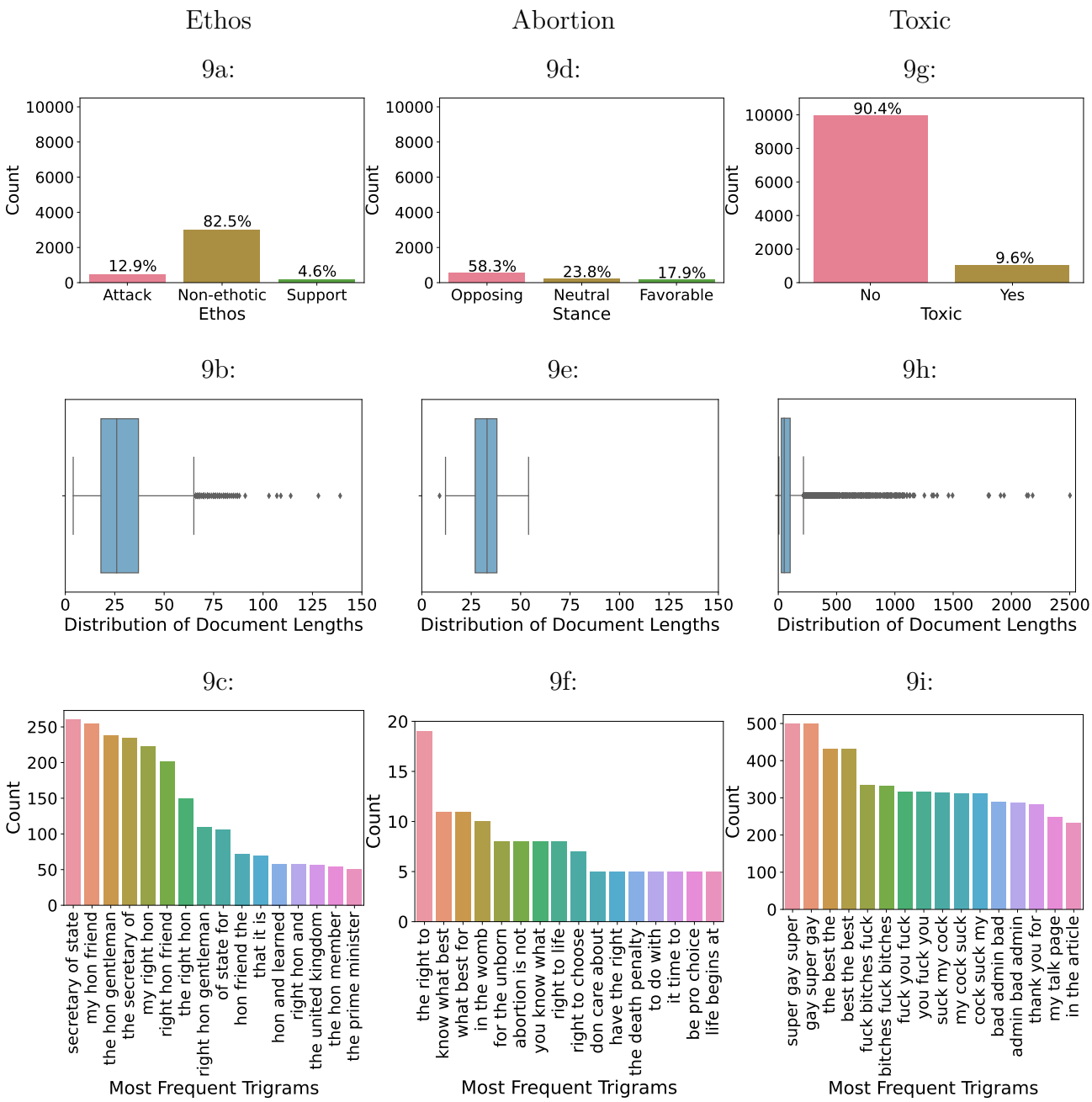


Figure 2.9: **Description of the Corpora.** Figures 9a-9c: Ethos Datatset. Figures 9d-9f: Legalization of Abortion Dataset. Figures 9g-9i: Sample of 11,000 comments from the Wikipedia Toxic Comment Dataset. Figures 9a-9g: Class label distributions. Figures 9b-9h: Boxplots visualizing the distribution of the number of tokens per document. Figures 9c-9i: Most frequent trigrams. If possible and reasonable, the figures' axes have the same scale.

over its unigrams' GloVe word vectors. Note that due to making use of pretrained feature representations that are not updated during training, GloVe Representation constitutes a transfer learning approach with feature extraction. By averaging over the unigrams' word embeddings, the word order, however, is not taken into account.

2.6.3 Text Preprocessing and Fine-Tuning Settings for the Transformer-Based Models

The Transformer-based models are applied after the documents have been transformed to the required input format. In each document, the tokens are lowercased and the special *'[CLS]'* and *'[SEP]'* tokens are added. Then, each token is converted to an index identifying its input embedding and is associated with an index identifying its segment embedding. Additionally, each document is padded to the same length. In the Ethos and Legalization of Abortion corpora, this length corresponds to the maximum document length among the training set documents, which is 139 and 54 tokens respectively. The comments from Wikipedia Talk pages pose a problem here: An inspection of the distribution of sequence lengths in the sampled subsets of the Wikipedia Toxic Comment Dataset (see Figure 2.9h) shows that the vast majority of comments are shorter than the maximum number of 512 tokens that BERT and RoBERTa can distinguish—but there is a long tail of comments exceeding 512 tokens. To address this issue, two different approaches are explored: For BERT, following the best strategy identified by Sun et al. (2019b), in each comment that is longer than 512 tokens, only the first 128 and the last 382 tokens are kept while the tokens positioned in the middle are removed. RoBERTa, in contrast, is replaced with the Longformer in the Toxic application. For the Longformer the sequence length is set to $2*512 = 1{,}024$ tokens. This ensures that in each run only a small one- or two-digit number of sequences that are longer than 1,024 tokens are truncated by removing tokens from the middle whilst padding the texts to a shared length that still can be processed with given memory restrictions.[32]

When adapting the pretrained Transformer-based models to the target tasks, the Adam algorithm as introduced by Loshchilov & Hutter (2019) with a linearly decaying global learning rate, no warmup, and no weight decay is employed. Dropout is set to 0.1. To fine-tune the models within the memory resources provided by Colab, small batch sizes are used. In the Ethos and Abortion applications, a batch size of 16 is selected. A batch in the Toxic application comprises 8 (and for the Longformer 4) text instances. Note that when selecting a small batch size (e.g. because of memory restrictions) this is not a disadvantage but rather the opposite: Research suggests that smaller batch sizes not only require less memory but also have better generalization performances (Keskar et al., 2017; Masters & Luschi, 2018). To ensure that the learning process with small batch sizes does

[32]Except for the removal of tokens positioned in the middle of overlong input documents, all described formatting steps for the Transformer-based models are implemented in Hugging Face's Transformers library and therefore can be easily applied.

not get too volatile, one merely has to account for the fact that smaller batch sizes require correspondingly smaller learning rates (Brownlee, 2019).

Moreover, for the pretrained models, the base size of the model architecture is used instead of the large or extra large model versions. So, for example, $\text{BERT}_{\text{BASE}}$ instead of $\text{BERT}_{\text{LARGE}}$ is applied. Larger models are likely to lead to higher performances. Yet, because they have more parameters, it takes more computing resources to fine-tune them and—especially for small data sets—fine-tuning might lead to results that vary more noticeably across random restarts (Devlin et al., 2019, p. 4176).

2.6.4 Random Oversampling

To handle the class imbalances in the Ethos and Wikipedia Toxic Comment Datasets, the training data are randomly oversampled. In random oversampling, instances of the minority classes are randomly sampled with replacement and added as identical copies to the training data such that the training data become more balanced (Brownlee, 2020). The presence of multiple minority class copies in the training data increases the loss caused by misclassifying minority class instances and hence induces the algorithm to put a stronger focus on correctly classifying minority class examples. To balance the Ethos and Wikipedia Toxic Comment Datasets but also to prevent too strong overfitting on the training data, the minority classes are moderately oversampled such that the size of the minority classes is 1/4th the size of the majority class.

2.6.5 Hyperparameter Tuning

For each evaluated combination of an algorithm and a preprocessing procedure, a grid search across sets of hyperparameter values is performed via five-fold cross-validation on the training set. For the Transformer-based models, the hyperparameter grid search explores model performances across combinations of different learning rates and epoch numbers. Accounting for the fact that in the optimization process the gradient updates are conducted based on small batches, relatively small global Adam learning rates {1e-05, 2e-05, 3e-05} are inspected. The number of epochs explored is $\{2, 3, 4\}$.[33]

At the end of hyperparameter tuning, the best performing set of hyperparameters according to the macro-averaged F_1-Score and overfitting considerations is selected. Then the model

[33]Note that for the Longformer (for which a batch size of 4 is used) the learning rate is set to 1e-05 and the number of epochs explored is $\{2, 3\}$. Hyperparameter tuning for the SVMs compares a linear kernel and a Radial Basis Function kernel. The explored values are $\{0.1, 1.0, 10.0\}$ for penalty weight $\mathcal{C}$, and—in the case of the Radial Basis Function kernel—values of $\{0.001, 0.01, 0.1\}$ are inspected for parameter γ, that specifies the radius of influence for single training examples. Regarding the XGBoost algorithms, the grid search explores 50 vs. 250 trees, each with a maximum depth of 5 vs. 8, and XGBoost learning rates of $0.001, 0.01$, and 0.1. For details on SVM and XGBoost hyperparameters see also scikit-learn Developers (2020a,c) and xgboost Developers (2020).

with the chosen hyperparameter setting is trained on the entire training data set and evaluated on the test set via the macro-averaged F_1-Score.

2.6.6 Results

The results for the Ethos, Abortion, and Toxic classification tasks are presented in Table 2.1. Figure 2.10 additionally visualizes the results for the Toxic application.

	Ethos	Abortion	Toxic0.5K	Toxic1K	Toxic2K	Toxic5K	Toxic10K
SVM BOW	0.566	0.526	0.711	0.754	0.782	0.802	0.817
SVM GloVe	0.585	0.545	0.739	0.786	0.789	0.822	0.840
XGBoost BOW	0.563	0.540	0.709	0.734	0.742	0.775	0.777
XGBoost GloVe	0.513	0.506	0.710	0.753	0.774	0.804	0.823
BERT	0.695	0.593	0.832	0.857	0.888	0.905	0.901
RoBERTa/Longf.	0.747	0.617	0.849	0.875	0.884	0.890	0.906

Table 2.1: **Macro-Averaged F_1-Scores.** Macro-averaged F_1-Scores of the evaluated models for the Ethos, Abortion and Toxic classification tasks. If there are C classes such that $y_i \in \{\mathcal{G}_1, \ldots, \mathcal{G}_c, \ldots, \mathcal{G}_C\}$, the F_1-Score for a particular class $\mathcal{G}_c$ is the harmonic mean of precision and recall for this class (Manning et al., 2008, p. 156). Recall indicates what proportion of instances that truly belong to class $\mathcal{G}_c$ have been correctly classified as being in $\mathcal{G}_c$. Precision informs about what share of instances that have been predicted to be in class $\mathcal{G}_c$ truly belong to class $\mathcal{G}_c$. The F_1-Score can range from 0 to 1 with 1 being the highest value signifying perfect classification. The macro-averaged F_1-Score is the unweighted mean of the F_1-Scores of each class (scikit-learn Developers, 2020b). By not weighting the F_1-Scores according to class sizes, algorithms that are bad at predicting the minority classes are penalized more severely (scikit-learn Developers, 2020b). In the Toxic application, for each tested training data set size, {500, 1,000, 2,000, 5,000, 10,000}, the mean of the macro-averaged F_1-Scores across the five iterations is shown. The column labeled Toxic0.5K gives the mean of the macro-averaged F_1-Scores for the Toxic classification task with a training set size of 500 instances. SVM BOW and XGBoost BOW denote SVM and XGBoost with bag-of-words preprocessing. SVM GloVe and XGBoost GloVe refer to SVM and XGBoost with GloVe representations. In RoBERTa/Longf., RoBERTa is applied for the Ethos and the Abortion target tasks whereas the Longformer is used for the Toxic comment classification tasks. Gray colored cells highlight the best performing model for the task.

Across all evaluated classification tasks and training data set sizes, the Transformer-based models for transfer learning tend to achieve higher macro-averaged F_1-Scores than the conventional machine learning algorithms SVM and XGBoost. As has been observed before, the classic machine learning algorithms produce acceptable results given the relatively simple representations of text they are applied on. However, when compared on the basis of

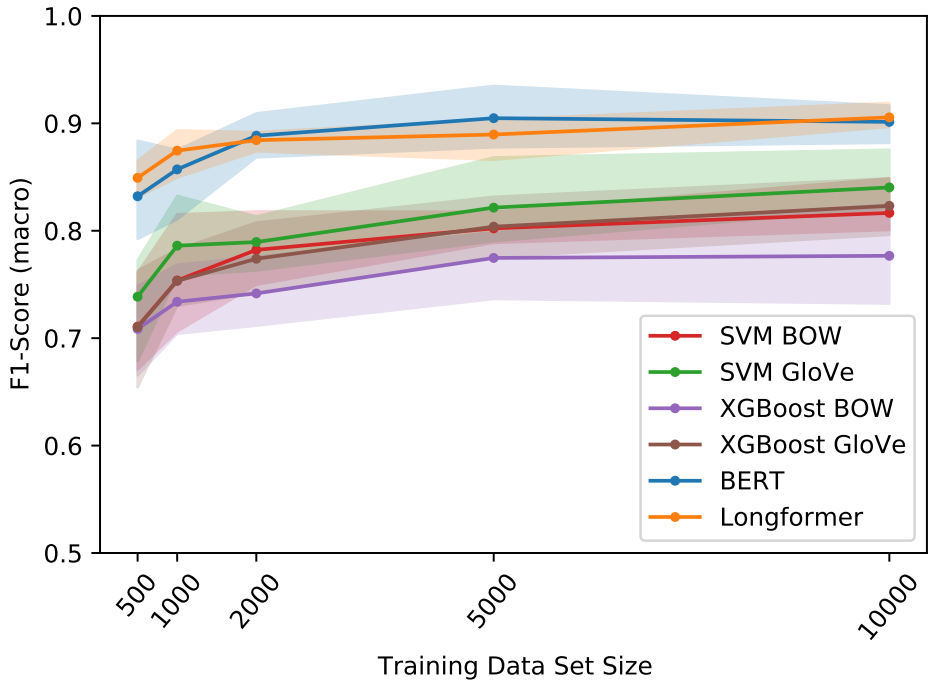


Figure 2.10: **Performances on Toxic Application with Varying Training Data Set Sizes.** For each training data set size and each model, the plotted dots indicate the mean of the test set macro-averaged F_1-Scores across the five iterations. The shaded areas range from the minimum to the maximum macro-averaged F_1-Score obtained across the five iterations.

the (mean) macro-averaged F_1-Scores presented in Table 2.1, BERT, RoBERTa, and the Longformer consistently outperform the best performing conventional model by a margin of at minimum 0.05 to 0.11. These moderate to considerably higher prediction performances across all evaluated textual styles, sequence lengths, and especially the smaller training data set sizes, demonstrate the potential benefits that neural transfer learning with Transformers can bring to analyses in which a researcher aims at having a valid text-based measure of a concept and thus seeks to replicate human codings as accurately as possible. Even if only a small to medium-sized training data set is available, social scientists that apply Transformer-based models in a transfer learning setting are likely to obtain more valid measures for concepts that they measure from texts.

A detailed examination of the macro-averaged F_1-Scores reveals further findings:

- Averaged GloVe representations partly, though not consistently, produce a slight advantage over basic BOW preprocessing. This emphasizes that employing transfer learning on conventional machine learning algorithms by extracting pretrained features (here: GloVe embeddings) and taking them as the data representation input might be beneficial—even if averaging over the embeddings erases information on word order and dependencies.

- For the Ethos and Abortion applications, RoBERTa outperforms BERT to a small extent. This finding is consistent with previous research (Liu et al., 2019, p. 7). In general, it is difficult to disentangle the effects of single modifications of the original BERT architecture and pretraining settings that BERT-extensions as RoBERTa implement (Aßenmacher & Heumann, 2020). It is likely, however, that one important contribution is the longer pretraining on more and more varied data. Whereas BERT is pretrained on a corpus of books and Wikipedia articles, RoBERTa is additionally pretrained on three more large data sets that are based on text passages from the web (Liu et al., 2019, p. 5-6). The larger and more heterogeneous pretraining corpus is likely to enable RoBERTa to produce representations that better generalize across a diverse set of target task corpora as inspected here.

- In the Ethos application, BERT and RoBERTa do not only exceed the performances of the other evaluated models but also the best performing model developed by Duthie & Budzynska (2018). To differentiate non-ethotic from positive and from negative ethotic sentences, Duthie & Budzynska (2018) had created an elaborate NLP pipeline including a POS tagger, dependency parsing, anaphora resolution, entity extraction, sentiment classification, and a deep RNN. Duthie & Budzynska (2018, p. 4045) report a macro-averaged F_1-Score of 0.65 for their best model. BERT and RoBERTa here surpass this performance. As the pretrained BERT and ROBERTa models are simply fine-tuned to the Ethos classification target task without implementing (and having to come up with) an extensive and complex preprocessing pipeline, this demonstrates the efficiency and power of transfer learning.

- With all models achieving only mediocre performances, the Abortion classification task, for which only 653 short Tweets are available as training instances, seems to be especially difficult. BERT and RoBERTa still surpass SVM and XGBoost but with a slightly smaller margin. By applying an SVM with a linear kernel based on word and character n-gram feature representations, Mohammad et al. (2017, p. 13) reach classification performance levels that are higher than the ones reached by the models presented here.[34] The Abortion classification task with short tweets in which the mere N-grams tend to be indicative of the stance toward the issue (Mohammad et al., 2017, p. 13), seems to be an example of a task in which deep learning models only produce a moderate advantage or—if it is easy to select BOW representations that very well capture linguistic variation that helps in discriminating the texts into the categories—even no advantage over traditional machine learning algorithms.

- Across all evaluated training data set sizes, the Transformer-based models with transfer learning tend to be better at solving the Toxic comment classification task compared to the conventional algorithms (see Figure 2.10). As is to be expected, the per-

[34]Mohammad et al. (2017) merely compute the F_1-Score for the favorable and opposing categories leaving out the neutral position. They report a score of 0.664 for their N-gram based SVM classifier (Mohammad et al., 2017, p. 13). Here the corresponding score values are 0.633 for BERT, 0.648 for RoBERTa as well as 0.616 for the best performing conventional model SVM GloVe.

formance levels for all models decrease with decreasing training data set sizes. Yet although the neural models have much more parameters to learn, their macro-averaged F_1-Scores do not decrease more sharply than those of the traditional machine learning algorithms. Especially as training data sets become small, the effectiveness of representations from pretrained models becomes salient. Here, the pretrained models seem to function as a quite effective input to the target task.

- Whereas the Longformer processes text sequences of 1,024 tokens, the input sequences for BERT were truncated at 512 tokens for the Toxic application. Despite this large difference in sequence lengths, BERT only slightly underperforms compared to the Longformer—and matches the Longformer for larger training data set sizes. As only a small share of comments in the Wikipedia Toxic Comment Dataset are longer than 512 tokens (see again Figure 2.9h), the Longformer's advantage of being able to process longer text sequences does not materialize here. Removing tokens from the middle of comments that exceed 512 tokens does not harm BERT's prediction performance and is an effective workaround in this application. For applications based on corpora in which the mass of the sequence length distribution is above 512 tokens, however, the Longformer's ability to process and capture the information contained in these longer documents, is likely to be important for prediction performance.

- Note that the time consumed during training differs substantively between the conventional and the Transformer models. Larger training data sets and smaller batch sizes increase the time required for fine-tuning the pretrained Transformer models. Across the applications presented here, the absolute training time varies between 1 and 276 seconds for SVM BOW, between 32 and 2,272 seconds for BERT and 31 to 9,707 seconds for RoBERTa/Longformer. Applying Transformer-based models requires higher computational resources not only regarding memory but also regarding time.

- An additional analysis that explores the effectiveness of zero-shot learning is conducted (see Appendix 2.J). Across all applications, across both employed pretrained models (RoBERTa and BART), and across all explored hypothesis formulations[35], the macro-averaged F_1-Scores are mediocre and substantially lower than for the fine-tuned models. The highest macro averaged F_1-Scores from zero-shot learning are 0.200 (Ethos), 0.455 (Abortion), and 0.470 (Toxic). Even if the prediction performances of the here implemented zero-shot learning framework are not sufficiently high in order to be applied in research projects in which researchers seek to as accurately as possible measure a priori-defined concepts from texts, this analysis nevertheless demonstrates what can be achieved with representations from pretrained models alone.

[35] A hypothesis formulation is an additional textual input required in the employed natural language inference (NLI) framework for zero-shot learning (see Appendix 2.J).

2.7 Discussion

Advances in NLP research on transfer learning and the attention mechanism, that is incorporated in the Transformer, have paved the way to a new mode of learning in which researchers can hope to achieve higher prediction performances by taking a readily available pretrained model and fine-tuning it, with a manageable amount of resources, to their NLP task of interest (Bommasani et al., 2021). These advances are of interest to social scientists that attempt to have valid measures of concepts from text data but may have limited amounts of training data and resources. To use the potential advantages for social science text analysis, this study has presented and applied Transformer-based models for transfer learning. In the supervised classification tasks evaluated in this study, transfer learning with Transformer models outperformed traditional machine learning across all tasks and data set sizes.

Employing transfer learning with Transformer-based models, however, will not always perform better compared to other machine learning algorithms and is not the most adequate strategy for each and every text-based research question. As the attention mechanism is specialized in capturing dependencies and contextual meanings, these models are likely to generate more accurate predictions if contextual information and long-range dependencies between tokens are relevant for the task at hand. They are less likely to provide much of an advantage if the function to be learned between textual inputs and desired outputs is less complex—for example because single N-grams are strongly indicative of class labels (see e.g. the Abortion application).

Transformer-based models for transfer learning furthermore are useful for supervised classification tasks in which the aim is to achieve an as high as possible prediction performance rather than having an interpretable model. Social scientists whose primary goal is to have as precise as possible text-based measures for concepts they employ may find Transformer-based models for transfer learning highly useful, whereas researchers whose primary goal is to know which textual features are most important in discriminating between class labeled documents (e.g. Slapin & Kirkland, 2020) are likely to be better served with directly interpretable models.

Moreover, due to the sequence length limitations of Transformer-based models, the applicability of these models is currently restricted to NLP tasks that operate on only moderately long text sequences. Research that seeks to reduce the memory resources consumed by the attention mechanism and thus allows for processing longer text sequences (e.g. Beltagy et al., 2020; Wang et al., 2020) is highly important. Further research progress in this direction would open up the potential of transfer learning with Transformers for a wider range of social science text analyses.

As neural transfer learning with Transformers is the basis of larger developments within AI research (Bommasani et al., 2021), it is important that social scientists understand these new learning modes and models—such that these learning modes and models can be

correctly and fruitfully applied and their risks critically assessed.

Data Availability Statement. The code of this study is openly available in figshare at https://doi.org/10.6084/m9.figshare.14394173.

Appendix to Introduction to Neural Transfer Learning with Transformers for Social Science Text Analysis

2.A Introduction to Deep Learning

This section provides an introduction to the basics of deep learning. First, based on the example of feedforward neural networks the core elements of neural network architectures are explicated. Then, the optimization process via stochastic gradient descent with back-propagation (Rumelhart et al., 1986) will be presented. Subsequently, the architecture of recurrent neural networks (RNNs) (Elman, 1990) is outlined.

2.A.1 Feedforward Neural Network

The most elementary deep learning model is a feedforward neural network (Goodfellow et al., 2016, p. 164). A feedforward neural network with L hidden layers, vector input $\boldsymbol{x}$, and a scalar output y can be visualized as in Figure 2.A.1 and be described as follows:[36]

$$\boldsymbol{h}_1 = \sigma_1(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \tag{2.17}$$

$$\boldsymbol{h}_2 = \sigma_2(\boldsymbol{W}_2\boldsymbol{h}_1 + \boldsymbol{b}_2) \tag{2.18}$$

$$\dots$$

$$\boldsymbol{h}_l = \sigma_l(\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{b}_l) \tag{2.19}$$

$$\dots$$

$$y = \sigma_o(\boldsymbol{w}_o\boldsymbol{h}_L + b_o) \tag{2.20}$$

The input to the neural network is the K_0-dimensional vector $\boldsymbol{x}$ (see Equation 2.17). $\boldsymbol{x}$ enters an affine function characterized by weight matrix $\boldsymbol{W}_1$ and bias vector $\boldsymbol{b}_1$, where

[36] Note that here, in accordance with standard notation in machine learning and NLP (Goldberg, 2016, p. 346), column vectors are used.

$\boldsymbol{W}_1 \in \mathbb{R}^{K_1 \times K_0}$, and $\boldsymbol{b}_1 \in \mathbb{R}^{K_1}$. σ_1 is a nonlinear activation function and $\boldsymbol{h}_1 \in \mathbb{R}^{K_1}$ is the K_1-dimensional representation of the data in the first hidden layer. This is, the neural networks takes the input data $\boldsymbol{x}$ and via combining an affine function with a nonlinear activation function generates a new, transformed representation of the original input: $\boldsymbol{h}_1$. The hidden state $\boldsymbol{h}_1$ in turn serves as the input for the next layer that produces representation $\boldsymbol{h}_2 \in \mathbb{R}^{K_2}$. This continues through the layers until the last hidden representation, $\boldsymbol{h}_L \in \mathbb{R}^{K_L}$, enters the output layer (see Equation 2.20).

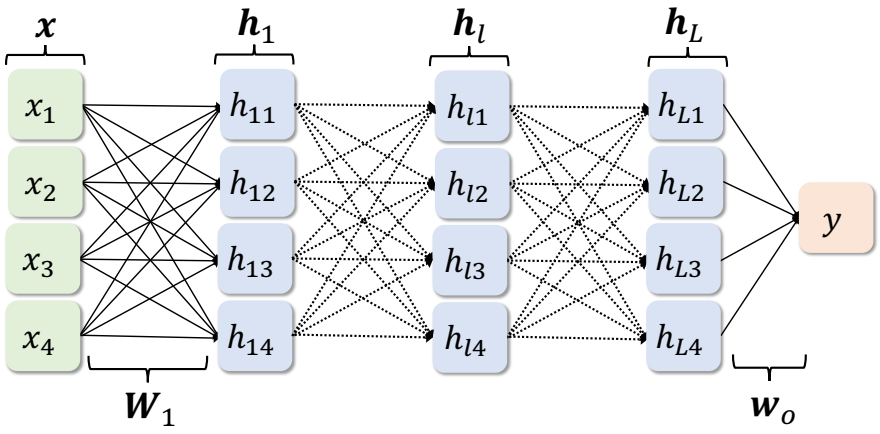


Figure 2.A.1: **A Feedforward Neural Network.** Feedforward neural network with L hidden layers, four units per hidden layer and scalar output y. The solid lines indicate the linear transformations of weight matrix $\boldsymbol{W}_1$. The dotted lines indicate the connections between several consecutive hidden layers.

The activation functions in neural networks are typically chosen to be nonlinear (Goodfellow et al., 2016, p. 168). The reason is that if the activation functions were set to be linear, the output of the neural network would merely be a linear function of $\boldsymbol{x}$ (Goodfellow et al., 2016, p. 168). Hence, the use of nonlinear activation functions is essential for the capacity of neural networks to approximate a wide range of functions and highly complex functions (Ruder, 2019a, p. 31).

In the hidden layers, the Rectified Linear Unit (ReLU) (Nair & Hinton, 2010) is often used as an activation function σ_l (Goodfellow et al., 2016, p. 171). If $\boldsymbol{q} = [q_1, \ldots, q_k, \ldots, q_K]$ is the K-dimensional vector resulting from the affine transformation in the lth hidden layer, $\boldsymbol{q} = \boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l$ (see Equation 2.19), then ReLU is applied on each element q_k:

$$\sigma_l(\boldsymbol{q})_k = max\{0, q_k\} \tag{2.21}$$

$\sigma_l(\boldsymbol{q})_k$ then is the kth element of hidden state vector $\boldsymbol{h}_l$.[37]

[37] Activation functions that are similar to ReLU are the Exponential Linear Unit (ELU) (Clevert et al., 2016), Leaky ReLU and the Gaussian Error Linear Unit (GELU) (Hendrycks & Gimpel, 2016). The latter is used in BERT (Devlin et al., 2019).

In the output layer, the activation function σ_o is selected so as to produce an output that matches the task-specific type of output values. In binary classification tasks with $y_i \in \{0, 1\}$ the standard logistic function, often simply referred to as the sigmoid function, is a common choice (Goodfellow et al., 2016, p. 179-180). For a single observational unit i, the sigmoid function's scalar output value gives the probability that $y_i = 1$. If y_i, however, can assume one out of C unordered response category values, $y_i \in \{\mathcal{G}_1, \ldots, \mathcal{G}_c, \ldots, \mathcal{G}_C\}$, then the softmax function (which is a generalization of the sigmoid function that takes as an input and produces as an output a vector of length C) is typically employed (Goodfellow et al., 2016, p. 180-181). For the ith example, the cth element of the softmax output vector gives the predicted probability that unit i falls into the cth class.

2.A.2 Optimization: Gradient Descent with Backpropagation

In supervised learning tasks, a neural network is provided with input $\boldsymbol{x}_i$ and corresponding output y_i for each training example. All the weights and bias terms are parameters to be learned in the process of optimization (Goodfellow et al., 2016, p. 165). The set of parameters hence is $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_l, \ldots, \boldsymbol{W}_L, \boldsymbol{W}_o, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_l, \ldots, \boldsymbol{b}_L, \boldsymbol{b}_o\}$.

For a single training example i, the loss function $\mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}))$ measures the discrepancy between the value predicted for unit i by model $\mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}})$, that is characterized by the estimated parameter set $\tilde{\boldsymbol{\theta}}$, and the true value y_i (Vapnik, 1991, p. 832). In the optimization process, the aim is to find the set of values for the weights and biases that minimizes the average of the observed losses over all training set instances, also known as the empirical risk: $\mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, \mathsf{f}(\boldsymbol{x}_i, \tilde{\boldsymbol{\theta}}))$ (Goodfellow et al., 2016, p. 272-273).

Neural networks commonly employ variants of gradient descent with backpropagation in the optimization process (Goodfellow et al., 2016, p. 173). To approach the local minimum of the empirical risk function, the gradient descent algorithm makes use of the fact that the direction of the negative gradient of function $\mathcal{R}_{emp}$ at current point $\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}$ gives the direction in which $\mathcal{R}_{emp}$ is decreasing fastest—the direction of the steepest descent (Goodfellow et al., 2016, p. 83). The gradient is a vector of partial derivatives. It is the derivative of $\mathcal{R}_{emp}$ at point $\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}$ and is denoted as $\nabla_{\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}} \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}})$ (Johnson, 2017, p. 2).

In the jth iteration, the gradient descent algorithm computes the negative gradient of $\mathcal{R}_{emp}$ at current point $\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}$ and then changes its position from $\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}$ into the direction of the negative gradient (Goodfellow et al., 2016, p. 83-84):

$$\tilde{\boldsymbol{\theta}}_{\boldsymbol{j+1}} = \tilde{\boldsymbol{\theta}}_{\boldsymbol{j}} - \eta \nabla_{\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}} \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}) \tag{2.22}$$

where $\eta \in \mathbb{R}_+$ is the learning rate. If η is small enough, then $\mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}}) \geq \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_{\boldsymbol{j+1}}) \geq \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_{\boldsymbol{j+2}}) \geq \ldots$. This is, repeatedly updating into the direction of the negative gradient with a suitably small learning rate η, will generate a sequence moving toward the local minimum (Li et al., 2020a).

In each iteration, the gradients for all parameters are computed via the backpropagation algorithm (Rumelhart et al., 1986).[38] A very frequently employed approach, known as mini-batch stochastic gradient descent, is to compute the gradients based on a random sample, a mini-batch, of S training set observations (Goodfellow et al., 2016, p. 275-276, 291):

$$\nabla_{\tilde{\boldsymbol{\theta}}_j} \mathcal{R}_{emp}(\tilde{\boldsymbol{\theta}}_j) = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\tilde{\boldsymbol{\theta}}_j} \mathcal{L}(y_s, \mathsf{f}(\boldsymbol{x}_s, \tilde{\boldsymbol{\theta}}_j)) \tag{2.23}$$

The learning rate η and the size of the mini-batch S are hyperparameters in training neural networks. Especially the learning rate is often attended to carefully (Li et al., 2020a). A too high learning rate leads to large fluctuations in the loss function values, whereas a too low learning rate implies slow convergence and risks that the learning process does not move away from a non-optimal region with a high loss value (Goodfellow et al., 2016, p. 291). Commonly, the learning rate is set to vary over the course of the training process (Goodfellow et al., 2016, p. 290-291). Furthermore, there are variants of stochastic gradient descent, e.g. AdaGrad (Duchi et al., 2011), RMSProp (Hinton et al., 2012), and Adam (Kingma & Ba, 2015), that have a different learning rate for each parameter (Goodfellow et al., 2016, p. 303-305).

2.A.3 Recurrent Neural Networks

The recurrent neural network (RNN) (Elman, 1990) is the most basic neural network to process sequential input data of variable length such as texts (Goodfellow et al., 2016, p. 367). Given an input sequence of T input embeddings $(\boldsymbol{z}_{[a_1]}, \ldots, \boldsymbol{z}_{[a_t]}, \ldots, \boldsymbol{z}_{[a_T]})$, RNNs sequentially process each token. Here, one input embedding $\boldsymbol{z}_{[a_t]}$ corresponds to one time step t and the hidden state $\boldsymbol{h}_t$ is updated at each time step. At each step t, the hidden state $\boldsymbol{h}_t$ is a function of the hidden state generated in the previous time step, $\boldsymbol{h}_{t-1}$, and new input data, $\boldsymbol{z}_{[a_t]}$ (see Figure 2.A.2) (Elman, 1990; Amidi & Amidi, 2019).

The hidden states $\boldsymbol{h}_t$, that are passed on and transformed through time, serve as the model's memory (Elman, 1990, p. 182; Ruder, 2019a, p. 32). They capture the information of the sequence that entered until t (Goldberg, 2016, p. 391). Due to this sequential architecture, RNNs theoretically can model dependencies over the entire range of an input sequence (Amidi & Amidi, 2019). But in practice, recurrent models have problems learning dependencies that extend beyond sequences of 10 or 20 tokens (Goodfellow et al., 2016, p. 396-399). The reason is that when backpropagating the gradients through the time steps (Backpropagation Through Time (BPTT)), the gradients may vanish and thus fail to transmit a signal over long ranges (Goodfellow et al., 2016, p. 396-399).

[38] The backpropagation algorithm makes use of the chain rule to compute the gradients. Helpful introductions to the backpropagation algorithm can be found in Li et al. (2020a), Li et al. (2020b) and Hansen (2019).

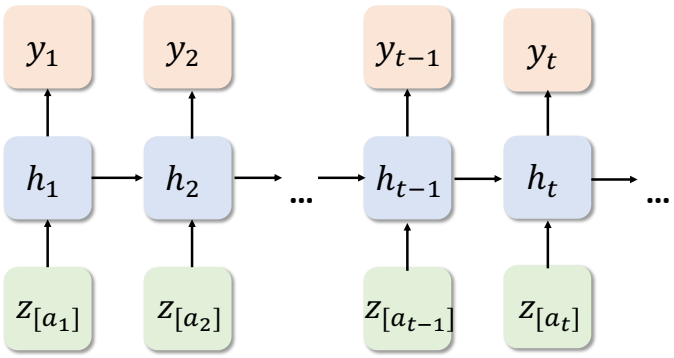


Figure 2.A.2: **A Recurrent Neural Network.** Architecture of a basic RNN unfolded through time. At time step t, the hidden state $\boldsymbol{h}_t$ is a function of the previous hidden state, $\boldsymbol{h}_{t-1}$, and current input embedding $\boldsymbol{z}_{[a_t]}$. $\boldsymbol{y}_t$ is the output produced at t.

The long short-term memory (LSTM) model (Hochreiter & Schmidhuber, 1997) extends the RNN with input, output, and forget gates that enable the model to accumulate, remember, and forget provided information (Goldberg, 2016, p. 399-400). This makes LSTMs better suited than the basic RNNs to model dependencies stretching over long time spans (Goldberg, 2016, p. 399-400).

2.B Zero-Shot Learning and the GPT-3

Ultimately the aim of the strand of NLP research focusing on zero-shot learning is to have a model that generalizes well to a wide spectrum of target tasks without being explicitly trained on the target tasks (Radford et al., 2019; Brown et al., 2020; Davison, 2020b). The work on the series of GPT models—OpenAI GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), and especially GPT-3 (Brown et al., 2020)—has demonstrated that large models that are pretrained on language modeling tasks on excessively large corpora can sometimes come close to achieving acceptable prediction performances without fine-tuning (i.e. without gradient updates) on target task-specific examples (Brown et al., 2020, p. 4). So far, the key to increasing the zero-shot no-fine-tuning learning performances seems to be an increase in the models' capacity to learn complex functions as determined by the number of model parameters (Brown et al., 2020, p. 4). (Whilst the original OpenAI GPT comprises 117 million parameters, GPT-2 has 1,542 million (Radford et al., 2019, p. 4) and GPT-3 has 175,000 million parameters (Brown et al., 2020, p. 1, 8).) Additionally, and in correspondence with an increase in model parameters, the size of the employed training corpora increases rapidly as well (Radford et al., 2019, p. 3; Brown et al., 2020, p. 5). Yet given its sheer size, re-training the GPT-3 is prohibitively expensive (Brown et al., 2020, p. 9; Riedl, 2020). Moreover, whereas typically the source code of pretrained

language models is open sourced by the companies (e.g. Google, Facebook, Microsoft) that developed these models, OpenAI decided not to share the code on GPT-3 and instead allows using GPT-3 for downstream tasks via an API, thereby raising questions regarding accessibility and replicability of pretrained language models for research (Brockman et al., 2020; Riedl, 2020).

2.C Subword Tokenization Algorithms

Subword tokenization algorithms try to find a balance between word-level tokenization (which tends to result in a large vocabulary—and hence a large embedding matrix that consumes a lot of memory) and character-level tokenization (which generates a small and flexible vocabulary but does not yield as well-performing representations of text) (Radford et al., 2019, p. 4; Hugging Face, 2020c). Subword tokenization algorithms typically result in vocabularies in which frequently occurring character sequences are merged to form words whereas less common character sequences become subwords or remain separated as single characters (Radford et al., 2019, p. 4; Hugging Face, 2020c). The Byte-Pair Encoding (BPE) algorithm and variants thereof are subword tokenization algorithms employed in many Transformer-based models (e.g. Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019). The base BPE algorithm starts with a list of all the unique characters in a corpus and then learns to merge the characters into longer character sequences (eventually forming subwords and words) until the desired vocabulary size is reached (Sennrich et al., 2016, p. 1717-1718). In the WordPiece variant of BPE, the algorithm merges at each step the character pair that, when merged, results in the highest increase in the likelihood of the training corpus compared to all other pairs (Schuster & Nakajima, 2012, p. 5150).

2.D Pretraining BERT

In the masked language modeling pretraining task, for each token q, that has been sampled for prediction, the updated token representation produced by the last encoder $\boldsymbol{h}_q^*$ is fed into a single-layer feedforward neural network with a softmax output layer to generate a probability distribution over the terms in the vocabulary predicting the term corresponding to q (see Figure 2.7 in the main article) (Alammar, 2018a; Devlin et al., 2019, p. 4174). For the next sentence prediction task, the representation for the *[CLS]* token, $\boldsymbol{h}_1^*$, is processed via a single-layer feedforward neural network with a softmax output to give the predicted probability of the second segment succeeding the first segment (see Figure 2.7 in the main article) (Alammar, 2018a; Devlin et al., 2019, p. 4174). The loss function in pretraining is the sum of the average loss from the masked language modeling task and the average loss from next sentence prediction (Devlin et al., 2019, p. 4183).

In order to learn the parameters in pretraining, the authors use the Adam algorithm, a variant of stochastic gradient descent, in which at the jth iteration for each individual parameter the estimate of the gradient's average for this parameter is updated based on a parameter-specific learning rate (Kingma & Ba, 2015; Devlin et al., 2019, p. 4183).[39] They use a learning rate schedule in which the global Adam learning rate (that is individually adapted per parameter) linearly increases during the first 10,000 iterations (the warmup) to reach a maximum value of 1e-4 and then is linearly decaying (Devlin et al., 2019, p. 4183). They furthermore regularize by employing an L^2 weight decay (Goodfellow et al., 2016, p. 226; Devlin et al., 2019, p. 4183). As an additional regularization strategy they use dropout (Srivastava et al., 2014) with dropout probability $p = 0.1$ (Devlin et al., 2019, p. 4183). In dropout, units and their corresponding connections are randomly dropped during training (Srivastava et al., 2014, p. 1929). Devlin et al. (2019, p. 4183) select a mini-batch size of 256 sequences and conduct 1,000,000 iterations, which implies that they train the model for around 40 epochs; i.e. they make around 40 passes over the entire 3.3 billion token pretraining data set.

2.E Additional Examples for Autoencoding Models: ALBERT and ELECTRA

ALBERT (Lan et al., 2020) aims at a parameter efficient design. By decoupling the size of the input embedding layers from the size of the hidden layers and by sharing parameters across all layers, ALBERT substantially reduces the number of parameters to be learned (e.g. by a factor of 18 comparing ALBERT-Large to $\text{BERT}_{\text{LARGE}}$) (Lan et al., 2020, p. 2, 4, 6). Parameter reduction has regularizing effects, and—because it saves computational resources—allows to construct a deeper model with more and/or larger hidden layers whose increased capacity benefits performance on target tasks while still comprising fewer parameters than the original $\text{BERT}_{\text{LARGE}}$ (Lan et al., 2020, p. 2, 7).

Whereas BERT, RoBERTa, and ALBERT make use of the masked language modeling task, ELECTRA introduces a new, more resource-efficient pretraining objective, named replaced token detection (Clark et al., 2020, p. 1). ELECTRA addresses the issue that in masked language modeling for each input sequence predictions are made only for those 15% of tokens that have been sampled for the task, thereby reducing the amount of what could be learned from each training sequence (Clark & Luong, 2020). In pretraining, ELECTRA has to predict for each input token in each sequence whether the token comes

[39] Here the individual learning rate is inversely proportional to the average of the squared gradient—such that the learning rate is smaller for large gradients and higher for smaller gradients (Goodfellow et al., 2016, p. 303-306). The gradient's average and the squared gradient's average are exponentially weighted moving averages with decay rates $\beta_1, \beta_2 \in [0, 1)$ to assign an exponentially decaying weight to gradients from long ago iterations (Kingma & Ba, 2015, p. 2; Goodfellow et al., 2016, p. 303-306). Devlin et al. (2019, p. 4183) set β_1 to 0.9 and β_2 to 0.999.

from the original sequence or has been replaced by a plausible fake token (Clark & Luong, 2020; Clark et al., 2020, p. 1, 3). Thus, ELECTRA (the discriminator) solves a binary classification task for each token and is much more efficient in pretraining requiring fewer computational resources (Clark & Luong, 2020; Clark et al., 2020, p. 3). The plausible fake tokens come from a generator that is trained on a masked language modeling task together with the ELECTRA discriminator (Clark et al., 2020, p. 3). After pretraining, the generator is removed and only the ELECTRA discriminator is used for fine-tuning (Clark & Luong, 2020).

2.F Additional Example for an Autoregressive Model: The XLNet

Strictly speaking, XLNet (Yang et al., 2019) is not an autoregressive model (Hugging Face, 2020b). Yet the permutation language modeling objective that it introduces builds on the autoregressive language modeling framework (Yang et al., 2019, p. 5756). The authors of XLNet seek a pretraining objective that learns bidirectional representations as in autoencoding models whilst overcoming problems of autoencoding representations: first, the pretrain-finetune discrepancy that results from the fact that *'[MASK]'* tokens only occur in pretraining, and, second, the assumption that the tokens selected for the masked language modeling task in one sequence are independent of each other (Yang et al., 2019, p. 5754-5755). Given a sequence whose tokens are indexed $(1, \ldots, T)$, the permutation language modeling objective makes use of the permutations of the token index $(1, \ldots, T)$ (Yang et al., 2019, p. 5756). For each possible permutation of $(1, \ldots, T)$, the task is to predict the next token in the permutation order given the previous tokens in the permutation (Yang et al., 2019, p. 5756). In doing so, the learned token representations can access information from left and right contexts whilst the autoregressive nature of the modeling objective avoids the pretrain-finetune discrepancy and the independence assumption (Yang et al., 2019, p. 5756).

2.G Examples for Sequence-to-Sequence Models: The T5 and BART

The T5 (Raffel et al., 2020) is very close to the original Transformer encoder-decoder architecture. It is based on the idea to consider all NLP tasks as text-to-text problems (Raffel et al., 2020, p. 2-3). To achieve this, each input sequence that is fed to the model is preceded by a task-specific prefix, that instructs the model what to do. For example (Raffel et al., 2020, p. 47ff.): A translation task in this scheme has the input *'translate from English to German: I love this movie.'* and the model is trained to output *'Ich liebe diesen*

Film.'. For a sentiment classification task on the SST-2 Dataset (Socher et al., 2013), the input would be: *'sst2 sentence: I love this movie.'* and the model is trained to predict one of *'positive'* or *'negative'*. The fact that there is a shared scheme for all NLP tasks, allows the T5 to be pretrained on a multitude of different NLP tasks before being fine-tuned on a specific target task (Raffel et al., 2020, p. 30-33). In the multitask pretraining mode, T5 is trained on a self-supervised objective similar to the masked language modeling task in BERT as well as various different supervised tasks (such as translation or natural language inference) (Raffel et al., 2020, p. 37). With this multitask pretraining setting, in which the parameters learned in pretraining are shared across different tasks, the T5, rather than being a standard sequential transfer learning model, implements a softened version of multitask learning (Raffel et al., 2020, p. 30).

BART (Lewis et al., 2020)—another well-known sequence-to-sequence model—is composed of an encoder and a decoder. Just as an autoencoding model, BART in pretraining is presented with a corrupted sequence and has to predict the original uncorrupted sequence (Lewis et al., 2020, p. 2). In pretraining, BART allows a wide range of different types of corrupting operations to be applied to the documents (Lewis et al., 2020, p. 2-3). Due to this autoencoding-style pretraining task, BART can be considered a BERT-like bidirectional encoder followed by an autoregressive unidirectional decoder (Lewis et al., 2020, p. 1-2). Because of the decoder, that learns to predict output tokens in an autoregressive manner, BART is better suited to perform text generation tasks than regular autoencoding models (Lewis et al., 2020, p. 1, 6). Additionally, BART performs similarly to RoBERTa on discriminative natural language understanding tasks (Lewis et al., 2020, p. 1, 6).

2.H Interpretability

One common method to make neural networks more interpretable is probing (also known as auxiliary prediction tasks) (Belinkov & Glass, 2019, p. 51). In probing, an element of a trained neural network (e.g. a set of contextualized token embeddings) is extracted, fixed, and fed into a simple classifier and then is applied to some task (e.g. POS tagging, coreference resolution) (Belinkov & Glass, 2019, p. 51; Tenney et al., 2019b, p. 1-3). If the prediction performance on the task is high, then this is taken as an indication that information required to address the task (e.g. syntactic information for POS tagging, semantic information for coreference resolution) is encoded in the tested element of the network (Belinkov & Glass, 2019, p. 51; Tenney et al., 2019b, p. 2). Accordingly, probing is one way to inspect what information a neural network has learned and which elements capture which information.

Another important aspect of interpretability is to assess the importance of input features for predicted outputs. The open-source library Captum (https://captum.ai/) implements several attribution algorithms that allow just that (Kokhlikyan et al., 2020, p. 3). Additionally, algorithms for attributing outputs to a hidden layer, as well as algorithms for

attributing hidden layer values to feature inputs, are also provided (Kokhlikyan et al., 2020, p. 3).

Most attribution algorithms can be considered as either being gradient-based or perturbation-based (Agarwal et al., 2021, p. 110). Perturbation-based algorithms make use of removed or altered input features to learn about the importance of input features (Ancona et al., 2018, p. 2; Agarwal et al., 2021, p. 110). Gradient-based algorithms make use of the gradient of the predicted output with regard to input features (Ancona et al., 2018, p. 2-3; Agarwal et al., 2021, p. 110).

For models that incorporate attention mechanisms, the analysis of patterns in attention weights α_{t,t^*} and the probing of attention heads is another interpretability-related research area (see e.g. Clark et al., 2019; Kobayashi et al., 2020).Tools for interpreting attention matrices are also provided by Captum (see, for example, the tutorial at https://captum.ai/tutorials/Bert_SQUAD_Interpret2).

Another set of methods related to interpretability is behavioral testing and the construction of adversarial examples (Belinkov & Glass, 2019, p. 54-58).[40] Here, the goal is to inspect a trained model's behavior when confronted with a challenging set of inputs or adversarial examples. In an award-winning paper, Ribeiro et al. (2020) present a methodology for behavioral testing. Their research findings emphasize that whilst the performance of NLP models as evaluated via accuracy measures on held-out test sets has risen substantially during the last years (see also e.g. Wang et al., 2019, p. 2), when evaluating the models via behavioral testing, it is revealed that accuracy-based performances on common benchmark data sets overestimate the models' linguistic and language understanding capabilities. BERT, for example, is found to have high failure rates for simple negation tests (e.g. classifying 84.4% of positive or neutral tweets in which a negative sentiment expression is negated into the negative category) (Ribeiro et al., 2020, p. 4905-4907).

2.I Deep Learning and Transfer Learning in Practice

To practically implement deep learning models, it is advisable to have access to a graphics processing unit (GPU). In contrast to a central processing unit (CPU), a GPU comprises many more cores and can conduct thousands of operations in parallel (Caulfield, 2009). GPUs thus handle tasks that can be broken down into smaller, simultaneously executable subtasks much more efficiently than CPUs (Caulfield, 2009). When training a neural network via stochastic gradient descent, every single hidden unit within a layer usually can be updated independently of the other hidden units in the same layer (Goodfellow et al., 2016, p. 440). Hence, neural networks lend themselves to parallel processing.

[40] In NLP, an adversarial example is a text sequence d_i^* that is close to a sequence d_i but has a different class label than d_i (Belinkov & Glass, 2019, p. 56).

A major route to access and use GPUs is via NVIDIA's CUDA framework (Goodfellow et al., 2016, p. 440-441). But instead of additionally learning how to write CUDA code, researchers use libraries that enable CUDA GPU processing (Goodfellow et al., 2016, p. 441). As of today, PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2016) are the most commonly used libraries that allow training neural networks via CUDA-enabled GPUs. Both libraries have Python interfaces. Therefore, to efficiently train deep learning models via GPU acceleration, researchers can use a programming language they are familiar with.

Another obstacle is having a GPU at hand that can be used for computation. The computing infrastructures of universities and research institutes typically provide their members access to GPU facilities. Free GPU usage also is available via Google Colaboratory (or Colab for short): https://colab.research.google.com/notebooks/intro.ipynb. Colab is a computing service that allows its user to run Python code via the browser (Google Colaboratory, 2020). Here, GPUs can be used free of cost. The free resources, however, are not guaranteed and there may be usage limits. One issue researchers have to keep in mind when using Colab is that at each session another type of GPU may be assigned. Documenting the used computing environment hence is vital to ensure traceability. Note, that full reproducibility *across* different computing platforms and *across* different versions of PyTorch and TensorFlow cannot be guaranteed (Freidank, 2020; Torch Contributors, 2021). However, there are measures that researchers can undertake to minimize nondeterministic elements (e.g. not using nondeterministic algorithms where possible and ensuring that batch allocation is reproducible) (see Torch Contributors, 2021).

2.J Application: Zero-Shot Learning

To explore how pretrained Transformer-based models would perform in a zero-shot learning setting, the approach of Yin et al. (2019) is followed. Yin et al. (2019, p. 3918-3919) frame zero-shot text classification as a natural language inference (NLI) task. In NLI, a model is presented with a premise and a hypothesis and then has to decide whether the hypothesis is true given the premise (ENTAILMENT), whether the hypothesis is false given the premise (CONTRADICTION), or whether the hypothesis is neither true nor false given the premise (NEUTRAL) (see Table 2.J.1) (Williams et al., 2018, p. 1112-1113).

In the zero-shot classification framework of Yin et al. (2019), a model is presented with the input text (taking the role of the premise) and a hypothesis that asks whether the input text belongs to a particular class. The model then has to predict whether this is the case or not. The model that Yin et al. (2019) use for zero-shot learning is a BERT model that has been trained on three different NLI data sets (Yin et al., 2019, p. 3919). (The NLI data sets are, of course, unrelated to the target tasks Yin et al. (2019) use for zero-shot learning evaluation.)

Here, in a similar approach, two pretrained Transformer-based models—RoBERTa (Liu et al., 2019) and BART (Lewis et al., 2020)—that have been further trained on the Multi-Genre Natural Language Inference (MNLI) data set (Williams et al., 2018) are used as models for zero-shot learning. The models are accessed from Hugging Face's Transformers and then are used in a zero-shot-classification pipeline that is based on an NLI-framework (see Davison, 2020a). For an illustration see Table 2.J.1.

	[CLS] ... Premise ... [SEP] ... Hypothesis ... [SEP]	predict one out of
NLI	[CLS] I am a lacto-vegetarian. [SEP] I eat one egg per week. [SEP]	ENTAILMENT, CONTRADICT., NEUTRAL
ZSL	[CLS] Ok, I see what you mean. [SEP] This comment is <u>toxic</u>. [SEP]	ENTAILMENT, CONTRADICT., NEUTRAL
	[CLS] Ok, I see what you mean. [SEP] This comment is <u>not toxic</u>. [SEP]	ENTAILMENT, CONTRADICT., NEUTRAL

Table 2.J.1: **Scheme in Natural Language Inference (NLI) and Zero-Shot Learning (ZSL).** In NLI, the model is provided with a premise followed by a hypothesis and has to decide whether the hypothesis is true (ENTAILMENT), false (CONTRADICTION), or neutral (NEUTRAL) given the premise. In ZSL, the sequence for which a prediction is to be made constitutes the premise. Each class label is presented to the model as a separate hypothesis. Here, the input sequence is "Ok, I see what you mean." and there are $C = 2$ class labels, namely: {toxic, not toxic}. For each sequence-hypothesis pair, the model has to predict one out of {ENTAILMENT, CONTRADICTION, NEUTRAL}.

An important point to note is that the zero-shot performance will also depend on the textual formulation of the hypothesis the model is presented with (Yin et al., 2019, p. 3921-3922). (The model takes as an input the text sequence for which a prediction is to be made followed by the hypothesis and thus the model learns representations for the input sequence and the hypothesis and will generate a prediction based on (the compatibility) of both inputs (Davison, 2020b).) Here, to explore the effect of different hypothesis formulations, two different hypothesis formulations are tried in each application (see Tables 2.J.2 and 2.J.3).

Ethos

Model	Hypothesis Formulation	Class Labels	F_1 (macro)
RoBERTa	The statement {} the ethos of another politician or party.	{attacks, does not refer to, supports}	0.200
RoBERTa	The statement expresses {} sentiment toward the character of another politician or party.	{negative, no, positive}	0.188
BART	The statement {} the ethos of another politician or party.	{attacks, does not refer to, supports}	0.184
BART	The statement expresses {} sentiment toward the character of another politician or party.	{negative, no, positive}	0.181

Abortion

Model	Hypothesis Formulation	Class Labels	F_1 (macro)
RoBERTa	The text is {} legalization of abortion.	{in favor of, neutral toward, against}	0.344
RoBERTa	The text expresses a {} stance toward legalization of abortion.	{positive, neutral, negative}	0.362
BART	The text is {} legalization of abortion.	{in favor of, neutral toward, against}	0.455
BART	The text expresses a {} stance toward legalization of abortion.	{positive, neutral, negative}	0.368

Table 2.J.2: **ZSL Results I.** Macro-averaged F_1-Scores obtained via zero-shot learning for the test sets of the Ethos and Abortion classification tasks. In each application, two pretrained models (RoBERTa and BART both trained on the MNLI) and two hypothesis formulations are explored. Gray colored numbers highlight the best performing model-hypothesis formulation combination for the task.

Toxic

Model	Hypothesis Formul.	Class Labels	F_1 (macro)
RoBERTa	This comment is {}.	{not toxic, toxic}	0.470
RoBERTa	This comment is {}.	{{neither obscene, nor threatening, nor insulting, and does not expresses hatred toward social groups and identities}, {obscene, or threatening, or insulting, or expresses hatred toward social groups and identities}}	0.213
BART	This comment is {}.	{not toxic, toxic}	0.378
BART	This comment is {}.	{{neither obscene, nor threatening, nor insulting, and does not expresses hatred toward social groups and identities}, {obscene, or threatening, or insulting, or expresses hatred toward social groups and identities}}	0.178

Table 2.J.3: **ZSL Results II.** Macro-averaged F_1-Scores obtained via zero-shot learning for one sampled test set (N = 1,000) of the Toxic classification task. Two pretrained models (RoBERTa and BART both trained on the MNLI) and two hypothesis formulations are explored. Gray colored numbers highlight the best performing model-hypothesis formulation combination for the task.

Moreover, note that if one has a target task with C class labels such that $y_i \in \{\mathcal{G}_1, \ldots, \mathcal{G}_c, \ldots, \mathcal{G}_C\}$, then each class label is presented to the model as one separate hypothesis (Davison, 2020a). Consequently, if there are, for example, $C = 4$ labels, the model will be fed with $C = 4$ different sequence-hypothesis pairs for each sequence. In the implementation in Hugging Face's zero-shot-classification pipeline, the model generates for each of the C sequence-

hypothesis pairs a prediction to belong to one of {ENTAILMENT, CONTRADICTION, NEUTRAL} (Davison, 2020a). Then, in order to aggregate the C separate predictions into a single one, the predicted score for ENTAILMENT is extracted for each hypothesis. Together, the C entailment scores serve as the input to a softmax function that returns a C-dimensional vector of predicted probabilities (which sum to one) and the cth element gives the probability that the sequence belongs to the cth class (Davison, 2020a).

The zero-shot classification results are presented in Tables 2.J.2 and 2.J.3. For each application, for each combination of an employed pretrained model and an explored hypothesis formulation, the macro-averaged F_1-Score for the test set is reported. Across applications, models, and hypothesis formulations, the achieved performance levels are mediocre compared to the macro-averaged F_1-Scores reached by the fine-tuned models (which are presented in Table 2.1 in the main article). Interestingly, the smallest reduction in performance compared to the fine-tuned models can be observed for the Abortion application. In contrast to the other two applications, that seek to measure concepts from text that are relatively difficult to adequately describe in words (ethos, toxicity), hypothesis formulation in the legalization of abortion application is relatively straightforward.

Data Availability Statement. Just as for the other analyses presented in this paper, the code for this zero-shot learning implementation is openly available in figshare at https://doi.org/10.6084/m9.figshare.14394173.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv. https://arxiv.org/abs/1603.04467

Agarwal, S., Jabbari, S., Agarwal, C., Upadhyay, S., Wu, S., & Lakkaraju, H. (2021). Towards the unification and robustness of perturbation and gradient based explanations. *Proceedings of Machine Learning Research*, 139, 110–119. https://proceedings.mlr.press/v139/agarwal21c.html

Ahlquist, J. S. & Breunig, C. (2012). Model-based clustering and typologies in the social sciences. *Political Analysis*, *20*(1), 92–112. https://doi.org/10.1093/pan/mpr039

Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In E. M. Bender, L. Derczynski, & P. Isabelle (Eds.), *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 1638–1649). Association for Computational Linguistics. https://aclanthology.org/C18-1139

Alammar, J. (2018a). The illustrated BERT, ELMo, and co. (How NLP cracked transfer learning). *Jay Alammar*. http://jalammar.github.io/illustrated-bert/

Alammar, J. (2018b). The illustrated Transformer. *Jay Alammar*. http://jalammar.github.io/illustrated-transformer/

Alammar, J. (2018c). Visualizing a neural machine translation model (mechanics of seq2seq models with attention). *Jay Alammar*. https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Amidi, A. & Amidi, S. (2019). *Recurrent neural networks cheatsheet* [Lecture Notes]. Stanford University. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Amsalem, E., Fogel-Dror, Y., Shenhav, S. R., & Sheafer, T. (2020). Fine-grained analysis of diversity levels in the news. *Communication Methods and Measures*, *14*(4), 266–284. https://doi.org/10.1080/19312458.2020.1825659

Anastasopoulos, L. J. & Bertelli, A. M. (2020). Understanding delegation through machine learning: A method and application to the European Union. *American Political Science Review*, *114*(1), 291–301. https://doi.org/10.1017/S0003055419000522

Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. (2018). Towards better understanding of gradient-based attribution methods for deep neural networks. In Y. Bengio & Y.

LeCun (Eds.), *6th International Conference on Learning Representations (ICLR 2018)*. https://openreview.net/pdf?id=Sy21R9JAW

Aribandi, V., Tay, Y., Schuster, T., Rao, J., Zheng, H. S., Mehta, S. V., Zhuang, H., Tran, V. Q., Bahri, D., Ni, J., Gupta, J., Hui, K., Ruder, S., & Metzler, D. (2022). Ext5: Towards extreme multi-task scaling for transfer learning. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=Vzh1BFUCiIX

Aßenmacher, M. & Heumann, C. (2020). On the comparability of pre-trained language models. In S. Ebling, D. Tuggener, M. Hürlimann, M. Cieliebak, & M. Volk (Eds.), *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS 2020)*. CEUR-WS.org. http://ceur-ws.org/Vol-2624/paper2.pdf

Aßenmacher, M., Schulze, P., & Heumann, C. (2021). Benchmarking down-scaled (not so large) pre-trained language models. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 14–27). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.2

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. arXiv. https://arxiv.org/abs/1607.06450

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., & Auli, M. (2021). XLS-R: Self-supervised cross-lingual speech representation learning at scale. arXiv. https://arxiv.org/abs/2111.09296

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 12449–12460). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. https://arxiv.org/abs/1409.0473

Bapna, A., Chung, Y.-a., Wu, N., Gulati, A., Jia, Y., Clark, J. H., Johnson, M., Riesa, J., Conneau, A., & Zhang, Y. (2021). SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training. arXiv. https://arxiv.org/abs/2110.10329

Barberá, P., Boydstun, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated text classification of news articles: A practical guide. *Political Analysis*, *29*(1), 19–42. https://doi.org/10.1017/pan.2020.8

Belinkov, Y. & Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7, 49–72. https://doi.org/10.1162/tacl_a_00254

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document Transformer. arXiv. https://arxiv.org/abs/2004.05150

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155. https://www.jmlr.org/papers/v3/bengio03a.html

Benoit, K. (2020). Text as data: An overview. In L. Curini & R. Franzese (Eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations* (pp. 461–497). SAGE Publications. https://www.doi.org/10.4135/9781526486387.n29

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. https://doi.org/10.1162/tacl_a_00051

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., & et al. (2021). On the opportunities and risks of foundation models. arXiv. https://arxiv.org/abs/2108.07258

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)* (pp. 144–152). Association for Computing Machinery. https://doi.org/10.1145/130385.130401

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, *16*(3), 199–215.

Brockman, G., Murati, M., Welinder, P., & OpenAI (2020). OpenAI API. *OpenAI*. https://openai.com/blog/openai-api/

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish,

S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. arXiv. https://arxiv.org/abs/2005.14165

Brownlee, J. (2019). How to control the stability of training neural networks with the batch size. *Machine Learning Mastery*. https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/

Brownlee, J. (2020). Random oversampling and undersampling for imbalanced classification. *Machine Learning Mastery*. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Budhwar, A., Kuboi, T., Dekhtyar, A., & Khosmood, F. (2018). Predicting the vote using legislative speech. In A. Zuiderwijk & C. C. Hinnant (Eds.), *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age* (pp. 1–10). Association for Computing Machinery. https://doi.org/10.1145/3209281.3209374

Caulfield, B. (2009). What's the difference between a CPU and a GPU? *blogs.nvidia*. https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

Ceron, A., Curini, L., & Iacus, S. M. (2015). Using sentiment analysis to monitor electoral campaigns: Method matters—Evidence from the United States and Italy. *Social Science Computer Review*, *33*(1), 3–20. https://doi.org/10.1177/0894439314521983

Ceron, A., Curini, L., Iacus, S. M., & Porro, G. (2014). Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. *New Media & Society*, *16*(2), 340–358. https://doi.org/10.1177/1461444813480466

Chang, C. & Masterson, M. (2020). Using word order in political text classification with long short-term memory models. *Political Analysis*, *28*(3), 395–411. https://doi.org/10.1017/pan.2019.46

Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In B. Krishnapuram & M. Shah (Eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse Transformers. arXiv. https://arxiv.org/abs/1904.10509

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1179

Chollet, F. (2021). *Deep learning with Python (2nd ed.)*. Manning Publications. https://www.manning.com/books/deep-learning-with-python-second-edition

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. In T. Linzen, G. Chrupała, Y. Belinkov, & D. Hupkes (Eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (pp. 276–286). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4828

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. In A. Rush (Ed.), *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net. https://openreview.net/forum?id=r1xMH1BtvB

Clark, K. & Luong, T. (2020). More efficient NLP model pre-training with ELECTRA. *Google AI Blog*. https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html

Clevert, D., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by Exponential Linear Units (ELUs). In Y. Bengio & Y. LeCun (Eds.), *4th International Conference on Learning Representations (ICLR 2016)*. https://arxiv.org/abs/1511.07289

Colleoni, E., Rozza, A., & Arvidsson, A. (2014). Echo chamber or public sphere? Predicting political orientation and measuring political homophily in Twitter using big data. *Journal of Communication*, *64*(2), 317–332. https://doi.org/10.1111/jcom.12084

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440–8451). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. arXiv. https://arxiv.org/abs/1901.02860

Davison, J. (2020a). New pipeline for zero-shot text classification. *discuss.huggingface.co*. Retrieved December 28, 2021 from https://discuss.huggingface.co/t/new-pipeline-for-zero-shot-text-classification/681

Davison, J. (2020b). Zero-shot learning in modern NLP. *Joe Davison Blog*. https://joeddav.github.io/blog/2020/05/29/ZSL.html

Denny, M. J. & Spirling, A. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, *26*(2), 168–189. https://doi.org/10.1017/pan.2017.44

Devlin, J. (2019). Multilingual BERT. *github.com/google-research*. https://github.com/google-research/bert/blob/master/multilingual.md

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. arXiv. https://arxiv.org/abs/1810.04805v1

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Di Cocco, J. & Monechi, B. (2021). How populist are parties? Measuring degrees of populism in party manifestos using supervised machine learning. *Political Analysis*, (pp. 1–17). https://doi.org/10.1017/pan.2021.29

Diermeier, D., Godbout, J.-F., Yu, B., & Kaufmann, S. (2011). Language and ideology in Congress. *British Journal of Political Science*, *42*(1), 31–55. https://doi.org/10.1017/S0007123411000160

Dixon, L. (2017). Hi! and welcome to our first toxicity classification challenge. *kaggle.com*. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/46064

D'Orazio, V., Landis, S. T., Palmer, G., & Schrodt, P. (2014). Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Political Analysis*, *22*(2), 224–242. https://doi.org/10.1093/pan/mpt030

Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv. https://arxiv.org/abs/1702.08608

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159. https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

Duthie, R. & Budzynska, K. (2018). A deep modular RNN approach for ethos mining. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial*

Intelligence, IJCAI-18 (pp. 4041–4047). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2018/562

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1016/0364-0213(90)90002-E

Ennser-Jedenastik, L. & Meyer, T. M. (2018). The impact of party cues on manual coding of political texts. *Political Science Research and Methods*, *6*(3), 625–633. https://doi.org/10.1017/psrm.2017.29

Firth, J. R. (1957). *Studies in linguistic analysis*. Publications of the Philological Society. Blackwell.

Fowler, E. F., Franz, M. M., Martin, G. J., Peskowitz, Z., & Ridout, T. N. (2021). Political advertising online and offline. *American Political Science Review*, *115*(1), 130–149. https://doi.org/10.1017/S0003055420000696

Freidank, M. (2020). Reproducibility issue with Transformers (BERT) and TF2.2. *github.com/NVIDIA*. Retrieved January 20, 2022 from https://github.com/NVIDIA/framework-determinism/issues/19

Fu, T.-J., Li, L., Gan, Z., Lin, K., Wang, W. Y., Wang, L., & Liu, Z. (2021). VIOLET: End-to-end video-language Transformers with masked visual-token modeling. arXiv. https://arxiv.org/abs/2111.12681

Gatt, A. & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61, 65–170. https://doi.org/10.1613/jair.5477

Glavaš, G., Nanni, F., & Ponzetto, S. P. (2017). Cross-lingual classification of topics in political texts. In D. Hovy, S. Volkova, D. Bamman, D. Jurgens, B. O'Connor, O. Tsur, & A. S. Doğruöz (Eds.), *Proceedings of the Second Workshop on NLP and Computational Social Science* (pp. 42–46). Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-2906

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*(1), 345–420. https://jair.org/index.php/jair/article/download/11030/26198/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org

Google Colaboratory (2020). Google Colaboratory Frequently Asked Questions. *research.google.com/colaboratory*. https://research.google.com/colaboratory/faq.html

Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A., & Bojanowski, P. (2021). Self-supervised pretraining of visual features in the wild. arXiv. https://arxiv.org/abs/2103.01988

Greene, K. T., Park, B., & Colaresi, M. (2019). Machine learning human rights and wrongs: How the successes and failures of supervised learning algorithms can inform the debate about information effects. *Political Analysis*, *27*(2), 223–230. https://doi.org/10.1017/pan.2018.11

Grimmer, J. & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Han, R., Gill, M., Spirling, A., & Cho, K. (2018). Conditional word embedding and hypothesis testing via Bayes-by-backprop. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 4890–4895). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1527

Hansen, C. (2019). Activation functions explained - GELU, SELU, ELU, ReLU and more. *Machine Learning From Scratch*. https://mlfromscratch.com/activation-functions-explained/

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. arXiv. https://arxiv.org/abs/1512.03385

Hendrycks, D. & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). arXiv. https://arxiv.org/abs/1606.08415

Hinton, G., Srivastava, N., & Swerky, K. (2012). *Neural networks for machine learning. Lecture 6a: Overview of mini-batch gradient descent* [Lecture Notes]. Coursera. http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Howard, J. & Ruder, S. (2018). Universal language model fine-tuning for text classification. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (pp. 328–339). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1031

Hu, M., Peng, Y., Huang, Z., Li, D., & Lv, Y. (2019). Open-domain targeted sentiment analysis via span-based extraction and classification. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 537–546). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1051

Hugging Face (2018). PyTorch BERT model. *github.com/huggingface*. https://github.com/huggingface/transformers/blob/v4.15.0/src/transformers/models/bert/modeling_bert.py

Hugging Face (2020a). Preprocessing. *huggingface.co*. https://huggingface.co/transformers/preprocessing.html

Hugging Face (2020b). Summary of the models. *huggingface.co*. https://huggingface.co/transformers/model_summary.html

Hugging Face (2020c). Tokenizer summary. *huggingface.co*. https://huggingface.co/transformers/tokenizer_summary.html

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Iyyer, M., Enns, P., Boyd-Graber, J., & Resnik, P. (2014). Political ideology detection using recursive neural networks. In K. Toutanova & H. Wu (Eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1113–1122). Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1105

Jacovi, A. & Goldberg, Y. (2020). Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4198–4205). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.386

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., Henaff, O. J., Botvinick, M., Zisserman, A., Vinyals, O., & Carreira, J. (2022). Perceiver IO: A general architecture for structured inputs & outputs. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=fILj7WpI-g

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., & Carreira, J. (2021). Perceiver: General perception with iterative attention. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 4651–4664). PMLR. https://proceedings.mlr.press/v139/jaegle21a.html

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning with applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7

Jawahar, G., Sagot, B., & Seddah, D. (2019). What does BERT learn about the structure of language? In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3651–3657). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1356

Jigsaw/Conversation AI (2018). Toxic comment classification challenge. *kaggle.com*. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

Johnson, J. (2017). *CS231n: Convolutional neural networks for visual recog-*

nition. Derivatives, backpropagation, and vectorization [Lecture Notes]. http://cs231n.stanford.edu/handouts/derivatives.pdf

Katagiri, A. & Min, E. (2019). The credibility of public and private signals: A document-based approach. *American Political Science Review*, *113*(1), 156–172. https://doi.org/10.1017/S0003055418000643

Kentaro, W. (2020). *gdown: Download a large file from Google Drive.* [Python package]. github.com/wkentaro. https://github.com/wkentaro/gdown

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv. https://arxiv.org/abs/1609.04836

Kim, J., Griggs, E., Kim, I. S., & Oh, A. (2021). Learning bill similarity with annotated and augmented corpora of bills. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 10048–10064). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.787

King, G., Lam, P., & Roberts, M. E. (2017). Computer-assisted keyword and document set discovery from unstructured text. *American Journal of Political Science*, *61*(4), 971–988. https://doi.org/10.1111/ajps.12291

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. http://arxiv.org/abs/1412.6980

Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Y. Bengio & Y. LeCun (Eds.), *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient Transformer. arXiv. https://arxiv.org/abs/2001.04451

Kobayashi, G., Kuribayashi, T., Yokoi, S., & Inui, K. (2020). Attention is not only a weight: Analyzing Transformers with vector norms. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7057–7075). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.574

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., & Reblitz-Richardson, O. (2020). Captum: A unified and generic model interpretability library for PyTorch. arXiv. https://arxiv.org/abs/2009.07896

Kozlowski, A. C., Taddy, M., & Evans, J. A. (2019). The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review*, *84*(5), 905–949. https://doi.org/10.1177/0003122419877135

Kwon, K. H., Priniski, J. H., & Chadha, M. (2018). Disentangling user samples: A supervised machine learning approach to proxy-population mismatch in Twitter research. *Communication Methods and Measures*, *12*(2-3), 216–237. https://doi.org/10.1080/19312458.2018.1430755

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In A. Rush (Ed.), *8th International Conference on Learning Representations (ICLR 2020)*. https://openreview.net/pdf?id=H1eA7AEtvS

Laver, M., Benoit, K., & Garry, J. (2003). Extracting policy positions from political texts using words as data. *American Political Science Review*, *97*(2), 311–331. https://doi.org/10.1017/S0003055403000698

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. *Proceedings of Machine Learning Research*, *32*(2), 1188–1196. http://proceedings.mlr.press/v32/le14.html

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1–5. http://jmlr.org/papers/v18/16-365.html

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7871–7880). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Li, F.-F., Krishna, R., & Xu, D. (2020a). *CS231n: Convolutional neural networks for visual recognition. Optimization I* [Lecture Notes]. Stanford University. https://cs231n.github.io/optimization-1/

Li, F.-F., Krishna, R., & Xu, D. (2020b). *CS231n: Convolutional neural networks for visual recognition. Optimization II* [Lecture Notes]. Stanford University. https://cs231n.github.io/optimization-2/

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer,

L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv. https://arxiv.org/abs/1907.11692

Loshchilov, I. & Hutter, F. (2019). Decoupled weight decay regularization. In T. Sainath (Ed.), *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net. https://openreview.net/forum?id=Bkg6RiCqY7

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

Luo, H., Ji, L., Shi, B., Huang, H., Duan, N., Li, T., Li, J., Bharti, T., & Zhou, M. (2020). UniVL: A unified video and language pre-training model for multimodal understanding and generation. arXiv. https://arxiv.org/abs/2002.06353

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1412–1421). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1166

MacCartney, B. (2014). *Understanding natural language understanding* [Presentation]. ACM SIGAI Bay Area Chapter Inaugural Meeting. https://nlp.stanford.edu/~wcmac/papers/20140716-UNLU.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É., Seddah, D., & Sagot, B. (2020). CamemBERT: a tasty French language model. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7203–7219). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.645

Masters, D. & Luschi, C. (2018). Revisiting small batch training for deep neural networks. arXiv. https://arxiv.org/abs/1804.07612

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2018). Learned in translation: Contextualized word vectors. arXiv. https://arxiv.org/abs/1708.00107

McCormick, C. & Ryan, N. (2019). BERT fine-tuning tutorial with PyTorch. *Chris McCormick*. https://mccormickml.com/2019/07/22/BERT-fine-tuning/

McKinney, W. (2010). Data structures for statistical computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference (SciPy 2010)* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Meidinger, M. & Aßenmacher, M. (2021). A new benchmark for NLP in social sciences: Evaluating the usefulness of pre-trained language models for classifying open-ended survey responses. In A. P. Rocha, L. Steels, & H. J. van den Herik (Eds.), *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021* (pp. 866–873). SciTePress. https://doi.org/10.5220/0010255108660873

Michael Waskom and Team (2020). *Seaborn*. [Python package]. Zenodo. https://zenodo.org/record/4379347

Mikhaylov, S., Laver, M., & Benoit, K. R. (2012). Coder reliability and misclassification in the human coding of party manifestos. *Political Analysis*, *20*(1), 78–91. https://doi.org/10.1093/pan/mpr047

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv. https://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3111–3119). Curran Associates Inc. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Miller, B., Linder, F., & Mebane, W. R. (2020). Active learning approaches for labeling text: Review and assessment of the performance of active learning approaches. *Political Analysis*, *28*(4), 532–551. https://doi.org/10.1017/pan.2020.4

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 1–38. https://doi.org/10.1016/j.artint.2018.07.007

Mitchell, M., Aguilar, J., Wilson, T., & Van Durme, B. (2013). Open domain targeted sentiment. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1643–1654). Association for Computational Linguistics. https://aclanthology.org/D13-1171

Mitts, T. (2019). From isolation to radicalization: Anti-Muslim hostility and support for ISIS in the West. *American Political Science Review*, *113*(1), 173–194. https://doi.org/10.1017/S0003055418000618

Mohammad, S. M., Sobhani, P., & Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Transactions on Internet Technology*, *17*(3), 1–22. https://doi.org/10.1145/3003433

Molnar, C. (2022). *Interpretable machine learning*. https://christophm.github.io/interpretable-ml-book/

Muchlinski, D., Yang, X., Birch, S., Macdonald, C., & Ounis, I. (2021). We

need to go deeper: Measuring electoral violence using convolutional neural networks and social media. *Political Science Research and Methods*, *9*(1), 122–139. https://doi.org/10.1017/psrm.2020.32

Münchener Digitalisierungszentrum der Bayerischen Staatsbibliothek (dbmdz) (2021). Model card for bert-base-german-uncased from dbmdz. *huggingface.co*. https://huggingface.co/dbmdz/bert-base-german-uncased

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve Restricted Boltzmann machines. In J. Fürnkranz & T. Joachims (Eds.), *Proceedings of the 27th International Conference on International Conference on Machine Learning* (pp. 807–814). Omnipress. https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

Nelson, L. K., Burk, D., Knudsen, M., & McCall, L. (2021). The future of coding: A comparison of hand-coding and three types of computer-assisted text analysis methods. *Sociological Methods & Research*, *50*(1), 202–237. https://doi.org/10.1177/0049124118769114

Nguyen, D. Q. & Tuan Nguyen, A. (2020). PhoBERT: Pre-trained language models for Vietnamese. In T. Cohn, Y. He, & Y. Liu (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 1037–1042). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.92

Nozza, D., Bianchi, F., & Hovy, D. (2020). What the [MASK]? Making sense of language-specific BERT models. arXiv. https://arxiv.org/abs/2003.02912

Oliphant, T. E. (2006). *A Guide to NumPy*. Trelgol Publishing.

Osnabrügge, M., Ash, E., & Morelli, M. (2021). Cross-domain topic classification for political texts. *Political Analysis*, (pp. 1–22). https://doi.org/10.1017/pan.2021.37

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 79–86). Association for Computational Linguistics. https://doi.org/10.3115/1118693.1118704

Park, B., Greene, K., & Colaresi, M. (2020). Human rights are (increasingly) plural: Learning the changing taxonomy of human rights from large-scale text reveals information effects. *American Political Science Review*, *114*(3), 888–910. https://doi.org/10.1017/S0003055420000258

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach,

H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Perry, P. O. & Benoit, K. (2017). Scaling text with the Class Affinity Model. arXiv. https://arxiv.org/abs/1710.08963

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018a). Deep contextualized word representations. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 2227–2237). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1202

Peters, M., Neumann, M., Zettlemoyer, L., & Yih, W.-t. (2018b). Dissecting contextual word embeddings: Architecture and representation. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1499–1509). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1179

Peters, M. E., Ruder, S., & Smith, N. A. (2019). To tune or not to tune? Adapting pretrained representations to diverse tasks. In I. Augenstein, S. Gella, S. Ruder, K. Kann, B. Can, J. Welbl, A. Conneau, X. Ren, & M. Rei (Eds.), *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)* (pp. 7–14). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4302

Pilehvar, M. T. & Camacho-Collados, J. (2020). *Embeddings in natural language processing: Theory and advances in vector representations of meaning.* Morgan & Claypool Publishers. https://doi.org/10.2200/S01057ED1V01Y202009HLT047

Pilny, A., McAninch, K., Slone, A., & Moore, K. (2019). Using supervised machine learning in automated content analysis: An example using relational uncertainty. *Communication Methods and Measures*, *13*(4), 287–304. https://doi.org/10.1080/19312458.2019.1650166

Porter, E. & Velez, Y. R. (2021). Placebo selection in survey experiments: An agnostic approach. *Political Analysis*, (pp. 1–14). https://doi.org/10.1017/pan.2021.16

Princeton University (2010). *About WordNet*. https://wordnet.princeton.edu/

R Core Team (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 8748–8763). PMLR. https://proceedings.mlr.press/v139/radford21a.html

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. Manuscript. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. Manuscript. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. http://jmlr.org/papers/v21/20-074.html

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-shot text-to-image generation. *Proceedings of Machine Learning Research*, 139, 8821–8831. https://proceedings.mlr.press/v139/ramesh21a.html

Ramey, A. J., Klingler, J. D., & Hollibaugh, G. E. (2019). Measuring elite personality using speech. *Political Science Research and Methods*, *7*(1), 163–184. https://doi.org/10.1017/psrm.2016.12

Raschka, S. (2020). *watermark*. [Python package]. github.com/rasbt. https://github.com/rasbt/watermark

Rehbein, I., Ponzetto, S. P., Adendorf, A., Bahnsen, O., Stoetzer, L., & Stuckenschmidt, H. (2021a). Come hither or go away? Recognising pre-electoral coalition signals in the news. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7798–7810). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.615

Rehbein, I., Ruppenhofer, J., & Bernauer, J. (2021b). Who is we? Disambiguating the referents of first person plural pronouns in parliamentary debates. In K. Evang, L.

Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 147–158). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.13

Reimers, N. & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Rheault, L., Beelen, K., Cochrane, C., & Hirst, G. (2016). Measuring emotion in parliamentary debates with automated textual analysis. *PLoS ONE*, *11*(12), e0168843. https://doi.org/10.1371/journal.pone.0168843

Rheault, L. & Cochrane, C. (2020). Word embeddings for the analysis of ideological placement in parliamentary corpora. *Political Analysis*, *28*(1), 112–133. https://doi.org/10.1017/pan.2019.26

Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4902–4912). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.442

Riedl, M. (2020). AI democratization in the era of GPT-3. *The Gradient*. https://thegradient.pub/ai-democratization-in-the-era-of-gpt-3/

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., & Fergus, R. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, *118*(15), e2016239118. https://doi.org/10.1073/pnas.2016239118

Roberts, M. E., Stewart, B. M., & Airoldi, E. M. (2016). A model of text for experimentation in the social sciences. *Journal of the American Statistical Association*, *111*(515), 988–1003. https://doi.org/10.1080/01621459.2016.1141684

Rodman, E. (2020). A timely intervention: Tracking the changing meanings of political concepts with word vectors. *Political Analysis*, *28*(1), 87–111. https://doi.org/10.1017/pan.2019.23

Rona-Tas, A., Cornuéjols, A., Blanchemanche, S., Duroy, A., & Martin, C. (2019). Enlisting supervised machine learning in mapping scientific uncertainty expressed in food risk analysis. *Sociological Methods & Research*, *48*(3), 608–641. https://doi.org/10.1177/0049124117729701

Rothe, S. & Schütze, H. (2015). AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual*

Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 1793–1803). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-1173

Ruder, S. (2018). NLP's ImageNet moment has arrived. *Sebastian Ruder*. https://ruder.io/nlp-imagenet/

Ruder, S. (2019a). *Neural transfer learning for natural language processing*. PhD thesis, National University of Ireland, Galway. https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf

Ruder, S. (2019b). Unsupervised cross-lingual representation learning. *Sebastian Ruder*. https://ruder.io/unsupervised-cross-lingual-learning/index.html

Ruder, S. (2020). *NLP-Progress*. https://nlpprogress.com/ (accessed August 04, 2020)

Ruder, S. (2021). Recent advances in language model fine-tuning. *Sebastian Ruder*. https://ruder.io/recent-advances-lm-fine-tuning/

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, S., & Sedlmair, M. (2018). More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures*, *12*(2–3), 140–157. https://doi.org/10.1080/19312458.2018.1455817

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., & Gurevych, I. (2021). How good is your tokenizer? on the monolingual performance of multilingual language models. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 3118–3135). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.243

Schuster, M. & Nakajima, K. (2012). Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5149–5152). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ICASSP.2012.6289079

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123. https://aclanthology.org/J98-1004

scikit-learn Developers (2020a). 1.4. Support Vector Machines. *scikit-learn*. https://scikit-learn.org/stable/modules/svm.html

scikit-learn Developers (2020b). Metrics and scoring: quantifying the quality of predictions. *scikit-learn*. https://scikit-learn.org/stable/modules/model_evaluation.html

scikit-learn Developers (2020c). RBF SVM Parameters. *scikit-learn*. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Sebők, M. & Kacsuk, Z. (2021). The multiclass classification of newspaper articles with machine learning: The hybrid binary snowball approach. *Political Analysis*, *29*(2), 236–249. https://doi.org/10.1017/pan.2020.27

Selivanov, D., Bickel, M., & Wang, Q. (2020). *text2vec: Modern text mining framework for R*. [R package]. CRAN. https://CRAN.R-project.org/package=text2vec

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In K. Erk & N. A. Smith (Eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1715–1725). Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1162

Slapin, J. B. & Kirkland, J. H. (2020). The sound of rebellion: Voting dissent and legislative speech in the UK House of Commons. *Legislative Studies Quarterly*, *45*(2), 153–176. https://doi.org/10.1111/lsq.12251

Slapin, J. B. & Proksch, S.-O. (2008). A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, *52*(3), 705–722. https://doi.org/10.1111/j.1540-5907.2008.00338.x

Smith, N. A. (2011). *Linguistic structure prediction*. Morgan & Claypool Publishers.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1631–1642). Association for Computational Linguistics. https://aclanthology.org/D13-1170

Song, H., Tolochko, P., Eberl, J.-M., Eisele, O., Greussing, E., Heidenreich, T., Lind, F., Galyga, S., & Boomgaarden, H. G. (2020). In validations we trust? The impact of imperfect human annotations as a gold standard on the quality of validation of automated content analysis. *Political Communication*, *37*(4), 550–572. https://doi.org/10.1080/10584609.2020.1723752

Spirling, A. & Rodriguez, P. L. (2020). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. Manuscript. http://arthurspirling.org/documents/embed.pdf

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019a). VideoBERT: A joint

model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/ICCV.2019.00756

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019b). How to fine-tune BERT for text classification? arXiv. https://arxiv.org/abs/1905.05583v3

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3104–3112). MIT Press. https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2021). Long Range Arena: A benchmark for efficient Transformers. In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*. https://openreview.net/forum?id=qVyeW-grC2k

Tenney, I., Das, D., & Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4593–4601). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1452

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., & Pavlick, E. (2019b). What do you learn from context? Probing for sentence structure in contextualized word representations. In T. Sainath (Ed.), *7th International Conference on Learning Representations, ICLR 2019*. https://openreview.net/pdf?id=SJzSgnRcKX

Theocharis, Y., Barberá, P., Fazekas, Z., Popa, S. A., & Parnet, O. (2016). A bad workman blames his tweets: The consequences of citizens' uncivil Twitter use when interacting with party candidates. *Journal of Communication*, *66*(6), 1007–1031. https://doi.org/10.1111/jcom.12259

Torch Contributors (2021). Reproducibility. *pytorch.org*. https://pytorch.org/docs/stable/notes/randomness.html

Turney, P. D. & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37, 141–188. https://doi.org/10.1613/jair.2934

Ushey, K., Allaire, J. J., Wickham, H., & Ritchie, G. (2020). *rstudioapi: Safely access the RStudio API*. (Version 0.11) [R package]. CRAN. https://CRAN.R-project.org/package=rstudioapi

van Rossum, G. & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Vapnik, V. (1991). Principles of risk minimization for learning theory. In J. Moody, S. Hanson, & R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems*

4 (pp. 831–838). Morgan-Kaufmann. https://proceedings.neurips.cc/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 3266–3280). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-attention with linear complexity. arXiv. https://arxiv.org/abs/2006.04768

Wankmüller, S. & Heumann, C. (2021). How to estimate continuous sentiments from texts using binary training data. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 182–192). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.16

Watanabe, K. (2021). Latent semantic scaling: A semisupervised text analysis technique for new domains and languages. *Communication Methods and Measures*, *15*(2), 81–102. https://doi.org/10.1080/19312458.2020.1832976

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2022). Finetuned language models are zero-shot learners. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=gEZrGCozdqR

Welbers, K., van Atteveldt, W., & Benoit, K. (2017). Text analysis in R. *Communication Methods and Measures*, *11*(4), 245–265. https://doi.org/10.1080/19312458.2017.1387238

Wickham, H. (2019). *stringr: Simple, consistent wrappers for common string operations*. (Version 1.4.0) [R package]. CRAN. https://CRAN.R-project.org/package=stringr

Williams, A., Nangia, N., & Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1112–1122). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1101

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). Hugging Face's Transformers: State-of-the-art natural language processing. arXiv. https://arxiv.org/abs/1910.03771v5

Wu, P. Y. & Mebane, W. R. (2021). MARMOT: A deep learning framework for constructing multimodal representations for vision-and-language tasks. arXiv. https://arxiv.org/abs/2109.11526v1

Wu, S. & Dredze, M. (2019). Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 833–844). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1077

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv. https://arxiv.org/abs/1609.08144

xgboost Developers (2020). Python API reference. https://xgboost.readthedocs.io/en/latest/python/python_api.html

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, & Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 483–498). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.41

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, (pp. 5753–5763). Curran Associates, Inc. http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

Yin, P., Neubig, G., Yih, W.-t., & Riedel, S. (2020). TaBERT: Pretraining for joint understanding of textual and tabular data. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8413–8426). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.745

Yin, W., Hay, J., & Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3914–3923). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1404

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27*, (pp. 3320–3328). Curran Associates, Inc. http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big Bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 17283–17297). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf

Zarrella, G. & Marsh, A. (2016). MITRE at SemEval-2016 task 6: Transfer learning for stance detection. In S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, & T. Zesch (Eds.), *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 458–463). Association for Computational Linguistics. https://doi.org/10.18653/v1/S16-1074

Zhang, H. & Pan, J. (2019). CASM: A deep-learning approach for identifying collective action events with text and image data from social media. *Sociological Methodology*, *49*(1), 1–57. https://doi.org/10.1177/0081175019860244

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)* (pp. 19–27). IEEE. https://doi.org/10.1109/ICCV.2015.11

Chapter 3

A Comparison of Approaches for Imbalanced Classification Problems in the Context of Retrieving Relevant Documents for an Analysis

Article.

The following article version is the version at the time this dissertation has been handed in. This article manuscript has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this article is published in the *Journal of Computational Social Science* and is available online at https://doi.org/10.1007/s42001-022-00191-7. The full citation information for the published article is as follows:

Wankmüller, S. (2022). A comparison of approaches for imbalanced classification problems in the context of retrieving relevant documents for an analysis. *Journal of Computational Social Science*, (p. 1–73). https://doi.org/10.1007/s42001-022-00191-7

The article manuscript also is available at https://arxiv.org/abs/2205.01600

Source Code. https://doi.org/10.6084/m9.figshare.19699840

A Comparison of Approaches for Imbalanced Classification Problems in the Context of Retrieving Relevant Documents for an Analysis

Abstract. One of the first steps in many text-based social science studies is to retrieve documents that are relevant for the analysis from large corpora of otherwise irrelevant documents. The conventional approach in social science to address this retrieval task is to apply a set of keywords and to consider those documents to be relevant that contain at least one of the keywords. But the application of incomplete keyword lists risks drawing biased inferences. More complex and costly methods, such as query expansion techniques, topic model-based classification rules, and active as well as passive supervised learning, could have the potential to more accurately separate relevant from irrelevant documents and thereby reduce the potential size of bias. Whether applying these more expensive approaches increases retrieval performance compared to keyword lists at all, and if so, by how much, is unclear as a comparison of these approaches is lacking. This study closes this gap by comparing these methods across three retrieval tasks associated with a data set of German tweets (Linder, 2017), the Social Bias Inference Corpus (SBIC) (Sap et al., 2020), and the Reuters-21578 corpus (Lewis, 1997). Results show that query expansion techniques and topic model-based classification rules in most studied settings tend to decrease rather than increase retrieval performance. Active supervised learning, however, if applied on a not too small set of labeled training instances (e.g. 1,000 documents), reaches a substantially higher retrieval performance than keyword lists.

Keywords. Imbalanced classification, Boolean query, keyword lists, query expansion, topic models, active learning

Acknowledgement. I am very grateful to Paul W. Thurner, Christian Heumann, Thomas Däubler, Peer Kröger, and the participants of the colloquium at the chair of Paul W. Thurner for helpful comments and suggestions on this work.

3.1 Introduction

When conducting a study on the basis of text data, at the very start of an analysis researchers are often confronted with a difficulty: Online platforms and other sources from which text data are generated usually cover multiple topics and hence tend to contain textual references toward a huge number of various entities. Social scientists, however, are typically interested in text elements referring to a single entity, e.g. a specific person, organization, object, event, or issue.

Imagine, for example, that a study seeks to examine how rape incidents are framed in newspaper articles (Baum et al., 2018), or that a study seeks to detect electoral violence

based on social media data (Muchlinski et al., 2021), or that a study seeks to measure attitudes expressed towards further European integration in speeches of political elites (Rauh et al., 2020). In all these studies, one of the first steps is to extract documents that refer to the entity of interest from a large, multi-thematic corpus of documents.[1] This is, researchers have to separate the relevant documents that refer to the entity of interest from the documents that focus on entities irrelevant for the analysis at hand. Newspaper articles that report about rape incidents have to be parted from those articles that do not. Tweets relating to electoral violence have to be extracted from the stream of all other tweets. And speech elements about the European integration have to be separated from elements in which the speaker talks about other entities.

Given a corpus comprising many diverse topics, it is likely that only a small proportion of documents relate to the entity of interest. Hence, the proportion of relevant documents is usually substantively smaller than the proportion of irrelevant documents in the data, and the task of separating relevant from irrelevant documents turns into an imbalanced classification problem (Manning et al., 2008, p. 155). How researchers address this imbalanced classification problem is highly important as the selection of documents affects the inferences drawn. More precisely: If there is a systematic bias in the selection of documents such that the value on a variable of interest is related to the question of whether a document is selected for analysis or not, the inferences that are made on the basis of the documents that have been selected for analysis are likely to be biased. Selection biases can be induced when the corpus is collected in the first place.[2] And selection biases can be induced when documents that refer to relevant entities are selected for analysis from the already collected corpus. This work focuses on the second step. The more accurately a method can separate relevant from irrelevant documents, the smaller the potential size of the bias resulting from this second selection step.

Despite the relevance of this problem, the question of how best to retrieve documents from large, heterogenous corpora so far has received little attention in social science research. In many applications researchers have relied on applying human-created sets of keywords and regard those documents as relevant that comprise at least one of the keywords (see e.g. Burnap et al., 2016; Jungherr et al., 2016; Beauchamp, 2017; Baum et al., 2018; Stier et al., 2018; Fogel-Dror et al., 2019; Rauh et al., 2020; Watanabe, 2021; Muchlinski et al., 2021). Yet research indicates that humans are not good at generating comprehensive keyword lists and are highly unreliable at the task (King et al., 2017, p. 973-975). This is, the keyword list generated by one human is likely to contain only a small amount of the universe of terms one could use to refer to a given entity of interest (King et al., 2017, p. 973-975). Moreover, the list of keywords that one human comes up with is likely to show little overlap with the keyword list generated by another human (King et al., 2017, p. 973-975). Joining forces

[1] A corpus is a set of documents. A document is the unit of observation. A document can be a very short to a very long text (e.g. a sentence, a speech, a newspaper article). Here, the term corpus refers to the set of documents a researcher has collected and from which he or she then seeks to retrieve the relevant documents.

[2] I thank Christian Heumann for pointing this out to me.

by combining keyword lists that researchers have created independently may alleviate the problem somewhat. But still, the conventional approach of using keywords to identify relevant documents is likely to be unreliable and thus is likely to lead to very different (and potentially biased) conclusions depending on which set of keywords the researchers have used (King et al., 2017, p. 974-976).

Other approaches for identifying relevant documents—such as passive and active supervised learning, query expansion techniques, or the construction of topic model-based classification rules—are less frequently employed in social science applications. These approaches also require human input, but they detect patterns or keywords the researchers do not have to know beforehand. Except for query expansion, these methods require the researchers to *recognize* documents or terms related to the entity of interest rather than requiring the researchers to *recall* such information a priori (King et al., 2017, p. 972). This does not preclude these techniques from generating selection biases. A supervised learning algorithm, for example, may systematically misclassify some documents as not being relevant based on features whose occurrence could be correlated with a main variable of the analysis. Yet as these approaches have the potential to extract patterns beyond what a team of researchers may come up with, these methods have the potential to more precisely separate relevant from non-relevant documents. And the higher the retrieval performance of a method, the smaller the maximum size of the selection bias induced by retrieving relevant documents.

These other techniques, however, also have a disadvantage: They are much more resource intensive to implement. Supervised learning algorithms require labeled training documents, query expansion techniques depend on a data source to operate on, and topic model-based classification rules require the estimation of a topic model. As the identification of relevant documents from a large heterogeneous corpus is likely to only constitute an early small step of an elaborate text analysis, considerations regarding the costs and benefits of a retrieval method have to be taken into account.

Hence, an ideal procedure reliably achieves a high retrieval performance such that it reduces the risk of incurring large selection biases and, simultaneously, is cost-effective enough to be conducted as a single step of an extensive study. In practice, the performance and the cost-effectiveness of a procedure are likely to depend on the characteristics of an application (such as the type of the entity of interest, or the heterogeneity vs. homogeneity of the documents in the corpus). If the entity of interest is a person or organization and there is only a small set of expressions that are usually used to refer to this entity, then a list of keywords may lead to a similar performance than the resource-intensive application of a supervised learning algorithm. If, on the other hand, the entity of interest is not easily denominated, then an acceptable retrieval performance may only be achieved by training a supervised learning algorithm.

Up to now, however, a systematic comparison of the performances of these different retrieval methods across social science applications is lacking. Thus, it is unclear what, if anything, could be gained in terms of retrieval performance by applying a more elaborate procedure.

This study seeks to answer this question by comparing the retrieval performance of a small set of predictive keywords to (1) query expansion techniques extending this initial set, (2) topic model-based classification rules, and (3) active as well as passive supervised learning. The procedures are compared on the basis of three retrieval tasks: (1) the identification of tweets referring to refugees, refugee policies, and the refugee crisis from a data set of 24,420 German tweets (Linder, 2017), (2) the retrieval of posts that are offensive toward mentally or physically disabled people from the Social Bias Inference Corpus (SBIC) (Sap et al., 2020) that covers 44,671 potentially toxic and offensive posts from various social media platforms, and (3) the extraction of newspaper articles referring to crude oil from the Reuters-21578 corpus (Lewis, 1997) that comprises economically focused newspaper articles of which 10,377 are assigned to a topic.

The results show that—with the model settings studied here—query expansion techniques as well as topic-model-based classification rules tend to decrease rather than increase retrieval performance compared to sets of predictive keywords. They only yield minimal improvements or acceptable results in specific settings. By contrast, active supervised learning—if implemented with a not too small number of labeled training documents—achieves relatively high retrieval performances across contexts. Moreover, in each application, active learning substantively improves upon the mediocre to acceptable results reached by the best performing lists of predictive keywords. The observed differences of the mean F_1-Scores achieved by active learning with 1,000 labeled training documents to the *maximum* F_1-Scores of keyword lists range between 0.218 and 0.295. Although active learning is designed to reduce the number of training documents that have to be annotated by human coders, it nevertheless is particularly resource-intensive. Yet the achieved performance enhancements are so considerable (and the consequences of selection biases potentially so severe) that researchers should consider spending more of their available resources on the step of separating relevant from irrelevant documents.

In the following Section 3.2, basic concepts relevant for discussing imbalanced classification problems in retrieval contexts are introduced. Afterward, the benefits and disadvantages of the usage of keyword lists, query expansion techniques, topic model-based classification rules, and passive as well as active supervised learning techniques in the context of identifying documents relevant for further analyses are discussed (Section 3.3). Then the procedures are applied to the data sets and their retrieval performances are inspected (Section 3.4). The final discussion in Section 3.5 summarizes what has been learned and points toward aspects that merit further study.

Before continuing, note that the vocabulary used in this study often makes use of the term *retrieval*. As this study focuses on contexts in which the task is to retrieve relevant documents from corpora of otherwise irrelevant documents, the usage of the term *retrieval* seems adequate. The task examined in this study, however, is different from the task that is typically examined in *document retrieval*. Document retrieval is a subfield of information retrieval in which the usual task is to rank documents according to their relevance for an explicitly stated user query (Manning et al., 2008, p. 14, 16). In this study, in contrast, the

	truly positive	**truly negative**	
predicted positive	True Positives (TP)	False Positives (FP)	$TP + FP$
predicted negative	False Negatives (FN)	True Negatives (TN)	$FN + TN$
	$TP + FN$	$FP + TN$	N

Table 3.1: **The Confusion Matrix.**

aim is to classify—rather than rank—documents as being relevant vs. not relevant. Moreover, not all of the approaches evaluated here require the query, that states the information need, to be expressed explicitly in the form of keywords or phrases.

3.2 Imbalanced Classification, Precision, Recall

Imbalanced classification problems are common in information retrieval tasks (Manning et al., 2008, p. 155). They are characterized by an imbalance in the proportions made up by one vs. the other category. When retrieving relevant documents from large corpora typically only a small fraction of documents fall into the positive relevant category whereas an overwhelming majority of documents are part of the negative irrelevant category (Manning et al., 2008, p. 155).

When evaluating the performance of a method in a situation of imbalance, the accuracy measure that gives the share of correctly classified documents is not adequate (Manning et al., 2008, p. 155). The reason is that a method that would assign all documents to the negative irrelevant category would get a very high accuracy value (Manning et al., 2008, p. 155) Thus, evaluation metrics that allow for a refined view, such as precision and recall, should be employed (Manning et al., 2008, p. 155). Precision and recall are defined as

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

TP, FP and FN are defined in Table 3.1. Precision and recall are in the range $[0, 1]$. However, if none of the documents is predicted to be positive, then $TP + FP = 0$, and precision is undefined. If there are no truly positive documents in the corpus, then $TP + FN = 0$, and recall is undefined. The higher precision and recall, the better.

Precision exclusively takes into account all documents that have been assigned to the positive category by the classification method and informs about the share of truly positive documents among all documents that are predicted to fall into the positive category. Recall, on the other hand, exclusively focuses on the truly relevant documents and informs about

the share of documents that have been identified as relevant among all truly relevant documents.

There is a trade-off between precision and recall (Manning et al., 2008, p. 156). A keyword list comprising many terms or a classification algorithm that is lenient in considering documents to be relevant will likely identify many of the truly relevant documents (high recall). Yet as the hurdle for being considered relevant is low, they will also classify many truly irrelevant documents into the relevant category (low precision). A keyword list consisting of a few specific terms or a classification algorithm with a high threshold for assigning documents to the relevant class will likely miss many relevant instances (low recall), but among those considered relevant many are likely to indeed be relevant (high precision).

In this study's context of identifying relevant documents to be used for further analyses, recall and precision should be as high as possible, but recall is the slightly more important metric: Recall operates on the set of all truly relevant documents and focuses on the inclusion vs. exclusion of relevant documents into the analysis—the analytic step at which selection biases may arise. If there is a correlation between the documents identified as relevant vs. not relevant and the value of the variable of interest, a selection bias is generated. This is, if truly relevant documents are systematically misclassified in the sense that the higher (or lower) the value on the variable of interest, the higher (or lower) the probability of being assigned to the negative irrelevant category, inferences that are made based on the set of instances classified into the positive category are biased. High recall values do not guarantee that there are no systematic misclassifications. But the higher recall, the smaller the maximum size of the selection bias that arises from systematic misclassifications of truly relevant documents.

Because of its exclusive focus on true and false positives, precision provides no information on the potential of selection bias due to the missing out of truly relevant documents. Nevertheless, precision should also be high. The lower precision, the fewer documents among those considered to be relevant by the classification method are indeed relevant. A considerable share of false positives among the set of documents classified to be relevant also has the potential to severely bias the inferences drawn or can impede the researcher from conducting any analysis at all because the retrieved documents are not those documents he or she seeks to analyze. Yet whereas low precision can be handled by a researcher in subsequent steps, low recall implies that a substantial proportion of truly relevant documents are never to be considered for analysis. Hence, falsely classifying a truly relevant document as irrelevant can be considered to be more severe than falsely predicting an irrelevant document to be relevant.

The trade-off between precision and recall is incorporated in the F_ω-measure, which is the weighted harmonic mean of precision and recall (Manning et al., 2008, p. 156):

$$F_\omega = \frac{(\omega^2 + 1) \cdot Precision \cdot Recall}{\omega^2 \cdot Precision + Recall} \tag{3.3}$$

The F_ω-measure also is in the range $[0, 1]$. ω is a real-valued factor balancing the importance of precision vs. recall (Manning et al., 2008, p. 156). For $\omega > 1$ recall, is considered more important than precision, and if $\omega < 1$, precision is weighted more than recall (Manning et al., 2008, p. 156). A very common choice for ω is 1 (Manning et al., 2008, p. 156). In this case, the F_1-measure (or synonymously: F_1-Score) is the harmonic mean between precision and recall (Manning et al., 2008, p. 156).

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3.4}$$

The F_1-Score is a widely used measure to evaluate the performance of classification tasks. Although recall here is considered the slightly more important measure, the F_1-Score—because it is the measure nearly always reported—will be employed to assess the performances of the retrieval approaches evaluated in the following. Nevertheless, recall and precision values will be reported in the Appendix.

3.3 Retrieval Approaches

3.3.1 Keyword Lists

In social science, a very widely used approach to identify documents on relevant entities is to set up a set of keywords and to consider those documents as relevant that contain at least one of the keywords (see for example the studies listed in Table 3.2). This procedure, in fact, is a keyword-based Boolean query in which the keywords are connected with the OR operator (Manning et al., 2008, p. 4). Slightly more advanced are Boolean queries in which in addition to the OR operator also the AND operator is used. Using the AND operator is important in situations in which expressions denoting the entity of interest are composed of more than a single term (e.g. *'United States'*).

The ways in which the authors come up with a set of keywords range from simply using the most obvious terms (e.g. Baum et al., 2018), to collecting a set of typical denominations for the entity of interest (e.g. Burnap et al., 2016; Jungherr et al., 2016; Beauchamp, 2017), to carefully thinking about, testing, and revising sets of keywords (e.g. Stier et al., 2018; Abdul Reda et al., 2021; Gessler & Hunger, 2021), to collecting keywords empirically based on word-usage in texts known to be about the entity (e.g. Zhang & Pan, 2019). Though these approaches vary in their complexity and costs, they are all still very cheap and relatively fast procedures. Another advantage of the usage of keyword lists for the extraction of relevant documents is that a researcher has full control over the terms that are included—and not included—as keywords.

Nevertheless, research suggests that the human construction of keyword lists is not reliable (King et al., 2017, p. 973-975). If a researcher generates a keyword list, then another

Study	Number of keywords	How are the keywords selected?	Operators in Boolean query
Puglisi & Snyder (2011)	11+	likely by the authors	OR, AND
King et al. (2013)	unspecified	likely by the authors	unclear
Burnap et al. (2016)	33	likely by the authors	OR
Jungherr et al. (2016)	86	by the authors	OR
Beauchamp (2017)	36	likely by the authors	OR
van Atteveldt et al. (2017)	1	by the authors	-
Baum et al. (2018)	2	likely by the authors	OR
Stier et al. (2018)	218	by the authors	OR
Fogel-Dror et al. (2019)	27-170	by the authors	OR
Katagiri & Min (2019)	unspecified	from COPDAB data bank	OR, AND
Zhang & Pan (2019)	50	empirically; frequency-based	OR
Rauh et al. (2020)	14	likely by the authors	OR
Uyheng & Carley (2020)	1	likely by the authors	-
Abdul Reda et al. (2021)	57	by the authors	OR, AND
Gessler & Hunger (2021)	94	by the authors; re-usage of lists created by other authors	OR
Muchlinski et al. (2021)	30-38	by the authors	OR
Watanabe (2021)	2-4	by the authors	OR

Table 3.2: **Social Science Studies Applying Keyword Lists.** This table exemplary lists social science studies that employ lists of keywords to retrieve documents or text elements that are relevant for (a part of) their analysis. A similar but older list of studies can be found in Linder (2017, p. 5). Note that the column *'Number of keywords'* gives the number of keywords the authors in the listed studies use to extract documents relating to one entity of interest. If the authors are interested in several entities, then typically several keyword lists are applied which is why here for some articles a range rather than a single number is given. Note also that Katagiri & Min (2019, p. 161) state that the keywords they use come from the Conflict and Peace Data Bank (COPDAB) (Azar, 2009). They do not specify how they extract keywords from this data bank.

researcher or the same researcher at another point in time is likely to construct a very different set of keywords. This is problematic: Depending on which human-generated set of search terms is used to identify relevant documents, inferences drawn can vary greatly (King et al., 2017, p. 974-976).

Moreover, this conventional procedure of human keyword list generation might lead to biased inferences if the terms that are used to denote an entity correlate with the values of the variable of interest. To illustrate: Imagine that a researcher is interested in attitudes toward Joe Biden as expressed in comments on an online platform during a given time period. The researcher analyzes the sentiments of all comments that contain the search term *'Biden'*. The obtained results can be biased if the attitudes expressed in comments that refer to Joe Biden as *'Biden'* or *'Joe Biden'* differ from the attitudes in comments that refer to him as *'Sleepy Joe'*. For keyword-based approaches to avoid such types of selection bias, a researcher thus has to set up a set of keywords that fully captures the universe of terms and expressions that are used to refer to the entity of interest in the given corpus.[3] But humans tend to perform very poorly when it comes to constructing an extensive set of search terms (King et al., 2017, p. 973-975).

There are several likely reasons for the problems human researchers encounter when trying to set up an extensive list of keywords. First, language is highly varied (Durrell, 2008). There are numerous ways to refer to the same entity—and entities also can be referred to indirectly without the usage of proper names or well-defined denominations (Baden et al., 2020, p. 167). Especially if the entity of interest is abstract and/or not easily denominated, the universe of terms and expressions referring to the entity is likely to be large and not easily captured (Baden et al., 2020, p. 167). Such entities are abundant in social science. Typical entities of interest, for example, are policies (e.g. the policies implemented to address the COVID-19 pandemic), concepts (e.g. European integration or homophobia), and occurrences (e.g. the 2015 European refugee crisis or the 2021 United States Capitol riot).

A second likely reason for the human inability to come up with a comprehensive keyword list is the effect of inhibitory processes (Bäuml, 2007; King et al., 2017, p. 974). After a set of concepts has been retrieved from memory, inhibitory processes suppress the representation of related, non-retrieved concepts in memory and thereby reduce the probability of those concepts being recovered (Bäuml, 2007). One approach that has the potential to alleviate this second aspect is the utilization of query expansion methods, which are discussed next.

[3]Such a comprehensive list of keywords implies low precision and thus would come with another problem: a large share of false positives. Nevertheless, a comprehensive list would imply perfect recall and thus would preclude selection bias due to false negatives.

3.3.2 Query Expansion

By being able to move beyond keywords that researchers are able to recall a priori, query expansion methods can be employed to create a more comprehensive set of search terms. Query expansion techniques expand the original query (i.e. the original set of keywords) by appending related terms (Azad & Deepak, 2019, p. 1699-1700). Here, the focus is on similarity-based automatic query expansion methods, that add new terms automatically—i.e. without interactive relevance feedback from the user—and make use of the similarity between the set of query terms and potential expansion terms (Azad & Deepak, 2019, p. 1700, 1706). The underlying hypothesis used here is the association hypothesis formulated by van Rijsbergen stating that "[i]f one index term is good at discriminating relevant from non-relevant documents, then any closely associated index term is also likely to be good at this" (van Rijsbergen, 2000, p. 11). The specific methods differ regarding

- the data source to extract candidate terms for the expansion,
- how candidate terms from this data source are ranked (such that the ranks reflect the relatedness to the original query), and
- how (many) additional terms are selected and integrated into the original query

(Azad & Deepak, 2019, p. 1701). Data sources from which expansion terms are identified can be the corpus from which relevant documents are to be retrieved, the documents retrieved by the initial query, human-created thesauri such as WordNet, knowledge bases such as Wikipedia, external corpora (such as a global collection of web texts), or a combination of these (Azad & Deepak, 2019, p. 1701-1704). If thesauri such as WordNet are employed as a data source, terms the thesaurus encodes to be related to the query terms can be considered candidate terms for expansion (Azad & Deepak, 2019, p. 1702). Path lengths between the synsets (word senses) in a thesaurus can be used to compute a similarity score between a query term and potential expansion terms (Azad & Deepak, 2019, p. 1705). In Wikipedia, the network of hyperlinks between articles can be used to extract articles about concepts related to the query terms (ALMasri et al., 2013). A similarity score, for example, can be computed based on shared ingoing and outgoing hyperlinks between articles (ALMasri et al., 2013, p. 6). If the data source for query expansion is the local corpus from which documents are to be retrieved or if the data source is an external global corpus, then the similarity between terms can be assessed via similarity measures that are computed based on the terms' vector representations (Azad & Deepak, 2019, p. 1706). A frequently used measure is cosine similarity:

$$sim_{cos}(a_1, a_2) = cos(\theta) = \frac{\boldsymbol{z}_{[a_1]} \cdot \boldsymbol{z}_{[a_2]}}{||\boldsymbol{z}_{[a_1]}||\ ||\boldsymbol{z}_{[a_2]}||} \tag{3.5}$$

where $\boldsymbol{z}_{[a_1]}$ and $\boldsymbol{z}_{[a_2]}$ are the vector representations of terms a_1 and a_2 respectively, $||\boldsymbol{z}_{[a_1]}||$ and $||\boldsymbol{z}_{[a_2]}||$ is the length of these vectors as computed by the Euclidean norm, and θ is the angle between the vectors. Cosine similarity gives the cosine of the angle between the term representation vectors $\boldsymbol{z}_{[a_1]}$ and $\boldsymbol{z}_{[a_2]}$ (Manning et al., 2008, p. 122). If the angle

between the vectors equals 0°, meaning that the vectors have the exact same orientation, the cosine is 1 (Moore & Siegel, 2013, p. 281). If the angle is 90°, meaning that the vectors are orthogonal to each other, then $cos(\theta) = 0$ (Moore & Siegel, 2013, p. 281).[4]

Frequently used term representations are word embeddings (see e.g. Diaz et al., 2016; Kuzi et al., 2016; Silva & Mendoza, 2020). A word embedding is a real-valued vector representation of a term. Important model architectures to learn word embeddings are the continuous bag-of-words (CBOW) and the Skip-gram models (Mikolov et al., 2013a) as well as Global Vectors (GloVe) (Pennington et al., 2014) and fastText (Bojanowski et al., 2017). In learning the word embedding for a target term a_t, these architectures make use of words occurring in a context window around a_t (Mikolov et al., 2013a, p. 4; Pennington et al., 2014, p. 1533-1535). In doing so, these procedures for learning word embeddings implicitly draw on the distributional hypothesis (Firth, 1957), which states that the meaning of a word can be deduced from the words it typically co-occurs with (Rodriguez & Spirling, 2022, p. 102). This in turn implies that syntactically or semantically similar terms are likely to have similar word embedding vectors that point into a similar direction (Bengio et al., 2003, p. 1139-1140; Mikolov et al., 2013b).

In similarity-based query expansion techniques, terms that are closest to the query terms are used as query expansion terms. The number of terms added varies from approach to approach between five to a few hundred (Azad & Deepak, 2019, p. 1714). In Silva & Mendoza (2020), for example, the original query is represented by a single vector that is computed by taking the weighted average of the word embeddings of all terms in the original query. Then the five terms whose embeddings have the highest cosine similarity with the embedding of the query are selected for expansion.

To sum up, researchers that implement query expansion methods require a data source for expansion, a way to compute a measure that captures the relatedness between terms, and a procedure that determines which and how many terms are added via which process. If they plan to represent terms as word embeddings, then either pretrained word embeddings are required or the embeddings have to be learned. Consequently, considerable resources and expertise is needed. Yet whereas individuals due to inhibitory processes may fail to create a comprehensive list of search terms, query expansion methods can uncover terms that denote the entity of interest and are used in the corpus at hand. As query expansion techniques have the potential to expand the initial query with synonymous and related terms, recall is likely to increase (Manning et al., 2008, p. 193). Precision, however, may decrease—especially if the added terms are homonyms or polysemes (i.e. terms that have different meanings that are conceptually distinct (homonyms) or related (polysemes)) (Manning & Schütze, 1999, p. 110; Manning et al., 2008, p. 193). It thus may be advantageous to use

[4]If the elements of the term vectors are non-negative, e.g. because they indicate the (weighted) frequency with which a term occurs across the documents in the corpus, then the angle between the vectors will be in the range $[0°, 90°]$ and cosine similarity will be in the range $[0, 1]$. If, on the other hand, elements of term representation vectors can become negative, then the vectors can also point into opposing directions. In the extreme, if the vectors point into diametrically opposing directions, then $cos(\theta) = -1$.

as a data source for query expansion a corpus or thesaurus that is specific to the domain of the retrieval task rather than a global corpus or a general thesaurus (Manning et al., 2008, p. 193). Moreover, query expansion techniques require researchers to a priori come up with an initial set of query terms (which will encode the researchers' assumptions), and there is no guarantee that the expansion starting from the initial set will capture all different denominations of the entity. For example, there is no guarantee that query expansion will succeed in moving from *'Biden'* to *'Sleepy Joe'*. Finally, if the entity of interest is also referred to with multi-term expressions (e.g. *'United States'*), then these only can be extracted if the term representations used by the expansion procedure also cover multi-term expressions. Word embeddings would have to be learned or be available also for bigrams and trigrams. This increases the methods' complexity, the computational resources required, and limits the availability of word embeddings that already have been pretrained on external global corpora.[5]

3.3.3 Topic Model-Based Classification Rules

Recently, Baden et al. (2020) have proposed a procedure in which documents are categorized based on classification rules that are built by researchers on the basis of topics estimated by a topic model. Baden et al. (2020) call their procedure Hybrid Content Analysis. The idea is to assign those documents to a pre-defined category that are estimated to be comprised to a considerable degree of topics that the researchers deem to be related to the category (Baden et al., 2020). Whilst Baden et al. (2020) formulate their method for multi-class or multi-label classification tasks in a descriptive manner, here the procedure is presented with precise mathematical expressions and the focus is exclusively on the binary classification task of retrieving relevant documents.

The family of topic models most widely applied in social science are Bayesian hierarchical mixed membership models that estimate a latent topic structure based on observed word frequencies in text documents (Blei et al., 2003, p. 993, 995-997; Blei & Lafferty, 2007, p. 18; Roberts et al., 2016a, p. 988; Zhao et al., 2021, p. 4713-4714). These topic models (which are here simply referred to as *topic models*) assume that each topic is a distribution over the terms in the corpus and each document is characterized by a distribution over topics (Blei et al., 2003, p. 995-997; Blei & Lafferty, 2007, p. 18). Given a corpus of N documents, topic models estimate a latent topic structure defined by $N \times K$ document-topic matrix $\boldsymbol{\Theta}$ and $K \times U$ topic-term matrix $\boldsymbol{B}$ (see Figure 3.1). Topic-term matrix $\boldsymbol{B} = [\boldsymbol{\beta}_1 | \ldots | \boldsymbol{\beta}_k | \ldots | \boldsymbol{\beta}_K]^\top$ gives for each topic, $k \in \{1, \ldots, K\}$, the estimated probability mass function across the U unique terms in the vocabulary: $\boldsymbol{\beta}_k = [\beta_{k1}, \ldots, \beta_{ku}, \ldots, \beta_{kU}]$, where β_{ku} is the probability for the uth term to occur given topic k. Document-topic

[5]Note that besides the similarity-based automatic query expansion approaches discussed so far, there are further expansion methods. Most prominently there are query language modeling and operations based on relevance feedback from the user or pseudo-relevant feedback (Lavrenko & Croft, 2001; Manning et al., 2008, p. 177-188; Azad & Deepak, 2019, p. 1709-1713).

matrix $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1 | \ldots | \boldsymbol{\theta}_i | \ldots | \boldsymbol{\theta}_N]^\top$ contains for each document d_i the estimated proportion assigned to each of K latent topics: $\boldsymbol{\theta}_i = [\theta_{i1}, \ldots, \theta_{ik}, \ldots, \theta_{iK}]$, with θ_{ik} being the estimated share of document d_i assigned to topic k.

Given the estimated latent topic structure characterized by $K \times U$ topic-term matrix $\boldsymbol{B}$ and $N \times K$ document-topic matrix $\boldsymbol{\Theta}$, the topic model-based classification rule building procedure proceeds as follows (see Figure 3.1) (Baden et al., 2020, p. 171-174):

1. Based on $K \times U$ topic-term matrix $\boldsymbol{B}$, the researcher inspects for each topic the most characteristic terms, e.g. the terms that are most likely to occur in a topic and the terms that are the most exclusive for a topic.[6] Given these terms that inform about the content of each topic, the researcher determines which topics refer to the entity of interest. The researcher then creates relevance matrix $\boldsymbol{C}$ of size $K \times 1$ whose elements are 1 if the topic is considered relevant and are 0 otherwise.

2. Then $N \times K$ document-topic matrix $\boldsymbol{\Theta}$ is multiplied with $\boldsymbol{C}$. The resulting vector $\boldsymbol{r} = [r_1, \ldots, r_i, \ldots, r_N]^\top$ gives for each document the sum over those topic shares that refer to relevant topics. r_i can be interpreted as the share of words in document d_i that come from relevant topics.

3. A threshold value $\xi \in [0, 1]$ is set. All documents for which $r_i >= \xi$ are considered to be relevant.

The procedure utilizes a topic model as an unsupervised tool to uncover information about the latent topic structure of a corpus. Leveraging this information for the retrieval of relevant documents allows researchers to operate without a set of explicit keywords. Rather than having to come up with information about to be retrieved documents a priori, researchers merely have to recognize topics that refer to relevant entities. As topic models are well known and frequently developed and applied in social science (e.g. Quinn et al., 2010; Grimmer, 2013; Roberts et al., 2014; Bauer et al., 2017; Maier et al., 2018; Baerg & Lowe, 2020; Eshima et al., 2021; Schulze et al., 2021) and furthermore are implemented in corresponding software packages (e.g. Grün & Hornik, 2011; Roberts et al., 2019), the procedure of building classification rules based on topic models seems to be easily accessible to the social science community.

Nevertheless, estimating a topic model in the first place induces costs. Especially the number of topics K has to be set a priori. To set a useful value for K typically several topic models with varying K are estimated and after a manual inspection of the most likely and most exclusive terms for a topic as well as the computation of performance metrics (e.g. held-out likelihood), researchers decide on a topic number (Roberts et al., 2016b, p. 60-62). Moreover, as topic models are unsupervised there is no way for researchers—beyond setting parameters as K—to guide the estimation process such that the results

[6] The *most likely* terms are the terms with the highest occurrence probabilities, β_{ku}, for a given topic k. The *most exclusive* terms refer to highly discriminating terms whose probability to occur is high for topic k but low for all or most other topics. Exclusivity can be measured as: $exclusivity_{ku} = \beta_{ku} / \sum_{j=1}^{K} \beta_{ju}$ (see for example Roberts et al., 2019, p. 12).

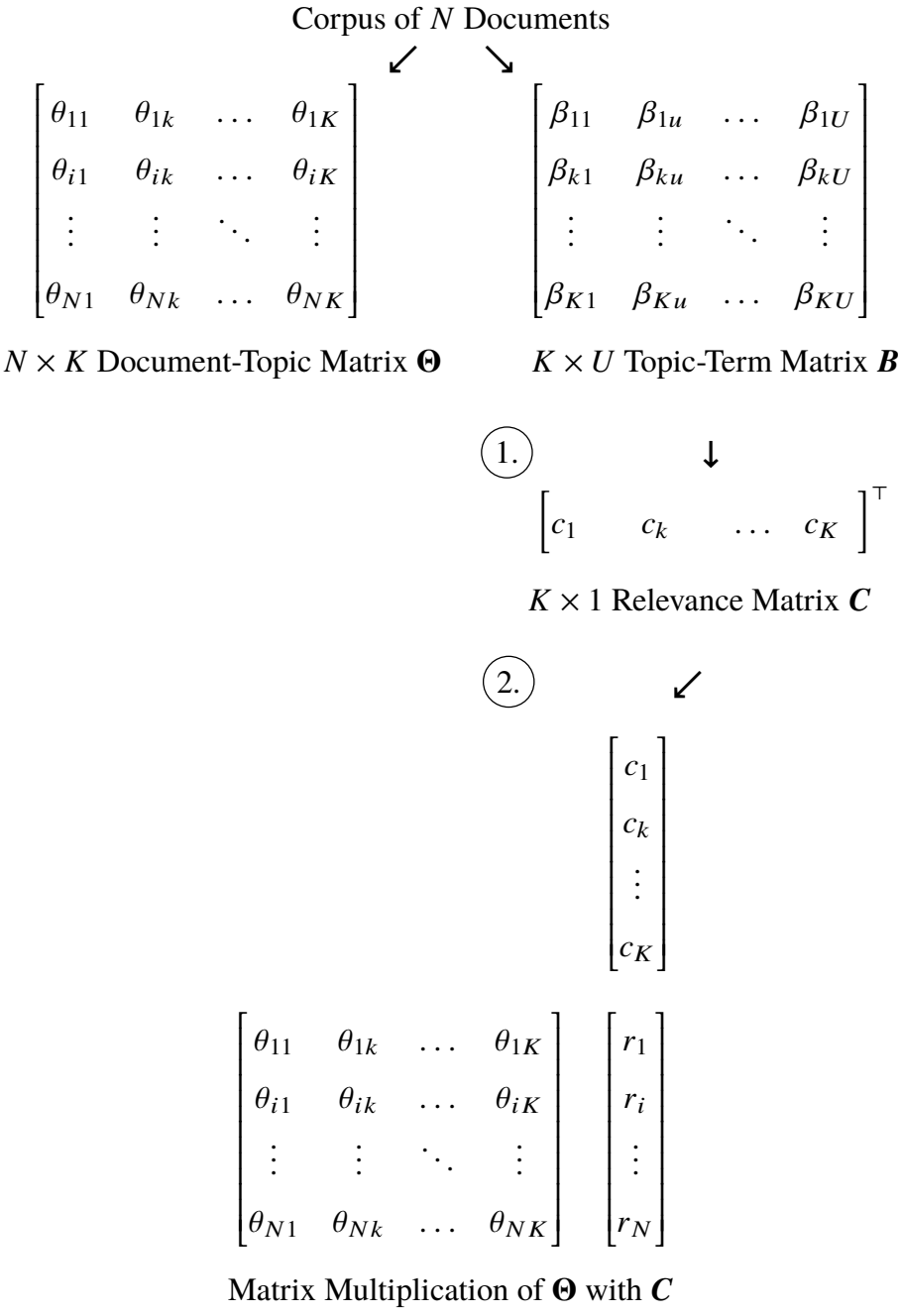


Figure 3.1: **Building Topic Model-Based Classification Rules.** Classification rules can be built from any topic model that on the basis of a corpus comprising N documents estimates a latent topic structure characterized by two matrices: $N \times K$ document-topic matrix $\boldsymbol{\Theta}$ and $K \times U$ topic-term matrix $\boldsymbol{B}$. β_{ku} is the estimated probability for the uth term to occur given topic k. θ_{ik} is the estimated share assigned to topic k in the ith document. The topic model-based classification rule procedure proceeds as follows: Step 1: Researchers inspect matrix $\boldsymbol{B}$, determine which topics are relevant, and create $K \times 1$ relevance matrix $\boldsymbol{C}$. Step 2: Matrix multiplication of $\boldsymbol{\Theta}$ with $\boldsymbol{C}$ yields the resulting vector $\boldsymbol{r}$. Step 3 (not shown): Documents with $r_i >=$ threshold $\xi \in [0, 1]$ are retrieved.

are related to the concepts of interest. Ideally one would like to have a topic model that produces one or several topics that refer to the entity of interest and are characterized by high semantic coherence as well as exclusivity. A coherent topic referring to the entity of interest would have high occurrence probabilities for frequently co-occurring terms that refer to the entity (Roberts et al., 2014, p. 1069; Roberts et al., 2019, p. 10). It would be clearly about the entity of interest rather than being a fuzzy topic without a nameable content. An exclusive topic would solely refer to the entity of interest and would not refer to any other entities.

It is not guaranteed, however, that there is a topic that distinctly covers the relevant entity. Additionally, topic models can generate topics that relate to several entities rather than a single entity. By selecting each topic that refers to the relevant entity but also relates to several non-relevant entities, a researcher will construct a topic model-based classification rule that will be characterized by high recall but low precision. For this reason, Baden et al. (2020, p. 173) suggest setting K to a rather high value such that topics are fine-grained. But whether this will work out in a given application is unclear as the latent topic structure uncovered by the topic model cannot be forced to neatly separate topics referring to relevant entities from topics referring to non-relevant entities.

3.3.4 Passive and Active Supervised Learning

Supervised learning algorithms are trained on the basis of a training data set. The training data set contains a set of documents with corresponding class labels or values. In the context of retrieving relevant documents, a training set document is assigned to the relevant class if it refers to the entity of interest and is assigned to the irrelevant class otherwise. Central to the supervised learning process is the loss function. The loss function returns a cost-signifying value which is a function of the discrepancy between the predicted and the true values of the training set documents. In an optimization process, the parameters of the supervised learning algorithm are moved toward values for which the loss function reaches a (local) minimum.

Supervised learning methods have the advantage that they come with supervision: The separation between relevant and irrelevant documents is encoded in the training data set and then learned by the model. This is a considerable advantage over automatic query expansion methods and topic model-based approaches. In the former, researchers cannot be entirely sure that the expansion really will include terms related to the initial query terms, and in the latter, it is unclear whether there will be coherent and exclusive topics referring to the entity of interest.

Moreover, as the true class assignments for the training set documents are known, supervised learning approaches allow researchers to use resampling techniques (e.g. cross-validation) in order to assess how well the retrieval of relevant documents works. The values for precision and recall not only provide information about the performance of the

retrieval method but also indicate the nature of the (mis)classifications. (Is the model lenient in assigning documents to the positive relevant class and therefore most of the relevant documents are retrieved (high recall) but there are many false positives among the retrieved documents (low precision), or is it rather the other way round?)

Furthermore, just as the topic model-based approach, supervised learning techniques depend on recognizing rather than recalling: When creating the training data set, coders read the training documents and assign them to the relevant vs. irrelevant class as specified in coding instructions. Hence, supervised learning techniques require the coders to merely recognize relevant documents rather than creating information on relevant documents from scratch.

Supervised learning methods, however, also come with disadvantages. First, the labeling of training documents by human coders is extremely costly. Precise coding instructions have to be formulated, the coders have to be trained and paid, and the intercoder reliability (e.g. measured by Krippendorff's α (Krippendorff, 2013, p. 277-294)) has to be assessed. Reading an adequately large sample of documents and labeling each as relevant vs. irrelevant (or having this being done by trained coders) takes time.

Second, in the context of retrieving relevant documents, it is likely that the share of relevant documents is small, and thus further problems arise: If the training set documents are randomly sampled from the entire corpus from which relevant documents are to be identified and only a small share of documents refer to the entity of interest, then a large number of training documents have to be sampled, read, and coded such that the training data set contains a sufficiently large number of documents falling into the positive relevant class for the supervised method to effectively learn the distinctions between the relevant and the irrelevant class. If, for example, 3% of documents are relevant, then after coding 1,000 randomly sampled training documents only about 30 documents will be assigned to the relevant category.[7]

What is more: If no adjustments are made, then each training set document has the same weight in the calculation of the value of the loss function. This is, the optimization algorithm attaches the same importance to the correct classification of each training set document. Yet in a retrieval situation characterized by imbalance, researchers typically care more about the correct classification of relevant training documents than irrelevant documents (see also argumentation in Section 3.2 above) (Branco et al., 2016, p. 2-4). Or put differently, missing a truly relevant document (false negative) is considered more problematic than falsely predicting an irrelevant document to be relevant (false positive) (Brownlee, 2020). Thus, there is the question of what to do to make the supervised learning algorithm focus on correctly detecting relevant documents.

The statistical learning community has devised a large spectrum of approaches to deal

[7]Note that research suggests that it is rather the number of training examples in the positive relevant class than the number of all documents in the training set that affects the amount of information provided to the learning method (Wang, 2020).

with imbalanced classification problems (for an overview see Branco et al., 2016). Among the most common and most easily applicable procedures that are employed to make the optimization algorithm put more weight on the correct classification of instances that are part of the relevant minority class are techniques that adjust the distribution of training set instances (Branco et al., 2016, p. 7-15, 21-27). This set of techniques comprises procedures such as random oversampling, random undersampling, and the synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002) (Branco et al., 2016, p. 22).[8] In random oversampling, minority class instances are randomly resampled with replacement and appended as exact replicas to the training data set (Wang, 2020, p. 9833). In random undersampling, randomly selected instances of the majority class are removed from the training set (Wang, 2020, p. 9831). Both resampling techniques are typically applied until a user-specified distribution of class labels is reached (e.g. until the minority class contains as many instances as the majority class) (Brownlee, 2021a). Thereby, both resampling strategies make the training set more balanced and thus put more weight on the minority class than in the original training set distribution. As random oversampling implies that resampled minority instances are added as exact duplicates, random oversampling can lead to overfitting on the training data and reduced generalization performance on the test data (Branco et al., 2016, p. 22). Moreover, oversampling implies higher computational costs (Branco et al., 2016, p. 22). In random undersampling, on the other hand, information from removed majority class instances is lost (Brownlee, 2021a).[9]

In addition to these techniques that adjust the training set distribution, a second set of methods to address imbalanced classification problems is the usage of cost-sensitive algorithms (Branco et al., 2016, p. 27 ff.). There are specifically developed modifications of algorithms that allow for incorporating higher costs for misclassifying instances of the minority class (for an overview of these special-purpose methods see Branco et al., 2016, p. 27-29). A more general method, however, is to set up a cost matrix that specifies which cell in the confusion matrix (see Table 3.1 in Section 3.2) is associated with which cost (Elkan, 2001; Brownlee, 2020). During training, the loss of each training instance takes into account the respective cost depending on which cell the instance is in (Elkan, 2001,

[8]SMOTE (Chawla et al., 2002) is a well-known technique in which the minority class is enlarged by adding synthetically generated minority class training examples. A synthetic training instance is created by the following process: For each feature, a feature value is randomly drawn from the line joining the feature value of a randomly sampled minority class instance and the feature value of one of its Q nearest neighbors (Chawla et al., 2002, p. 328-329). This implies that SMOTE is "operating in 'feature space' rather than 'data space"' (Chawla et al., 2002, p. 328) of data that are represented in tabular form (Brownlee, 2021b). (Thus, SMOTE can be applied on a bag-of-words-based document-feature matrix but not original sequential text data.) In contrast to a simple random oversampling procedure, SMOTE adds new instances rather than exact copies to the training data and thereby reduces the risk of overfitting (Chawla, 2005, p. 860). (Note that typically SMOTE is combined with random undersampling (Chawla et al., 2002, p. 330). There are various modifications of SMOTE or combinations of SMOTE with other techniques and models (Branco et al., 2016, p. 25-26).)

[9]Besides these random resampling techniques mentioned here, there are methods that perform oversampling or undersampling in an informed way; e.g. based on distance criteria (see Branco et al., 2016, p. 23-24).

p. 973). In this way, higher costs can be specified for false negatives than for false positives and be directly incorporated into the training process.

The idea of the cost matrix also underlies the techniques that modify the distribution of training instances (Elkan, 2001, p. 975). The undersampling rates for the majority class or the oversampling rates for the minority class ideally should reflect the cost induced by misclassifying an instance from the respective class (Brownlee, 2020). For example, if falsely predicting an instance from the positive minority class to be negative is considered 10 times more costly than falsely predicting an instance from the negative majority class to be positive, then the cost of a false negative is 10, and the cost for a false positive 1 (and true positives and true negatives induce no costs) (Branco et al., 2016, p. 36). Positive minority class instances then could be randomly oversampled such that their number increases by a factor of 10, or the majority class instances could be undersampled such that their number decreases by a factor of 1/10 (Branco et al., 2016, p. 36).[10]

In practice, however, all discussed techniques suffer from the problem that researchers often cannot specify precise values for misclassification costs (Brownlee, 2020). In the context of the task of retrieving relevant documents, researchers may be able to say that false negatives are more costly than false positives but how much so is likely to be highly difficult to specify (Branco et al., 2016, p. 3; Brownlee, 2020).

The focus of the so far mentioned methods for imbalanced classification problems has been on the difference in the misclassification costs associated with instances from the positive minority vs. negative majority class. Yet there are other types of costs that also should be considered: As elaborated above, the annotation of training documents is costly due to the resources required. And in the context of imbalanced classification problems annotating a random sample of documents is inefficient as a disproportionately large number of documents has to be annotated until an acceptable number of instances from the minority class is labeled. These training set annotation costs are the focus of active learning strategies.

Active learning refers to learning techniques in which the learning algorithm itself indicates which training instances should be labeled next (Settles, 2010, p. 4). The idea is to let the learning algorithm select instances for labeling that are likely to be informative for the learning process (Settles, 2010, p. 5). Such instances could be, for example, those instances whose prediction the learner is most uncertain about (Settles, 2010, p. 5). The underlying

[10]Note that the outlined relationship between cost ratios and over- or undersampling rates only holds if the threshold at which the classifier considers an instance to fall into the positive rather than the negative class is at $p = 0.5$ (Elkan, 2001, p. 975). Note furthermore that although it would be good practice for resampling rates to reflect an underlying distribution of misclassification costs as specified in the cost matrix, resampling with rates reflecting misclassification costs will not yield the same results as incorporating misclassification costs into the learning process (Branco et al., 2016, p. 36). One reason, for example, is that in random undersampling instances are removed entirely (Branco et al., 2016, p. 36). For information on the relationship between oversampling/undersampling, cost-sensitive learning, and domain adaptation see Kouw & Loog (2019, p. 4-5, 7).

hypothesis is that by letting the learner actively select the instances from which it seeks to learn, an as high as possible prediction accuracy can be achieved with an as small as possible number of annotated training instances (Settles, 2010, p. 4, 5). Active learning stands in contrast to the usual supervised learning procedure in which the training set instances are being randomly sampled, annotated, and then handed over to the learning algorithm. When juxtaposing active learning to this usual supervised learning procedure, the latter sometimes is called passive learning (Miller et al., 2020, p. 534).

Active learning is useful in situations in which unlabeled training instances are abundant but the labeling process is costly (Settles, 2010, p. 4). There are several different scenarios in which active learning can be applied (see Settles, 2010, p. 8-12). In this study, the focus is on pool-based sampling. In pool-based sampling, a large collection of instances has been collected from some data distribution in one step (Settles, 2010, p. 11). At the start of the learning algorithm, labels are obtained only for a very small set of instances, denoted $\mathcal{I}$, whilst the other instances are part of the large pool of unlabeled instances $\mathcal{U}$ (Settles, 2010, p. 11). In each iteration of the active learning algorithm, the algorithm is trained on instances in the labeled set $\mathcal{I}$ and makes predictions for all instances in pool $\mathcal{U}$ (Lewis & Gale, 1994, p. 4; Settles, 2010, p. 6, 11). The instances in pool $\mathcal{U}$ then are ranked according to how much information the learner would gather from an instance if it were labeled (Settles, 2010, p. 11-12). Then the most informative instances in $\mathcal{U}$ are selected and labeled (e.g. by human coders) (Settles, 2010, p. 6). The newly labeled instances are added to set $\mathcal{I}$ and a new iteration starts (Settles, 2010, p. 6).[11]

In the active learning community, several different strategies of how the informativeness of an instance is defined and how the most informative instances are selected have been developed (for an overview see Settles, 2010, p. 12 ff.). These strategies are termed query strategies (Settles, 2010, p. 12). Here, the "[p]erhaps [...] simplest and most commonly used query framework" (Settles, 2010, p. 12) will be presented: uncertainty sampling (Lewis & Gale, 1994). In uncertainty sampling, those instances are considered to be the most informative about which the learning algorithm expresses the highest uncertainty (Lewis & Gale, 1994, p. 4). In the context of the binary document retrieval classification task, the uncertainty could be said to be highest for instances for which the predicted probability to belong to the relevant class is closest to 0.5 (Lewis & Gale, 1994, p. 4).[12] The usage of such a definition of uncertainty and informativeness is only possible for learning methods that return predicted probabilities (Settles, 2010, p. 12). For methods that do not, other

[11]Ideally, a single instance is selected and labeled in each iteration (Lewis & Gale, 1994, p. 4). Yet re-training a model is often costly and time-consuming. An economic alternative is batch-mode active learning (Settles, 2010, p. 35). Here a batch of instances is selected and labeled in each iteration (Settles, 2010, p. 35). When selecting a batch of instances, there is the question of which instances to select. Selecting the K most informative instances is one strategy that, however, ignores the homogeneity of the selected instances (Settles, 2010, p. 35). Alternative approaches that seek to increase the heterogeneity among the selected instances have been developed (see Settles, 2010, p. 35).

[12]Note that in multi-class classification tasks, it is less straightforward to operationalize uncertainty. Here one can distinguish between least confident sampling, margin sampling, and entropy-based sampling (for precise definitions see Settles, 2010, p. 12-13).

uncertainty-based sampling strategies have been developed (see Settles, 2010, p. 14-15). With regard to SVMs, Tong & Koller (2001) have introduced three theoretically motivated query strategies. In their Simple Margin strategy, the data point that is closest to the hyperplane is selected to be labeled next (Tong & Koller, 2001, p. 53-54).

One important aspect to be kept in mind when applying active learning techniques is that because the training instances are not sampled randomly from the underlying corpus but are purposefully selected, the distribution of the class labels in training data set $\mathcal{I}$ and in unlabeled pool $\mathcal{U}$ is different from the distribution of labels in the entire corpus (Miller et al., 2020, p. 539). If the expected generalization error is to be estimated, then one option is to randomly sample a set of instances from the corpus at the very start of the analysis (Tong & Koller, 2001, p. 57; Miller et al., 2020, p. 539, 541). This set then is annotated and set aside such that it neither can become part of set $\mathcal{I}$ nor set $\mathcal{U}$ (Tong & Koller, 2001, p. 57; Miller et al., 2020, p. 539, 541). After each learning iteration or a fixed number of iterations, the performance of the active learning algorithm then can be evaluated on this independent test set (Tong & Koller, 2001, p. 57; Miller et al., 2020, p. 539, 541).

Empirically, one can say that in a majority of published works active learning reaches the same level of prediction accuracy with fewer training instances than supervised learning with random sampling of training instances (passive learning) (Lewis & Gale, 1994; Tong & Koller, 2001; Ertekin et al., 2007; Settles, 2010; Miller et al., 2020). This is especially the case if data sets are imbalanced (Ertekin et al., 2007, p. 131; Ein-Dor et al., 2020, p. 7954; Miller et al., 2020, p. 543-544). Closer inspections show that during the learning process, the training set $\mathcal{I}$, which is selected by the active learning algorithm, is more balanced (or over the course of active learning iterations becomes more balanced) than the original data distribution (Ertekin et al., 2007, p. 133-134; Miller et al., 2020, p. 545). One likely reason for this observation is that active learning algorithms tend to pick instances for labeling from the uncertain region between the classes, and in this region of the feature space, the class distribution tends to be more balanced (Ertekin et al., 2007, p. 129, 133-134). A more balanced distribution implies that more weight is given to the minority class instances. Another likely reason for the superior performance and efficiency of active compared to passive learning is that because active learning algorithms tend to pick instances close to the boundary between the classes, they are able to learn the class boundary with a smaller number of training instances (Settles, 2010, p. 28).

3.4 Comparison

In the following section, retrieving documents via keyword lists is compared to a query expansion technique, topic model-based classification rules and active as well as passive supervised learning on the basis of three retrieval tasks. The source code of this analysis can be accessed via figshare at https://doi.org/10.6084/m9.figshare.19699840. The analysis is

conducted in R (R Core Team, 2020) and Python (van Rossum & Drake, 2009).[13]

3.4.1 Data

Twitter: The first inspected retrieval task operates on a corpus comprising 24,420 German tweets. These tweets are a random sample of all tweets in German language in a larger collection of tweets that has been collected by Barberá (2016). Linder (2017) sampled 24,420 German tweets and used CrowdFlower workers to label the sampled tweets. For each tweet, the label indicates whether the tweet refers to refugees, refugee policies, and the refugee crisis and thus is considered relevant or not (Linder, 2017, p. 23-24). The task of retrieving the relevant tweets from this corpus indeed is an imbalanced classification problem as only 727 out of the 24,420 tweets (2.98%) are labeled to be about the refugee topic.

SBIC: The aim of the second retrieval task is to extract all posts from the Social Bias Inference Corpus (SBIC) (Sap et al., 2020) that have been labeled to be offensive toward mentally or physically disabled people. The SBIC includes 44,671 potentially toxic and offensive posts from Reddit, Twitter, and three websites of online hate communities (Sap et al., 2020, p. 5480).[14] The SBIC was collected with the aim of studying implied—rather than explicitly stated—social biases (Sap et al., 2020, p. 5477). The subreddits and websites selected to be included in the SBIC constitute intentionally offensive online communities (Sap et al., 2020, p. 5480). The additionally included Reddit comments and tweet data sets were collected such that there is an increased likelihood that the content of the collected posts is offensive (e.g. by selecting tweets that include hashtags known to be racist or sexist) (Sap et al., 2020, p. 5480). Sap et al. (2020) used Amazon Mechanical Turk for the annotation of the posts. For each post, the coder indicated, amongst others, whether the post is offensive and if so, whether the target is an individual (meaning that the post is a personal insult) or a group (implying that the post offends a social group, e.g. women, people of color) (Sap et al., 2020, p. 5479-5480). If one or several groups were targeted, the coders were asked to name the targeted group or groups (Sap et al., 2020, p. 5479-5480). The authors merge the 1,414 targeted groups into seven larger group categories (Sap et al., 2020, p. 5481). One of these group categories is mentally or physically disabled

[13]For the analyses pertaining to active and passive supervised learning with the pretrained language representation model BERT, the Python code is run in Google Colab (Google Colaboratory, 2020) in order to have access to a GPU. The employed R packages are data.table (Dowle & Srinivasan, 2020), dplyr (Wickham et al., 2021), facetscales (Oller Moreno, 2021), ggplot2 (Wickham, 2016), lsa (Wild, 2020), plot3D (Soetaert, 2019), quanteda (Benoit et al., 2018), RcppParallel (Allaire et al., 2020), rstudioapi (Ushey et al., 2020), stm (Roberts et al., 2019), stringr (Wickham, 2019), text2vec (Selivanov et al., 2020), and xtable (Dahl et al., 2019). The used Python packages and libraries are Beautiful Soup (Richardson, 2020), gdown (Kentaro, 2020), imbalanced-learn (Lemaître et al., 2017), matplotlib (Hunter, 2007), NumPy (Oliphant, 2006), pandas (McKinney, 2010), seaborn (Michael Waskom and Team, 2020), scikit-learn (Pedregosa et al., 2011), PyTorch (Paszke et al., 2019), watermark (Raschka, 2020), and Hugging Face's Transformers (Wolf et al., 2020). If a GPU was used, an NVIDIA Tesla P100-PCIE-16GB was employed.

[14]For a detailed elaboration about the exact composition of the SBIC see Sap et al. (2020, p. 5480)

people. 2.15% of the 44,671 posts are annotated as being offensive toward the disabled.[15] The category of disabled people is selected as the focus of this study because this group category is the most coherent capturing a well-defined group of people.

Reuters: The third retrieval task is to identify all newspaper articles in the Reuters-21578 corpus (Lewis, 1997) that refer to the topic surrounding crude oil. Reuters-21578 (Lewis, 1997) is a widely used corpus for evaluating retrieval approaches (Tong & Koller, 2001; Ertekin et al., 2007; Hugging Face, 2021). The corpus contains 21,578 newspaper articles that were published on the Reuters financial newswire service in 1987 (Lewis, 1997; Hugging Face, 2021). 10,377 articles are assigned to one or several out of 135 economic subject categories called topics (Lewis, 1997). These categories are e.g. *'gold'*, *'grain'*, *'cotton'*. Here, the 10,377 topic-annotated articles are used for the analysis. The aim is to identify the 566 (5.45%) newspaper articles that are labeled to be about the crude oil topic. The topic is the fourth largest. It is large enough to possibly contain enough documents for the algorithms to learn from and at the same time is small enough such that the identification of crude oil articles can be considered an imbalanced classification problem.

The three data sets employed here are selected with the aim to achieve and represent various types of retrieval tasks common in social science. Tweets, posts from online platforms, and newspaper articles are types of documents that are often analyzed in social science and whose analysis typically involves some preliminary retrieval step (see e.g. King et al., 2013; Beauchamp, 2017; Baum et al., 2018; Stier et al., 2018; Fogel-Dror et al., 2019; Zhang & Pan, 2019; Watanabe, 2021; Muchlinski et al., 2021). The entities of interest in social science studies vary widely with regard to their nature and their level of abstraction. Zhang & Pan (2019) study collective action events; Baum et al. (2018) focus on rape incidents; Puglisi & Snyder (2011) retrieve information on persons involved in political scandals; Uyheng & Carley (2020) extract tweets referring to the COVID-19 pandemic; Jungherr et al. (2016) examine parties, candidates, and campaign events during an election campaign; and the entities of interest for Fogel-Dror et al. (2019) are Israel and the Palestinian Authority. In this study, the entities of interest include a multi-dimensional topic that covers abstract policies, occurrences as well as a social group (refugee policies, refugee crisis, refugees), a one-dimensional topic about a single economic product (crude oil), and a specific social group (disabled people) that is referred to in a specific (namely: offending) way. Moreover, the corpora from which documents are retrieved in social science can be thematically highly heterogeneous (as is the case with the corpus of German tweets here and with the Weibo posts studied by Zhang & Pan (2019)). Corpora, however, also can be more homogeneous with regard to topics, linguistic style, or attitudes (see e.g. the corpus of speeches from leaders of EU institutions and member states employed by Rauh et al. (2020) and the SBIC corpus here). Note also that the task of retrieving posts that offend disabled people in-

[15]Note that each post was annotated by three independent coders and that the data shared by Sap et al. (2020) lists each annotation separately. Here the SBIC is preprocessed such that the post is considered to be offensive toward a group category if at least one annotator indicated that a group falling into this category was targeted.

volves retrieving posts that are of a specific kind (namely: offending) and refer to a specific entity (disabled people). Such a retrieval task is a common first step in sentiment analyses in which the aim is to extract documents that express an attitude toward a specific entity. The documents to be identified in such cases are required not only to refer to the entity of interest but also to be of a specific kind (namely: attitude expressing in contrast to being objective or fact-based).

3.4.2 Implementation of Approaches

3.4.2.1 Keyword Lists

In order to compare the retrieval performance of keyword lists with the other discussed methods, keyword lists have to be generated for each of the three retrieval tasks. Due to what is known from research on the human construction of keyword lists, however, the keyword lists created by humans are likely to overlap very little and thus are likely to be unreliable (King et al., 2017, p. 973-975). This poses a problem for the planned comparison because it would be best to have a challenging and reliable basis against which the other approaches can be compared to. To address this problem, the keyword lists are not constructed by humans but rather from the set of the most predictive keywords for the positive relevant class.

To identify predictive keywords, for each of the three studied corpora, the documents are preprocessed into a document-feature matrix.[16] Then, logistic regression with regularization is applied. The regularization is introduced via the least absolute shrinkage and selection operator (LASSO; L^1 penalty) or ridge regression (L^2 penalty) depending on the outcome of hyperparameter tuning. The model is trained on the entire data set and then the 50 most predictive terms (i.e. the terms with the highest coefficients) are extracted. The extracted terms are listed in Tables 3.A.1 to 3.A.3 in Appendix 3.A. From the set of 50 most predictive terms, 10 keywords are randomly sampled, where the probability of drawing a term is proportional to the relative size of the term's coefficient. The 10 sampled keywords constitute one keyword list. The sampling of keywords from the set of predictive terms is repeated 100 times such that, for each evaluated corpus, there are 100 keyword lists of length 10 that serve as a basis for evaluation and comparison.[17]

[16]The documents are preprocessed by tokenization into unigrams, lowercasing, removing terms that occur in less than 5 documents or less than 5 times throughout the corpus, and applying a Boolean weighting on the entries of the document-feature matrix such that a 1 signals the occurrence of a term in a document and a 0 indicates the absence of the term in a document.

[17]Note that the keyword lists comprising empirically highly predictive terms are not only applied on the corpora to evaluate the retrieval performance of keyword lists, but also form the basis for query expansion (see Section 3.4.2.2). The query expansion technique makes use of GloVe word embeddings (Pennington et al., 2014) trained on the local corpora at hand and also makes use of externally obtained GloVe word embeddings trained on large global corpora. In the case of the locally trained word embeddings, there is a learned word embedding for each predictive term. Thus, the set of extracted highly predictive terms

In contrast to human-constructed keyword lists for which it would be difficult to judge whether the lists perform on the higher or lower end of all lists that humans could possibly generate for the posed retrieval tasks, the here constructed keyword lists mark the situation of a good start in which the selected keywords are highly indicative for the relevant class.

3.4.2.2 Query Expansion

The keyword lists serve as the starting point for query expansion. Each keyword list is expanded via the following procedure:

1. Take a set of trained word embeddings, here denoted by $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_u, \ldots, \boldsymbol{z}_U\}$.[18]
2. For each keyword s_v in the keyword list $\{s_1, \ldots, s_V\}$:
   (a) Get the word embedding of the keyword: $\boldsymbol{z}_{[s_v]}$
   (b) Compute the cosine similarity between $\boldsymbol{z}_{[s_v]}$ and each word embedding $\boldsymbol{z}_u$ in the set $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_u, \ldots, \boldsymbol{z}_U\}$:

$$sim_{cos}(s_v, z_u) = \frac{\boldsymbol{z}_{[s_v]} \cdot \boldsymbol{z}_u}{||\boldsymbol{z}_{[s_v]}||\ ||\boldsymbol{z}_u||} \tag{3.6}$$

   (c) Take the M terms that are not keyword s_v itself and have the highest cosine similarity with keyword s_v. Add these M terms to the keyword list.

This query expansion strategy makes use of word embedding representations and the cosine similarity as has been done in previous studies (e.g. Kuzi et al., 2016; Silva & Mendoza, 2020). By not merging the keyword list into a single word vector representation but rather expanding the keyword list for each keyword separately, this expansion method allows moving into a different direction for each keyword. This might help in extracting a more varied range of linguistic denominations for the entity of interest and might be especially useful if the entity is abstract or combines several dimensions (such as e.g. is the case with the refugee topic that combines policies, occurrences, and a group of people). A similar procedure for query expansion has been studied by Kuzi et al. (2016).

can be directly used as starting terms for query expansion. In the case of the globally pretrained word embeddings, however, not all of the highly predictive terms have a corresponding global word embedding. Hence, for the globally pretrained embeddings, the 50 most predictive terms *for which a globally pretrained word embedding is available* are extracted. If a predictive term has no corresponding global embedding, the set of extracted predictive terms is enlarged with the next most predictive term until there are 50 extracted terms. Consequently, in Tables 3.A.1 to 3.A.3 in Appendix 3.A, for each corpus two lists of the most predictive features are shown. Moreover, for the evaluation of the initial keyword lists of 10 predictive keywords, the local keyword lists have to be used because the global keyword lists have been adapted for the purpose of query expansion on the global word embedding space.

[18] If necessary, the set of word embeddings is reduced to those embeddings whose terms occur in the corpus of interest and in the keyword list.

For each evaluated retrieval task, two different sets of word embeddings are used: embeddings that have been externally pretrained on large global corpora and embeddings trained locally on the corpus from which documents are to be retrieved. With regard to the globally pretrained embeddings, for the English SBIC and the Reuters corpus, GloVe embeddings with 300 dimensions that have been trained on CommonCrawl data are made use of (Pennington et al., 2014).[19] For the German Twitter data set, 300-dimensional GloVe embeddings trained on the German Wikipedia are employed.[20]

To get locally trained embeddings, on each corpus examined here, a GloVe embedding model is trained. GloVe embeddings with 300 dimensions are obtained for all unigram features that occur at least 5 times in the corpus. In training, a symmetric context window size of six tokens on either side of the target feature as well as a decreasing weighting function is used (such that a token that is q tokens away from the target feature counts $1/q$ to the co-occurrence count) (Pennington et al., 2014). After training, following the approach in Pennington et al. (2014), the word embedding matrix and the context word embedding matrix are summed to yield the finally applied embedding matrix. Note that in an analysis of a large spectrum of settings for training word embeddings, Rodriguez & Spirling (2022) found that the here used popular setting of using 300-dimensional embeddings with a symmetric window size of six tokens tends to be a setting that yields good performances whilst at the same time being cost-effective.

The number of expansion terms M is set to increase from 1 to 9 such that after the expansion the lists of originally 10 keywords then comprise between 20 and 100 keywords. The original and the expanded keyword lists are applied on the lowercased documents. Following the logic of a Boolean query with the OR operator, a document is predicted to belong to the positive relevant class if it contains at least one of the keywords in the keyword list.

3.4.2.3 Topic Model-Based Classification Rules

When constructing topic model-based classification rules, there are three steps at which researchers have to make decisions that are likely to substantively affect the results. First, after having selected a specific type of topic model that is to be used, the number of to be estimated topics K has to be set. Second, for the construction of a topic model-based classification rule, a researcher has to determine how many and which of the estimated topics are considered to be about the entity of interest. Finally, threshold value $\xi \in [0, 1]$ has to be set. If the sum of topic shares relating to relevant topics in a document is $\geq \xi$, the document is predicted to be relevant. In each of these decision steps, a researcher may

[19]The embeddings can be downloaded from https://nlp.stanford.edu/projects/glove/. GloVe embeddings here are used because they tend to be frequently employed in social science (Rodriguez & Spirling, 2022, p. 104).

[20]The embeddings can be downloaded from https://deepset.ai/german-word-embeddings.

be guided by expertise and/or an exploration of the results that are caused by deciding on one or another option.

Whilst in practice a researcher has to finally settle for one of the options in each step such that a single classification rule is produced, here the aim rather is to comprehensively evaluate topic model-based classification rules and also to inspect how well topic model-based classification rules can perform if optimal decisions (w.r.t. retrieval performance) are made. Consequently, specific values for the number of topics, the number of relevant topics, and threshold values are set within reasonable ranges a priori. Then, the retrieval performance for all combinations of these values is evaluated. More precisely: On each corpus, seven topic models—each with a different number of topics $K \in \{5, 15, 30, 50, 70, 90, 110\}$—are estimated. Then, for each estimated topic model with a specific topic number, initially, only one topic is considered relevant, then two topics, and then three. For each number of topics considered to be relevant, all possible combinations regarding the question *which* topics are considered relevant are evaluated. This implies that all ways of choosing one, two, and three relevant topics (irrespective of the order in which they are selected) from the overall sets of 5, 15, 30, 50, 70, 90, and 110 topics are determined and evaluated. This amounts to 426,725 combinations—all of which are evaluated here.[21]

Finally, for each of the 426,725 combinations, four different threshold values ξ are inspected: $0.1, 0.3, 0.5$, and 0.7. Whereas $\xi = 0.7$ only considers those documents to be relevant that have 70% of the words they contain estimated to be generated by relevant topics, $\xi = 0.1$ is the most lenient solution in which all documents are classified to be relevant that have 10% of their words assigned to relevant topics. As ξ increases, recall is likely to decrease and precision is likely to increase.

The type of topic model estimated here is a Correlated Topic Model (CTM) (Blei & Lafferty, 2007). CTM extends the basic Latent Dirichlet Allocation (LDA) (Blei et al., 2003) by allowing topic proportions to be correlated. For more details on the CTM see Blei & Lafferty (2007).[22]

[21] For example, in a topic model with $K = 15$ topics, there are 15 ways to select one relevant topic from 15 topics (namely: the first, the second, ..., and the 15th); and there are $\binom{15}{2} = 105$ ways of choosing two relevant topics from the set of 15 topics, and there are $\binom{15}{3} = 455$ ways to pick three topics from 15 topics.

[22] The CTM is estimated via the stm R-package (Roberts et al., 2019) that is originally designed to estimate the Structural Topic Model (STM) (Roberts et al., 2016a). If no document-level variables are specified in the STM (as is done here), the STM reduces to the CTM (Roberts et al., 2016a, p. 991). In estimation, the approximate variational expectation-maximization algorithm as described in Roberts et al. (2016a, p. 992-993) is employed. This estimation procedure tends to be faster and tends to produce higher held-out log-likelihood values than the original variational approximation algorithm for the CTM presented in Blei & Lafferty (2007) (Roberts et al., 2019, p. 29-30). The model is initialized via spectral initialization (Arora et al., 2013; Roberts et al., 2016b, p. 82-85; Roberts et al., 2019, p. 11). The model is considered to have converged if the relative change in the approximate lower bound on the marginal likelihood from one step to the next is smaller than 1e-04 (Roberts et al., 2016a, p. 992; Roberts et al., 2019, p. 10, 28).

3.4.2.4 Active and Passive Supervised Learning

Two types of supervised learning methods are employed. First, support vector machines (SVMs) (Boser et al., 1992; Cortes & Vapnik, 1995), and second, BERT (standing for Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019).

SVMs have been applied frequently and relatively successfully to text classification tasks in social science (Diermeier et al., 2011; D'Orazio et al., 2014; Baum et al., 2018; Pilny et al., 2019; Sebők & Kacsuk, 2021; Erlich et al., 2021). They also have been applied in active learning settings (Miller et al., 2020). An SVM operates on a document-feature matrix $\boldsymbol{X}$. In a document-feature matrix, each document is represented as a feature vector of length U: $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iu}, \ldots, x_{iU}]$. The information contained in the vector's entries, x_{iu}, is typically based on the frequency with which each of the U textual features occurs in the ith document (Turney & Pantel, 2010, p. 147). Given the document feature vectors, $(\boldsymbol{x}_i)_{i=1}^N$, and corresponding binary class labels, $(y_i)_{i=1}^N$, where $y_i \in \{-1, +1\}$, an SVM tries to find a hyperplane, that separates the training documents as well as possible into the two classes (Cortes & Vapnik, 1995).[23]

The document-feature matrix regards each document as a *bag of words* (Turney & Pantel, 2010, p. 147). A bag-of-words-based representation only encodes information on the weighted frequency with which the terms occur in documents but disregards word order, contextual information, and dependencies between the tokens in a document (Turney & Pantel, 2010, p. 147). But a document is a sequence—not a bag—of tokens among which dependencies exist. Moreover, the meaning of a word often depends on the context of other words in which it is embedded in. (Take, for instance, the homonyms *'bank'* or *'party'*.) In order to also use a supervised learning method that processes a document as a sequence

[23]To create the required vector representation for each document, here the following text preprocessing steps are applied: The documents are tokenized into unigram tokens. Punctuation, symbols, numbers, and URLs are removed. The tokens are lowercased and stemmed. Subsequently, terms whose mean tf-idf value across all documents in which they occur belongs to the lowest 0.1% (Twitter, SBIC) or 0.2% (Reuters) of mean tf-idf values of all terms in the corpus are discarded. Also, terms that occur in only one (Twitter) or two (SBIC, Reuters) documents are removed. Finally, a Boolean weighting scheme, in which only the absence (0) vs. presence (1) of a term in a corpus is recorded, is applied on the document-feature matrix. To determine the hyperparameter values for the SVMs, hyperparameter tuning via a grid search across sets of hyperparameter values is conducted in a stratified 5-fold cross-validation setting on one fold of the training data. A linear kernel and a Radial Basis Function (RBF) kernel are tried. Moreover, for the inverse regularization parameter $\mathcal{C}$, that governs the trade-off between the slack variables and the training error, the values $\{0.1, 1.0, 10.0, 100.0\}$ (linear) and $\{0.1, 1.0, 10.0\}$ (RBF) are inspected. Additionally, for the RBF's parameter γ, which governs the training example's radius of influence, the values $\{0.001, 0.01, 0.1\}$ are evaluated. (On the precise definition of $\mathcal{C}$ and γ see scikit-learn Developers (2020b) and scikit-learn Developers (2020a).) The folds are stratified such that the share of instances falling into the relevant minority class is the same across all folds. In each cross-validation iteration, in the folds used for training, random oversampling of the minority class is conducted such that the number of relevant minority class examples increases by a factor of 5. Among the inspected hyperparameter settings, the setting that achieves the highest F_1-Score regarding the prediction of the relevant minority class and does not exhibit excessive overfitting is selected.

of tokens and captures dependencies between tokens as well as context-dependent meanings of tokens, the Transformer-based language representation model BERT is additionally employed here.

BERT is a deep neural network based on the Transformer architecture (Vaswani et al., 2017). The central element of the Transformer architecture is the (self-)attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017). This mechanism allows the representation of each token to include information from the representations of other tokens (in the same sequence) (Vaswani et al., 2017, p. 6001-6002); thereby enabling the model to produce token representations that encode contextual information and token dependencies.

Typically, BERT is applied in a sequential transfer learning setting (Devlin et al., 2019, p. 4175, 4179). In sequential transfer learning, a model first is pretrained on a source task (Ruder, 2019, p. 64). In pretraining, the aim is to learn model parameters such that the model can function as a well-generalizing input to a large range of different target tasks (Ruder, 2019, p. 64). Then, in the following adaptation phase, the pretrained model (with its pretrained parameters) serves as the input for the training process on the target task (Ruder, 2019, p. 64). The transferral of information (in the form of pretrained model parameters) to the learning process of a target task tends to reduce the number of training instances required to reach the same level of prediction performance than when not applying transfer learning and training the model from scratch (Howard & Ruder, 2018, p. 334).

This characteristic of pretrained deep language representation models to reduce the number of required training instances is highly important for the application of deep neural networks in practice: In text classification tasks, deep neural networks tend to outperform conventional machine learning methods (such as SVMs) that are often applied on bag-of-words representations (Socher et al., 2013; Ruder, 2020). But deep neural networks have a much higher number of parameters to learn than conventional models and thus require much more training instances. In situations in which the annotation of training instances is expensive or inefficient—such as in the context of retrieval with a strong imbalance between the relevant vs. irrelevant class—applying a deep neural network from scratch may become prohibitively expensive. In a transfer learning setting, however, an already pretrained deep language representation model merely has to be fine-tuned to the target task at hand. If the pretrained model generalizes well, the number of training instances required to reach the same level of performance as a deep neural network that is not used in a transfer learning setting is reduced by several times (Howard & Ruder, 2018, p. 334). This allows deep neural networks to be applied to natural language processing tasks for which only relatively few training instances are available. Moreover, Ein-Dor et al. (2020) show that especially in imbalanced classification settings active learning strategies can further improve the prediction performance of BERT such that even fewer training instances are needed for the same performance levels.[24]

[24]For an introduction to transfer learning with Transformer-based language representation models such as BERT see Wankmüller (2021).

There are two limiting factors when applying BERT: First, due to memory limitations, BERT cannot process text sequences that are longer than 512 tokens (Devlin et al., 2019, p. 4183). This poses no problem for the Twitter corpus which has a maximum sequence length of 73 tokens. In the Reuters news corpus, however, whilst the largest share of articles is shorter than 512 tokens, there is a long tail of longer articles comprising up to around 1,500 tokens.[25] Following the procedure by Sun et al. (2019), Reuters news stories that exceed 512 tokens are reduced to the maximum accepted token length by keeping the first 128 and keeping the last 382 tokens whilst discarding the remaining tokens in the middle.[26] The maximum sequence length recorded for the SBIC is 354 tokens. In order to reduce the required memory capacities, the few posts that are longer than 250 tokens are shortened to 250 tokens by keeping the first 100 and the last 150 tokens.

The second limiting factor is that the prediction performance achieved by BERT after fine-tuning on the target task can vary considerably—even if the same training data set is used for fine-tuning and only the random seeds, that initialize the optimization process and set the order of the training data, differ (Devlin et al., 2019, p. 4176; Phang et al., 2019, p. 5-7; Dodge et al., 2020). Especially when the training data set is small (e.g. smaller than 10,000 or 5,000 documents), fine-tuning with BERT has been observed to yield unstable prediction performances (Devlin et al., 2019, p. 4176; Phang et al., 2019, p. 5-7). Recently, Mosbach et al. (2021) established that the variance in the prediction performance of BERT models, that have been fine-tuned on the same training data set with different seeds, to a large extent is likely due to vanishing gradients in the fine-tuning optimization process. Mosbach et al. (2021, p. 5) also note that it is not that small training data sets per se yield unstable performances, but rather that if small data sets are fine-tuned for the same number of epochs than larger data sets (typically for 3 epochs), then this implies that smaller data sets are fine-tuned for a substantively smaller number of training iterations—which in turn negatively affects the learning rate schedule and the generalization ability (Mosbach et al., 2021, p. 4-5). Finally, Mosbach et al. (2021, p. 2, 8-9) show that fine-tuning with a small learning rate (in the paper: 2e-05), with warmup, bias correction, and a large number of epochs (in the paper: 20) not only tends to increase prediction performances but also significantly decreases the performance instability in fine-tuning. Here, the advice of Mosbach et al. (2021) is followed. For BERT, the AdamW algorithm (Loshchilov & Hutter, 2019) with bias correction, a warmup period lasting 10% of the training steps, and a global learning rate of 2e-05 is used. Training is conducted for 20 epochs. Dropout is set to 0.1. The batch size is set to 16.

For all applications, the pretrained BERT models are taken from Hugging Face's Transformers open source library (Wolf et al., 2020). The BERT model, which here is used as a pretrained input for the English applications based on the SBIC and the Reuters corpus, has been pretrained on the English Wikipedia and the BooksCorpus (Zhu et al., 2015).

[25] There is a single outlier article that is as long as 3,797 tokens.

[26] Note that to meet the input format required by BERT in single sequence text classification tasks, two additional special tokens, *'[CLS]'* and *'[SEP]'*, have to be added (Devlin et al., 2019, p. 4174).

For the data set of German tweets, a German BERT model pretrained on, amongst others, Wikipedia and CommonCrawl data by the digital library team at the Bavarian State Library is used (Münchener Digitalisierungszentrum der Bayerischen Staatsbibliothek (dbmdz), 2021). All BERT models are employed in the base (rather than the large) model version and operate on lowercased (rather than cased) tokens.

For both models, SVM and BERT, an active and a passive supervised learning procedure is implemented. The procedures consist of the following steps: (If the procedures differ between the active and the passive learning setting, it will be explicitly pointed out.)

- The data are randomly separated into 10 (SBIC, Twitter) or 5 (Reuters) equally sized folds.
- Then, for each fold g of the 10 (SBIC, Twitter) or 5 (Reuters) folds the data have been separated into, the following steps are conducted:
    1. Fold g is set aside as a test set.
    2. From the remaining folds, 250 instances are randomly sampled to form the initial set of labeled instances $\mathcal{I}$. The other instances constitute the pool of unlabeled instances $\mathcal{U}$.
    3. The model is trained on the instances in set $\mathcal{I}$ and afterward makes predictions for all instances in pool $\mathcal{U}$ and the set aside test fold g. Recall, precision, and the F_1-Score for the predictions made for pool $\mathcal{U}$ and test fold g are separately recorded. During training in the passive learning setting, random oversampling of the instances falling into the positive relevant class is conducted such that the number of positive relevant instances increases by a factor of 5—thereby reflecting a cost matrix in which the cost of a false negative prediction is set to 5 and the cost of a false positive prediction is set to 1. In the active learning setting, no random oversampling is conducted.
    4. A batch of 50 instances from pool $\mathcal{U}$ is added to the set of labeled instances in set $\mathcal{I}$. In passive learning, these 50 instances are randomly sampled from pool $\mathcal{U}$. In active learning, the following query strategies are applied: In the active learning setting with BERT, the 50 instances whose predicted probability to fall into the positive relevant class is closest to 0.5 are selected. When applying an SVM for active learning, the 50 instances with the smallest perpendicular distance to the hyperplane are retrieved and added to $\mathcal{I}$.
    5. Steps 3 and 4 are repeated for 15 iterations, i.e. until set $\mathcal{I}$ comprises 1,000 labeled instances.

Hence, passive supervised learning with random oversampling and pool-based active learning with uncertainty sampling are applied. As the described learning procedures are repeated for 10 (SBIC, Twitter) or 5 (Reuters) times and are evaluated on each of the 10 (SBIC, Twitter) or 5 (Reuters) folds the data have been separated into, this allows taking

the mean of the F_1-Scores across the 10 (SBIC, Twitter) or 5 (Reuters) test folds as an estimate of the expected generalization error of the applied models.

3.4.3 Results

The results are presented in Figures 3.2 to 3.9 and Tables 3.3 to 3.6.

3.4.3.1 Keyword Lists and Query Expansion

Figure 3.2 visualizes for each of the three studied retrieval tasks (Twitter, SBIC, Reuters) the F_1-Scores resulting from the application of the 100 keyword lists of 10 highly predictive terms as well as the evolution of the F_1-Scores across the query expansion procedure based on locally trained GloVe embeddings (top row) and globally trained GloVe embeddings (bottom row).

In general, the retrieval performances of the initial keyword lists of 10 predictive keywords are mediocre. Only the initial keyword lists for the Reuters corpus achieve what could be called acceptable performance levels. The maximum F_1-Scores reached by the initial lists of 10 predictive keywords are 0.417 (Twitter), 0.404 (SBIC), and 0.645 (Reuters).[27] Moreover, also with the here used empirically driven procedure for the construction of keyword lists, the variation in the initial keyword lists' retrieval performances is considerable. The difference between the maximum and the minimum F_1-Scores is 0.267 (Twitter), 0.270 (SBIC), and 0.381 (Reuters).

Interestingly, the applied query expansion technique tends to decrease rather than increase the F_1-Score and only shows some improvement of the F_1-Score for the Twitter and SBIC data sets—and only if operating on the basis of word embeddings that are trained on large global external corpora rather than the local corpus at hand.

There are several factors that are likely to play a role here. First, when retrieving those terms that have the highest cosine similarity with an initial starting term, the terms retrieved from the global embedding space seem semantically or syntactically related to the initial term, whereas this is not the case for the local word embeddings (as an example see Table 3.3). One reason why the local embedding space does not yield word embeddings that position related terms closely together could be that the three corpora used here are relatively small. The information provided by the context window-based co-occurrence counts of terms thus could be too little for the embeddings to be effectively trained.

Second, in the global embedding space, terms with high cosine similarities seem to be closely related to the initial query term (see again Table 3.3). Adding these related terms

[27] Note that the local lists of 10 predictive keywords have to be used for evaluating the retrieval performance of keyword lists as the global keyword lists have been adapted for the purpose of query expansion on the global word embedding space (see Footnote 17).

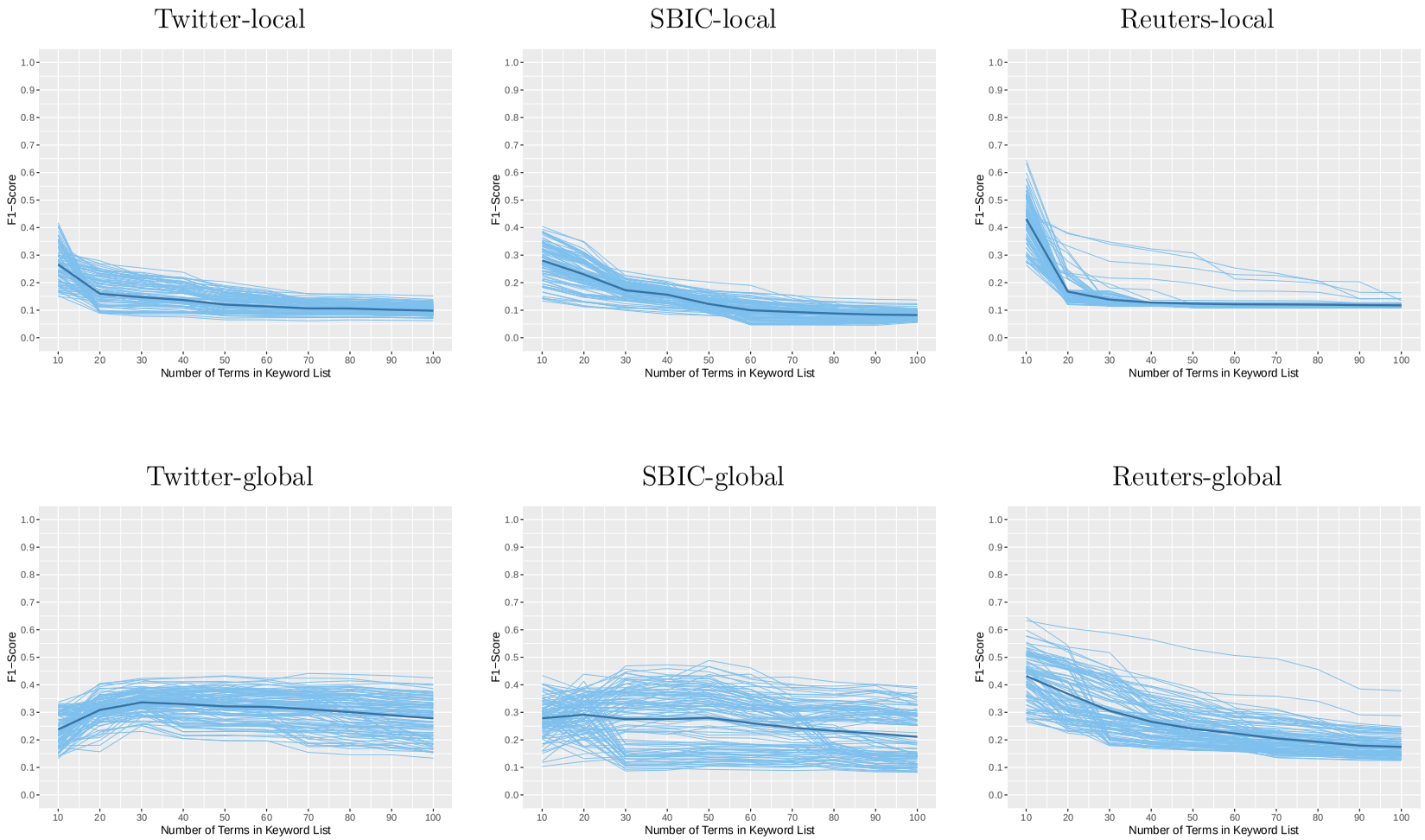


Figure 3.2: **F_1-Scores for Retrieving Relevant Documents with Keyword Lists and Query Expansion.** This plot shows the F_1-Scores resulting from the application of the keyword lists of 10 highly predictive terms as well as the evolution of the F_1-Scores across the query expansion procedure based on locally trained GloVe embeddings (top row) and globally trained GloVe embeddings (bottom row). For each of the sampled 100 keyword lists that then are expanded, one light blue line is plotted. The thick dark blue line gives the mean over the 100 lists.

nevertheless decreases the retrieval performance (as measured by the F_1-Score) for the Reuters corpus. In the case of the Twitter and SBIC corpora, adding these terms with the highest cosine similarities for some iterations at some points (especially at the beginning) slightly increases the F_1-Score, whereas at other points there are decreasing or no visible effects. Here, a second factor comes into play: As is to be expected, query expansion increases recall and decreases precision (see Figures 3.B.2 and 3.B.1 in Appendix 3.B). Hence, in general, query expansion is only worthwhile if—and as long as—the increase in recall outweighs the decrease in precision. Applying the initial set of 10 highly predictive keywords on the Twitter and SBIC data sets, yields a retrieval result that is characterized by low recall and high precision, whereas applying the initial set of 10 highly predictive keywords on the Reuters corpus, leads to very high (sometimes even perfect) recall and low precision (see Figures 3.B.2 and 3.B.1 in Appendix 3.B). Whereas in the second situation, there is no room for query expansion to further improve the retrieval performance via increasing recall (and thus the F_1-Score for the Reuters corpus is moving downward), in the low-recall-high-precision situation of the Twitter and SBIC data sets there is at least the potential for query expansion to increase recall without causing a too strong decrease in precision. This potential is realized in some iterations at some expansion sets, but the decrease in precision more often than not tends to outweigh the increase in recall.

A further reason why query expansion does not perform very well also for global embeddings is the meaning conflation deficiency (Pilehvar & Camacho-Collados, 2020, p. 60): Because word embedding models such as GloVe represent one term by a single embedding vector, a polyseme or homonym is likely to have the various meanings that it refers to encoded within its single representation vector (Neelakantan et al., 2014, p. 1059). The meanings get subsumed into one representation (Schütze, 1998, p. 102). Here, it seems that the conflation of meanings for the GloVe embeddings that have been pretrained on large, global corpora proceeds unequally: The global embedding space tends to position polysemous or homonymous terms close to terms that are semantically or syntactically related to the most common and general meaning of the polysemous or homonymous term (see for example the term *'vegetables'* in Table 3.3). Query expansion in the global embedding space thus fails if an initial query term is a polyseme or homonym and its intended meaning is highly context-specific.

3.4.3.2 Topic Model-Based Classification Rules

Figure 3.3 presents the F_1-Scores reached by topic model-based classification rules. The most notable aspect is that the retrieval performance of topic model-based classification rules is low for the Twitter and SBIC corpora and relatively high for the Reuters corpus. The highest F_1-Score reached in the Twitter retrieval task is 0.253 and regarding the SBIC is 0.175, whereas on the Reuters corpus a score of 0.685 is achieved.

To better understand this result, the terms with the highest occurrence probabilities and the terms with the highest FREX-Score are inspected. The FREX metric is the weighted

Embeddings	Initial term	Terms with the highest cosine similarity to the initial term
local	retard	anymore, blend, float, sex, arguments, college, 93, meanjokes, fever
local	vegetables	100,000, name, U+1f407, knew, combination, traveled, pulled, strip, developed
local	epileptic	oj, blond, include, tactics, crown, tampons, demands, prostitutes, newspapers
global	retard	retards, retarded, dumbass, moron, idiot, faggot, fuckin, stfu, stupid
global	vegetables	veggies, fruits, vegetable, potatoes, carrots, tomatoes, meats, onions, cooked
global	epileptic	seizures, seizure, psychotic, schizophrenic, epileptics, fainting, migraine, spasms, disorder

Table 3.3: **Example SBIC Expansion Terms.** This table gives for each of the highly predictive terms *'retard'*, *'vegetables'*, and *'epileptic'* the nine terms with the highest cosine similarity in the local and global embedding spaces.

harmonic mean of a term's occurrence probability β_{ku} and a term's exclusivity (which is given by $\beta_{ku}/\sum_{j=1}^{K}\beta_{ju}$) (Roberts et al., 2016a, p. 993):

$$FREX_{ku} = \left(\frac{\omega}{ECDF(\beta_{ku}/\sum_{j=1}^{K}\beta_{ju})} + \frac{1-\omega}{ECDF(\beta_{ku})})\right)^{-1} \tag{3.7}$$

where $ECDF$ stands for empirical cumulative distribution function and ω is the weight balancing the two measures. Here ω is set to 0.5.

This inspection (see Tables 3.4, 3.5, and 3.6) reveals that whether and in how far there are exclusive and coherent topics that relate to the entity of interest likely determines whether a topic model-based classification rule can effectively retrieve relevant documents or not.

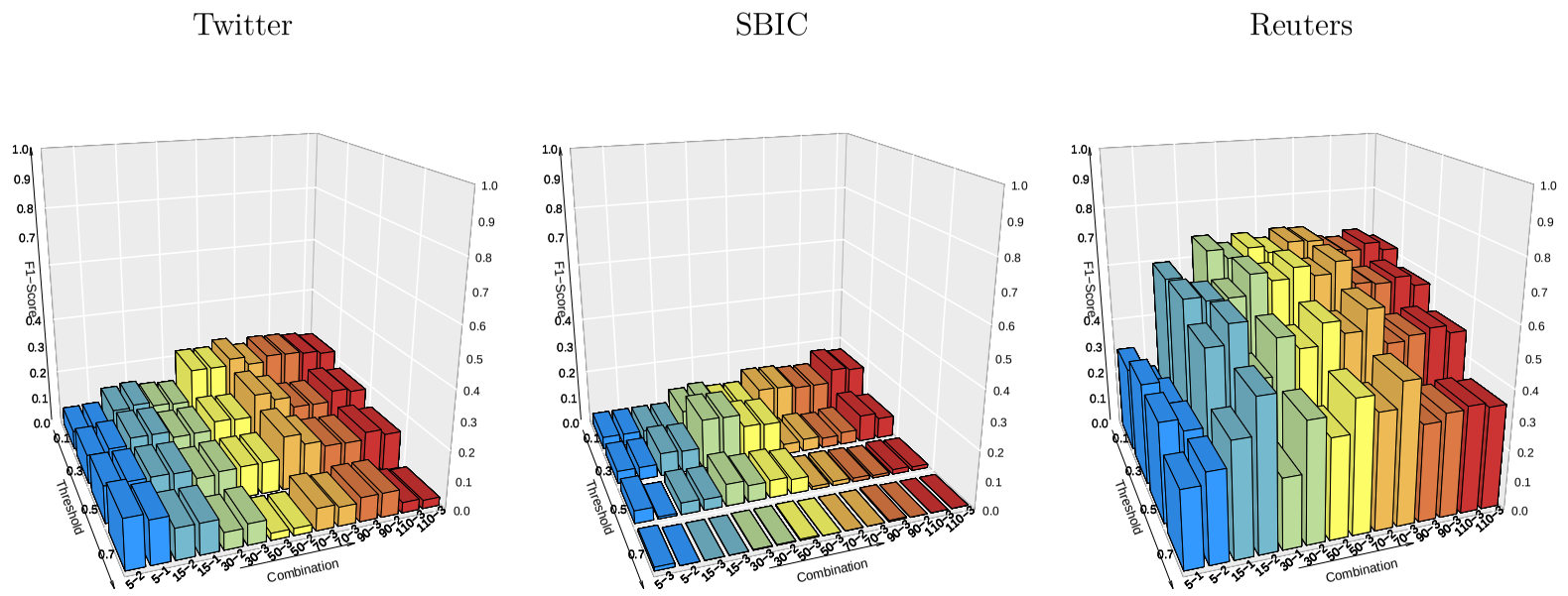


Figure 3.3: **F_1-Scores for Retrieving Relevant Documents with Topic Model-Based Classification Rules.** The height of a bar indicates the F_1-Score resulting from the application of a topic model-based classification rule. For each number of topics $K \in \{5, 15, 30, 50, 70, 90, 110\}$, those two combinations out of all explored combinations regarding the question how many and which topics are considered relevant are shown that reach the highest F_1-Score for the given topic number. The labels of the combinations are such that the first number indicates the number of topics K and the second number denotes the number of topics considered to be relevant by the combination. For example: In the Twitter data set the two best performing combinations for a topic model with $K = 70$ topics both regard three topics as being related to the Twitter topic. Accordingly, the label for both is 70-3. (The two combinations, of course, differ regarding *which* three topics they assume to be relevant.) For each combination, the F_1-Score for each evaluated threshold value $\xi \in \{0.1, 0.3, 0.5, 0.7\}$ is given. Classification rules that assign none of the documents to the positive relevant class have a recall value of 0 and an undefined value for precision and the F_1-Score. Undefined values here are visualized by the value 0.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Prob.	@therealliont	alt	zeit	klein	ne
	facebook	stund	seid	zwei	woch
	lass	los	scheiss	#pegida	nach
FREX	gepostet	fertig	zeit	#pegida	wert
	facebook	cool	#emabiggestfans1d	#nopegida	passt
	monday-giveaway	wahrschein	gonn	setz	woch

	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
Prob.	toll	fall	best	the	eigent
	richtig	nochmal	nacht	of	sonntag
	dabei	h	onlin	hatt	eig
FREX	toll	nochmal	schulmobel	team	eigent
	wenigst	fall	videos	artikel	lag
	manchmal	#breslau	onlin	#techjob	sonntag

	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15
Prob.	wurd	xd	sich	gern	@youtube-playlist
	viel	mensch	suss	sammelt	hinzugefugt
	lohnt	frag	vielleicht	kind	@youtub
FREX	bordelldatenbank.eu	wunderschon	reinhard_4711	sammelt	#younow
	erforscht	frag	mallorcamagazin	zeig	nightcor
	publiziert	geil	sich	glucklich	vs

	Topic 16	Topic 17	Topic 18	Topic 19	Topic 20
Prob.	#iphonegam	#iphon	weiss	lang	oh
	fahrt	steh	welt	veranstalt	schnell
	hotel	gruss	end	event	nix
FREX	antalya	#iphon	kompakt	lkr	mag
	erendiz	#blondin	deintraum	veranstalt	aufgeregt
	sightseeing	#blondinenwitz	haus	event	sing

	Topic 21	Topic 22	Topic 23	Topic 24	Topic 25
Prob.	endlich	bitt	folg	beim	war
	gross	abgeschloss	warum	spass	grad
	brauch	frau	#votesami	seit	sowas
FREX	#immortalis	mission	erfullt	abonni	geschlecht
	immortalis	bitt	belohn	total	mutt
	pvp-gefecht	aufgab	international	herzlich	star

	Topic 26	Topic 27	Topic 28	Topic 29	Topic 30
Prob.	#kca	komm	find	retweet	halt
	twitt	steht	voll	leut	zuruck
	#votedagi	klar	mach	bett	uhr
FREX	gezuchtet	#hamburg	wahr	@ischtaris	anschau
	ratselhaft	geplant	find	bett	zuruck
	#votedagi	hamburg	zeigt	#kca	uberhaupt

Table 3.4: **Twitter: Terms with the Highest Probability and the Highest FREX-Score.** For each topic in the CTM with 30 topics estimated on the Twitter corpus, this table presents the 3 terms with the highest probability (Prob.) and the 3 terms with the highest FREX-Score (FREX). See Topic 4 for the here only moderately coherent Pegida topic. Note that German umlauts here are removed as the preprocessing procedure for the CTM involved stemming—which here also implied removing umlauts.

The entity of interest in the Twitter data set is multi-dimensional. It includes refugees as a social group, refugee policies, and occurrences revolving around the refugee crisis. When examining the most likely and exclusive terms for topic models estimated on the Twitter corpus, it becomes clear that not each aspect of this multi-dimensional refugee topic is captured in a coherent and exclusive topic (see for example Table 3.4). In each model for

$K \geq 30$ there is one relatively coherent topic on Pegida, an anti-Islam and anti-immigration movement that held many demonstrations in the context of the refugee crisis. In addition to this topic, there are further more or less integrated topics that touch refugees and refugee policies without, however, being exclusively about these entities. Thus, the topic models do not offer a set of topics that, taken together, cover all dimensions of the refugee topic in an exclusive manner.

Regarding the SBIC, the situation is even more disadvantageous for the application of topic model-based classification rules. Across all CTMs estimated on the SBIC, there is no topic that identifiably relates to disabled people in a disrespectful way (as an example see Table 3.5). The CTMs with higher topic numbers include some topics that very slightly touch disabilities, but these topics are not coherent. Applying topic model-based classification rules in this situation is futile. Among all evaluated 426,725 $\times$ 4 = 1,706,900 settings, an F_1-Score of 0.175 is as good as it maximally gets.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8
Prob.	U+1F602	go	now	more	sinc	as	time	bitch
	rt	we	right	take	move	year	back	got
	bad	out	left	doe	fire	had	actual	hoe
FREX	U+1F602	go	#releasethememo	mani	season	ago	comput	U+1F612
	U+1F62D	let	now	take	U+1F643	best	hitler	these
	bad	tonight	hillari	wors	move	almost	finish	retard

	Topic 9	Topic 10	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15	Topic 16
Prob.	her	day	by	there	don't	dark	man	guy
	she	today	children	eat	know	movi	into	gay
	make	play	hit	lot	women	an	ask	posit
FREX	girlfriend	april	=	hide	sexist	chees	bar	gay
	her	fool	bottl	eat	women	prostitut	walk	fag
	cri	humid	mosquito	space	don't	humor	into	straight

	Topic 17	Topic 18	Topic 19	Topic 20	Topic 21	Topic 22	Topic 23	Topic 24
Prob.	well	love	hate	your	our	black	muslim	he
	made	shit	who	will	white	call	‘	was
	friend	i'm	r	their	us	between	red	his
FREX	oh	dirti	reason	your	immigr	pizza	‘	kid
	god	love	crime	peac	russia	black	rose	dad
	^	yo	asshol	educ	nation	common	ice	father

	Topic 25	Topic 26	Topic 27	Topic 28	Topic 29	Topic 30
Prob.	look	tri	off	see	>	would
	good	start	im	here	<	one
	incel	stori	done	post	s	can
FREX	hair	case	piss	post	>	would
	look	ethiopia	youtub	see	<	never
	normal	touch	im	pictur	number	one

Table 3.5: **SBIC: Terms with the Highest Probability and the Highest FREX-Score.** For each topic in the CTM with 30 topics estimated on the SBIC, this table presents the 3 terms with the highest probability (Prob.) and the 3 terms with the highest FREX-Score (FREX). See Topic 8 for a non-coherent and non-exclusive topic that slightly touches disrespectful posts about disabled people.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8
Prob.	cts	profit	pct	share	export	bank	ec	unit
	avg	oper	growth	pct	nil	rate	european	subsidiari
	shrs	gain	rise	stock	coffe	pct	communiti	agreement
FREX	shrs	profit	gnp	smc	seamen	9-13	ec	mhi
	avg	pretax	economi	ucpb	prev	bank	communiti	cetus
	cts	extraordinari	growth	calmat	ibc	9-7	ecus	squibb

	Topic 9	Topic 10	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15	Topic 16
Prob.	pct	vs	trade	franc	gulf	oil	offer	dollar
	billion	loss	u.	french	ship	price	share	rate
	januari	rev	japan	group	u.	gas	sharehold	currenc
FREX	unadjust	rev	lyng	ferruzzi	missil	opec	caesar	louvr
	januari	vs	chip	cgct	warship	herrington	sosnoff	miyazawa
	fell	mths	miti	cpc	tehran	bpd	cyacq	poehl

	Topic 17	Topic 18	Topic 19	Topic 20	Topic 21	Topic 22	Topic 23	Topic 24
Prob.	share	billion	tonn	ltd	price	food	analyst	canadian
	stock	trade	wheat	plc	contract	beef	earn	canada
	dividend	reserv	export	pct	cent	philippin	market	credit
FREX	dividend	fed	beet	csr	octan	satur	analyst	card
	payabl	surplus	cane	transcanada	kwacha	diseas	ibm	canadian
	payout	taiwan	rapese	monier	sulphur	nppc	rumor	nova

	Topic 25	Topic 26	Topic 27	Topic 28	Topic 29	Topic 30
Prob.	quarter	price	chemic	court	gold	debt
	first	produc	u.	file	mine	payment
	earn	stock	busi	general	ton	loan
FREX	fourth	cocoa	gaf	gencorp	assay	debt
	quarter	buffer	hanson	afg	uranium	payment
	earn	icco	borg-warn	court	gold	repay

Table 3.6: **Reuters: Terms with the Highest Probability and the Highest FREX-Score.** For each topic in the CTM with 30 topics estimated on the Reuters corpus, this table presents the 3 terms with the highest probability (Prob.) and the 3 terms with the highest FREX-Score (FREX). See Topic 14 for the crude oil topic and see Topic 13 for the military topic that, at times, touches crude oil.

The situation is entirely different for the crude oil topic. For $K \geq 30$ each estimated topic model contains at least one coherent topic that clearly refers to aspects of crude oil (e.g. *'opec'*, *'bpd'*, *'oil'*; see Table 3.6). These coherent crude oil topics are not completely but relatively exclusive. Some of the crude oil topics also cover another energy source (namely: *'gas'*) and there is one reappearing conflict topic that refers to military aspects but also touches crude oil (*'gulf'*, *'missil'*, *'warship'*, *'oil'*). Other than that, no other entities are substantially covered by crude oil topics. Building topic model-based classification rules on the basis of these crude oil topics yields relatively high recall and precision values.

Hence, topic model-based classification rules can be a useful tool—but only if the estimated topics coherently and exclusively cover the entity of interest in all its aspects.

In all three applications, and as is to be expected, high recall and low precision values tend to be achieved for topic models with smaller number of topics and lower values for threshold ξ, whereas low recall and high precision values tend to result from topic models

with a higher number of topics and higher values for ξ (see Figure 3.C.1 in Appendix 3.C).[28] Classification rules that use topic models with a higher topic number and lower threshold ξ neither tend to exhibit the highest recall nor the highest precision values, but they tend to strike the best balance between recall and precision and achieve the highest F_1-Scores (see Figure 3.3 here and Figure 3.C.1 in Appendix 3.C).

3.4.3.3 Active and Passive Supervised Learning

For each of the three studied retrieval tasks (Twitter, SBIC, Reuters), for each employed supervised learning model (BERT and SVM), for each applied learning setting (passive learning with random oversampling as well as pool-based active learning with uncertainty sampling), and for each of the 10 (Twitter, SBIC) or 5 (Reuters) conducted iterations, Figure 3.4 depicts the F_1-Score achieved on the set aside test set (fold g) as the number of labeled training documents in set $\mathcal{I}$ increases from 250 to 1,000. Passive supervised learning results are visualized by blue lines, active learning results are given in red. The thick and dark lines give the mean F_1-Scores across the iterations. They visualize the estimate of the expected generalization error.[29]

Across all three applications and for BERT as well as SVM, active learning with uncertainty sampling tends to dominate passive learning with random oversampling. Passive learning with random oversampling on average only shows a similar or higher F_1-Score for the first learning iteration (i.e. at the start when training is conducted on the randomly sampled training set of 250 labeled instances). Then, however, the active learning retrieval performance strongly increases such that for the same number of labeled training instances active learning, on average, produces a higher F_1-Score than passive learning.

One likely reason for this difference between passive and active learning is revealed in Figure 3.5 that contains the same information as Figure 3.4, except that on the y-axis not the F_1-Score but the share of documents from the positive relevant class in the training set is shown. The black dashed line visualizes the share of relevant documents in the entire corpus and thus would be the expected share of relevant documents in a randomly sampled training set if neither random oversampling nor active learning were conducted. In passive learning with random oversampling (shown in blue) the 50 training instances, that are added in each step to the set of labeled training instances $\mathcal{I}$, are randomly sampled from pool $\mathcal{U}$. Then, the relevant instances in set $\mathcal{I}$ are randomly oversampled by a factor of 5.

[28] A too high ξ, however, at times may lead to a classification rule in which none of the documents is assigned to the positive relevant class—thereby producing an undefined value for precision and the F_1-Score (here visualized by 0).

[29] Note that the x-axis denotes the number of unique labeled instances in set $\mathcal{I}$. As in passive learning with random oversampling the documents from the relevant minority class in set $\mathcal{I}$ are randomly resampled with replacement to then form the training set on which the model is trained, in passive learning the size of the training set is larger than the size of set $\mathcal{I}$. Yet the size of $\mathcal{I}$ indicates the number of unique documents on which training is performed and—as only unique documents have to be annotated—it indicates the annotation costs.

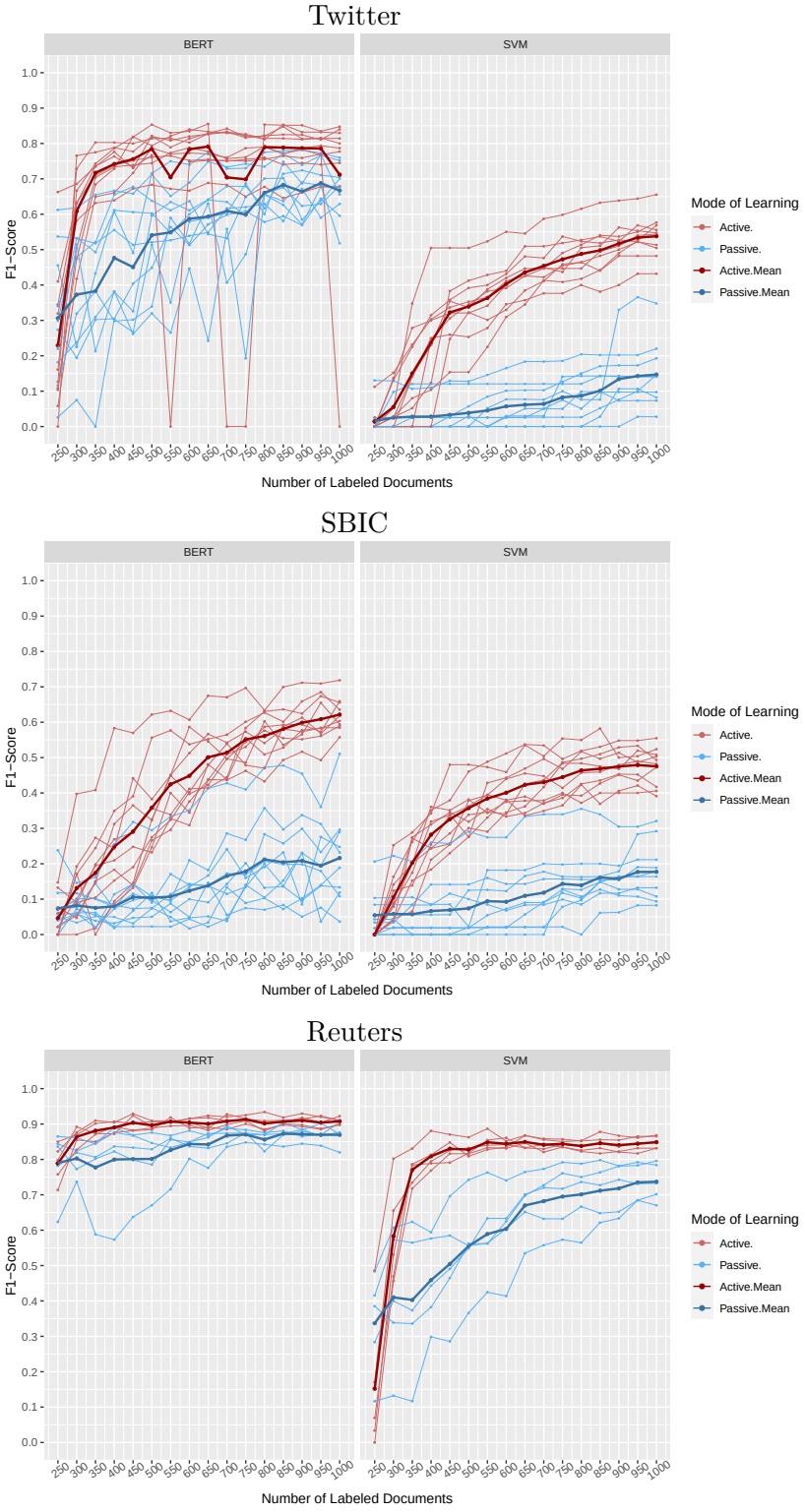


Figure 3.4: **Retrieving Relevant Documents with Active and Passive Supervised Learning.** F_1-Scores achieved on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. Passive supervised learning results are visualized by blue lines, active learning results are given in red. For each of the 10 (Twitter, SBIC) or 5 (Reuters) conducted iterations, one light colored line is plotted. The thick and dark lines give the means across the iterations. If a trained model assigns none of the documents to the positive relevant class, then it has a recall value of 0 and an undefined value for precision and the F_1-Score. Undefined values here are visualized by the value 0.

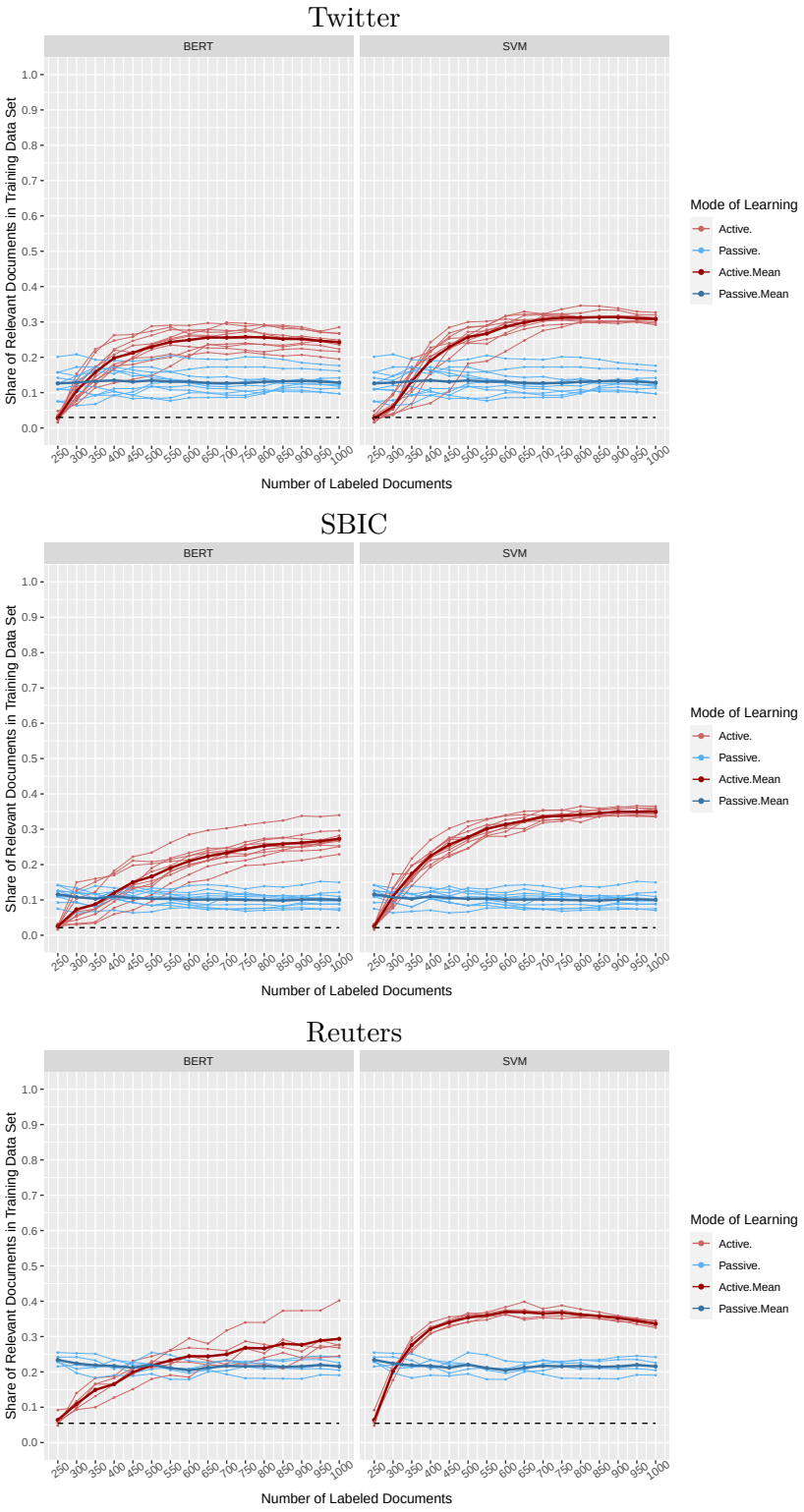


Figure 3.5: **Share of Relevant Documents in the Training Set.** Share of documents in the training set that fall into the positive relevant class as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. Passive supervised learning shares are visualized by blue lines, active learning shares are given in red. For each of the 10 (Twitter, SBIC) or 5 (Reuters) conducted iterations, one light colored line is plotted. The thick and dark lines give the means across the iterations. The black dashed line visualizes the share of relevant documents in the entire corpus.

For this reason, the share of positive training instances in the passive learning setting is higher than in the corpus (black dashed line) but remains relatively constant across the training steps. In active learning (shown in red), no random oversampling is conducted—which is why at the beginning the share of relevant documents at about equals the share of relevant documents in the corpus. Then, however, active learning at each step selects the 50 instances the algorithm is most uncertain about. As has been observed in other studies before (Ertekin et al., 2007, p. 133-134; Miller et al., 2020, p. 545), this implies that disproportionately many instances from the relevant minority class are selected into set $\mathcal{I}$. The share of positive training instances increases substantively—which in turn tends to increase generalization performance on the test set as shown in Figure 3.4.

When decomposing the F_1-Score into recall and precision (see Figure 3.D.1 in Appendix 3.D), it is revealed that the supervised models' recall values gradually improve as the number of training instances increases. The precision values early reach higher levels and exhibit a more volatile path. The observed retrieval performance enhancements hence seem to be caused by the fact that the models from step to step are becoming better at identifying a larger share of the truly relevant documents from the corpora. Models trained in active rather than passive learning mode tend to yield higher recall values.

Yet there is the question of whether active learning exhibits superior performance to passive learning with random oversampling simply because after a certain number of training steps the share of training instances is higher for active than for passive learning or whether active learning dominates passive learning (also) because active learning, due to focusing on the uncertain region between the classes and due to operating on unique—rather than duplicated—positive training instances, learns a better generalizing class boundary with fewer training instances (Settles, 2010, p. 28). To inspect this question, for the SVMs, passive learning with random oversampling is repeated and positive relevant documents are randomly oversampled such that their number increases by a factor of 10 (Reuters), 17 (Twitter), or 20 (SBIC) (instead of by a factor of 5 as before). This results in higher shares of relevant documents in the training set for passive learning (see the right column in Figure 3.6). However, the prediction performance on the test set as measured by the F_1-Score either does not or does only minimally increase compared to the situation of random oversampling by a factor of 5 (see the left column in Figure 3.6). Moreover, although the share of relevant documents in the stronger oversampled passive learning training data sets is similar to those of active learning, active learning still yields considerably higher F_1-Scores. This indicates that from a certain point, merely duplicating positive instances by random oversampling has no or only a small effect on the class boundary learned by the SVM. The finding also indicates that active learning improves upon passive learning because it is effectively able to select a large share of truly positive documents for training that are not duplicated but unique and because its selection of uncertain documents provides crucial information on the class boundary.

When applying SVMs to imbalanced data sets, the problem, in general, is that the learned hyperplane tends to be positioned too close to the positive minority class instances (Akbani

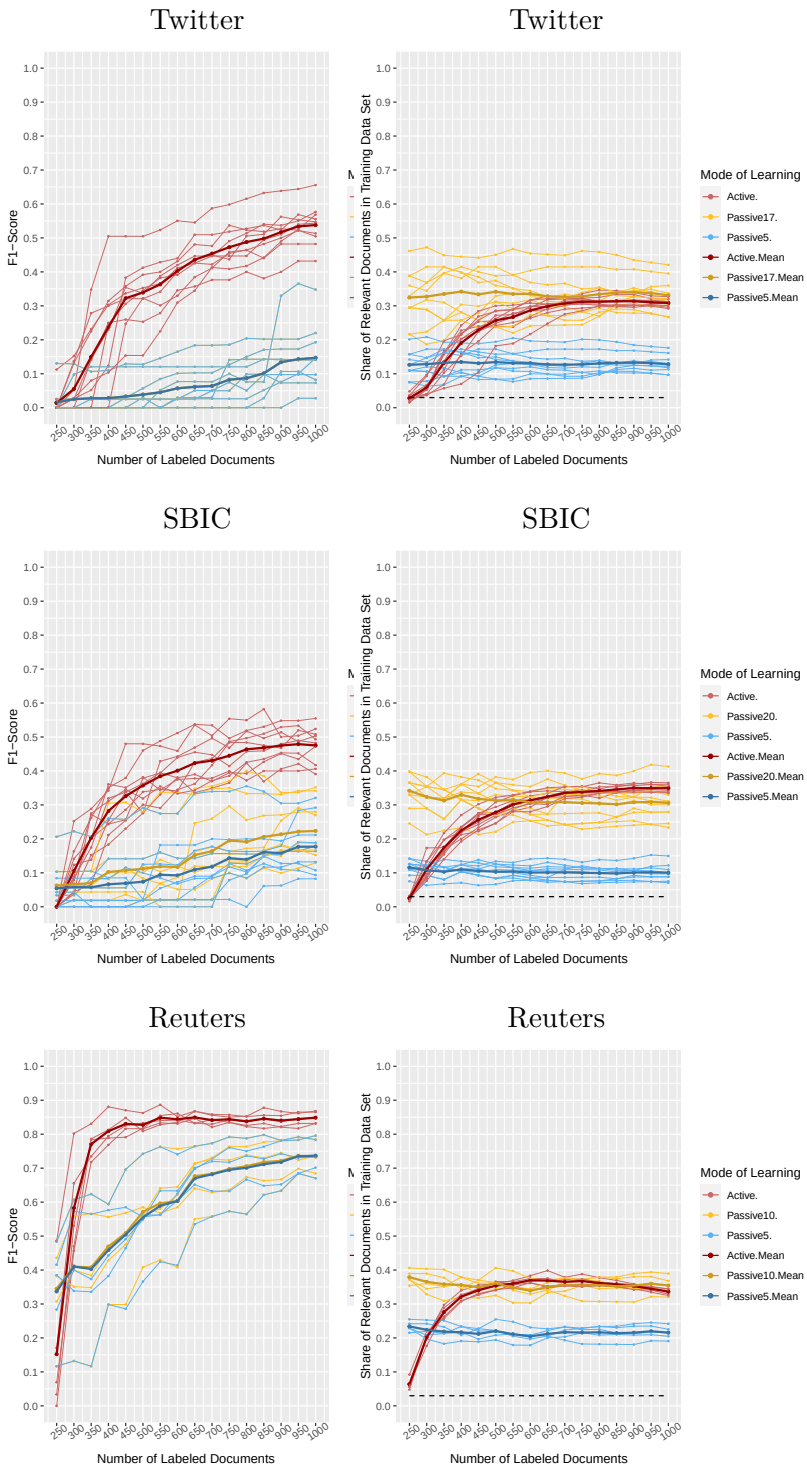


Figure 3.6: **SVM: Comparing Passive Learning with Two Oversampling Factors to Active Learning. Left column:** F_1-Scores achieved by the SVMs on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. Undefined values here are visualized by the value 0. **Right column:** Share of documents in the training set that falls into the positive relevant class as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. The black dashed line visualizes the share of relevant documents in the entire corpus. **Both columns:** The results are presented for passive learning with a random oversampling factor of 5 (blue lines), passive learning with random oversampling factors of 17 (Twitter), 20 (SBIC), and 10 (Reuters) (golden lines), as well as pool-based active learning with uncertainty sampling (red lines). The thick and dark lines give the means across the iterations.

et al., 2004, p. 40-44). The reason is that whereas the many negative training instances occur across the entire area *belonging* to the negative class, the few positive instances only occur at a few points within the area *belonging* to the positive class (Akbani et al., 2004, p. 40-44). Hence, the boundary of the negative area is well represented in the training data whereas the boundary of the positive area is not (for an illustration see Akbani et al., 2004, p. 43-44). The hyperplane can be moved toward the negative side by giving more weight to positive instances, e.g. by duplicating them via random oversampling or by introducing different costs for misclassifying positive vs. negative instances (Veropoulos et al., 1999). A problem that often arises when doing so, however, is that the hyperplane tends to overfit on the positive instances. Its orientation and shape too strongly tends to reflect the positions of positive instances (Akbani et al., 2004, p. 46). In active learning, in contrast, the training instances the learning algorithm requests to be labeled and added next are unique—which has a positive effect on generalization performance.

Another important observation concerns the performances' variability (see again Figure 3.4): For all models and learning modes, given a fixed number of labeled training instances, the F_1-Scores on the set aside test sets can vary considerably between iterations. Which set of documents is randomly sampled to form the (initial) training set and which set aside test fold is used for evaluation thus can have a profound effect on the measured retrieval performance.

A further observation is that BERT on average tends to outperform SVM (see also Figure 3.E.1 in Appendix 3.E). Hence, applying the Transformer-based pretrained language representation model BERT with its ability to learn context-dependent meanings of tokens and with the information acquired in the pretraining phase, here tends to be better able to identify the few relevant documents, than an SVM operating on bag-of-words representations. That BERT only can process sequences of a maximum of 512 tokens is not a particularly limiting or performance reducing factor here. The performance difference between the two learning methods is distinct and relatively consistent with regard to the Twitter and Reuters retrieval tasks. It is less clear-cut for the SBIC. This is rather surprising as in the SBIC the disrespectful remarks toward disabled people are implied rather than stated explicitly.

Note also that, with regard to the Twitter and SBIC retrieval tasks, BERT exhibits a high level of instability from one learning step to the next as the number of labeled training instances $\mathcal{I}$ increases by a batch of 50 documents (see Figure 3.E.2 in Appendix 3.E). After adding a new batch of 50 labeled training instances and fine-tuning BERT on this new, slightly expanded training data set, the F_1-Score achieved by BERT on the test set may not only increase but also decrease considerably. The strongest de- and increases can be observed for active learning on the Twitter data set where drops and rises of the F_1-Score by a value of about 0.85 occur.

As noted in Section 3.4.2.4 above, BERT's prediction performance can exhibit considerable variance across random initializations—even if trained on the exact same training data set. Here, to make predictions and performances more stable, precautions have been

taken by choosing a small learning rate of 2e-05 and setting the number of epochs to 20 such that many training iterations are conducted. To evaluate how the level and the stability of BERT's prediction performance changes if the hyperparameters are set to more conventional values (e.g. combining a learning rate of 2e-05 or 3e-05 with 3 or 4 epochs), BERT is also trained using hyperparameter values in these common ranges. Figure 3.F.1 in Appendix 3.F visualizes the F_1-Scores of these models. The instability of the learning paths is only slightly to moderately higher (see Figure 3.F.2 in Appendix 3.F). Yet for the Twitter and SBIC tasks and especially as the training data sets are very small, the F_1-Scores of BERT models with common hyperparameter values are substantively lower than the F_1-Scores of BERT models that have been trained for 20 epochs. In the Twitter application, for example, the mean F_1-Score of active learning with a BERT model that is trained for 20 epochs on a set of 500 labeled training instances is 0.568 higher than active learning with a BERT model trained for 3 epochs on 500 training instances. Hence, although training over 20 epochs takes proportionally more computing time than training over 3 or 4 epochs,[30] when applying BERT to small data sets—a scenario which is likely in the retrieval settings focused on here—training for many epochs (and thus presenting each document in the small training data set many times to the model) seems important to enhance performance.

Nevertheless, because BERT tends to exhibit a relatively high degree of variability in its performance even if training is conducted for a larger number of epochs, monitoring retrieval performance with a set aside test set seems important for researchers to detect situations in which (likely due to vanishing gradients) retrieval performance drops to low values. Such situations can be easily fixed by, for example, choosing another random seed for initialization.

To conclude, if a team of researchers has the resources to retrieve documents referring to their entity of interest via supervised learning and they have a fixed number of training instances they can maximally label, then active learning is likely to yield better results than passive learning. Moreover, if none or only a small share of documents exceeds the maximum number of tokens that Transformer-based language representation models as BERT can process, then applying a BERT-like model that is trained with a small learning rate for a large number of epochs is likely to achieve better results than applying conventional supervised machine learning methods (such as SVM) on bag-of-words-based representations.[31] But the predictions made by BERT (and hence also BERT's retrieval performance) is prone to considerable variation depending on the initializing random seed, the initial training data set, and the changes to the training data set from one learning step to the next. This problem, however, is mitigated by the fact that mediocre performances can be easily detected if performance is monitored with a set aside test set.

[30] For example, training BERT for 20 epochs on 1,000 tweets takes on average 172 seconds, whereas training BERT on 1,000 tweets for 3 epochs takes on average 26 seconds.

[31] If a large share of documents in the corpus at hand are longer than the 512 tokens that can be processed by BERT, Transformer-based models that can process longer sequences of tokens, e.g. the Longformer (Beltagy et al., 2020), can be applied.

3.4.3.4 Comparison Across Approaches

The central question this study seeks to answer is as follows: What, if anything, can be gained by applying more costly retrieval approaches such as query expansion, topic model-based classification rules, or supervised learning instead of the relatively simple and inexpensive usage of a Boolean query with a keyword list? In order to finally answer this question and compare the approaches against each other, Figures 3.7, 3.8, and 3.9 summarize the retrieval performance—as measured by the F_1-Score—of the evaluated approaches on the retrieval tasks associated with the Twitter corpus (Figure 3.7), the SBIC (Figure 3.8), and the Reuters-21578 corpus (Figure 3.9). In each figure, the left panel gives the F_1-Scores for the lists of 10 predictive keywords that then are expanded in the local and global embedding spaces. The middle panel shows the F_1-Scores of topic model-based classification rules with different values for threshold ξ. The right panel visualizes the F_1-Scores for active as well as passive supervised learning with SVM and BERT.

In general, the direct comparison shows that—when taking keyword lists comprising 10 empirically predictive terms as the baseline—the application of more complex and more expensive retrieval techniques does not guarantee better retrieval results.

Query expansion techniques here rather decrease than increase the F_1-Score. Minimal improvements only occur sporadically in the embedding space trained on external global corpora if the increase in recall outweighs the decrease in precision. The farther the expansion, the worse the results tend to become.

In general, it seems that the identification of newspaper articles referring to crude oil from the Reuters corpus is a more simple task than the retrieval of tweets relating to the multidimensional refugee topic or the extraction of posts that refer to an entity (disabled people) and are of a particular kind (here: disrespectful). All approaches show higher F_1-Scores on the Reuters corpus, and lower scores for the other two evaluated retrieval tasks.

Topic model-based classification rules work relatively well for the Reuters corpus but not the other corpora. Hence, if there are no coherent and exclusive topics that cover the entity of interest in all its aspects, topic model-based classification rules exhibit rather poor retrieval performances. In the Twitter and SBIC data sets, the F_1-Score reached by the topic model-based classification rules are in the lower range of the values achieved by the lists of predictive keywords. If, on the other hand, coherent and exclusive topics relating to the entity of interest exist (as is the case for the Reuters corpus), acceptable retrieval results are possible. Here, gains over the best performing keyword lists are achieved for combinations with larger topic numbers and smaller values for threshold ξ. The best-performing topic model-based classification rule on the Reuters corpus is based on a CTM with 70 topics that considers two topics to be relevant and predicts documents to be relevant that have 10% of their words assigned to these two relevant topics. This topic model-based classification rule reaches an F_1-Score of 0.685, which is 0.04 higher than the F_1-Score of the best performing keyword list.

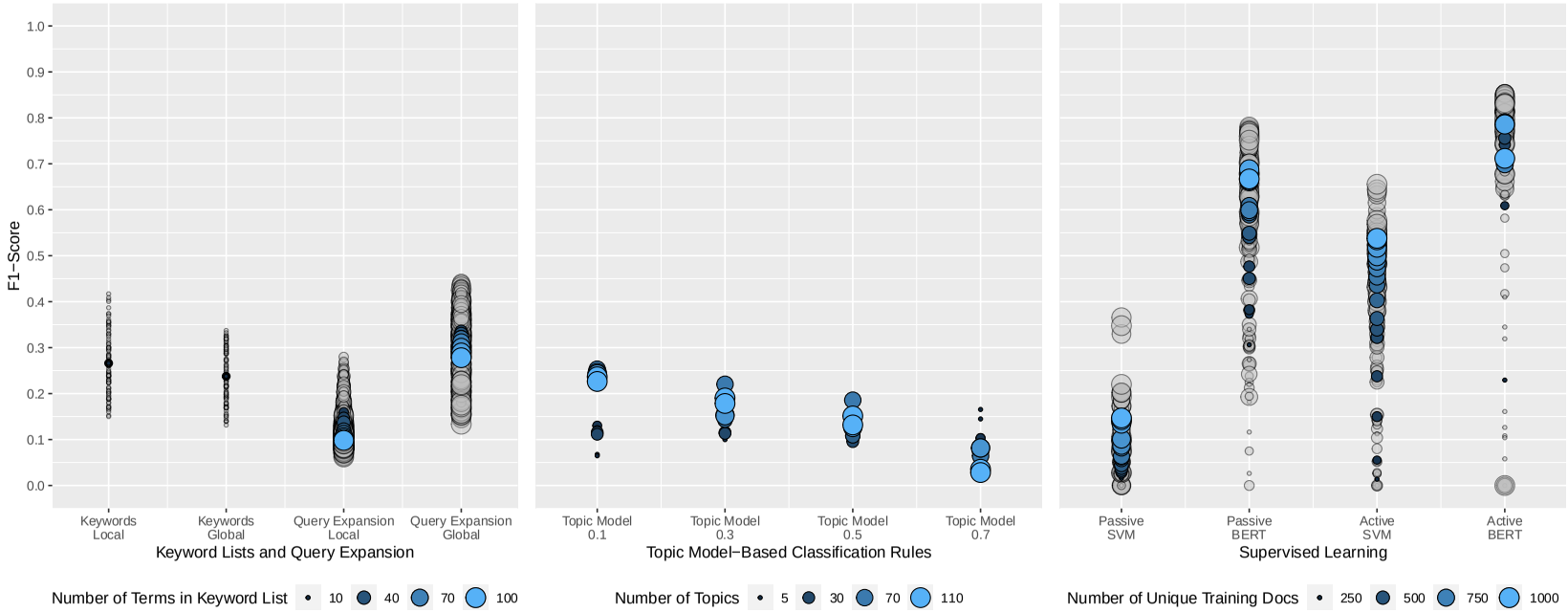


Figure 3.7: **Twitter: Comparison of Retrieval Methods.** This plot summarizes the retrieval performances of the here evaluated approaches on the retrieval task associated with the Twitter corpus. **Left panel:** F_1-Scores for the 100 lists of 10 predictive keywords that then are expanded in the local and global embedding spaces. A gray transparent dot marks the F_1-Score reached by a single (expanded) keyword list. A blue opaque dot marks the mean of the F_1-Scores across 100 (expanded) keyword lists that contain the same number of search terms. The larger the number of terms in an (expanded) keyword list, the larger the size and the lighter the color of the printed dots. **Panel at the center:** F_1-Scores of the topic model-based classification rules with different values for threshold ξ. The larger the number of topics in an estimated CTM, the larger the size and the lighter the color of the printed dots. (From all CTMs with a given topic number that have been estimated here, the best two performing combinations regarding the question of how many and which topics are considered relevant are presented.) **Right panel:** F_1-Scores for active as well as passive supervised learning with SVM and BERT. A gray transparent dot marks the F_1-Score reached by a model that has been trained by a given number of unique training documents and is evaluated on one set aside test set g. A blue opaque dot marks the mean of the F_1-Scores of 10 such models that have been trained on the same number of training documents. Each of the 10 models is evaluated on another test set and together the 10 test sets constitute the entire Twitter corpus. The larger the number of unique labeled training instances, the larger the size and the lighter the color of the printed dots.

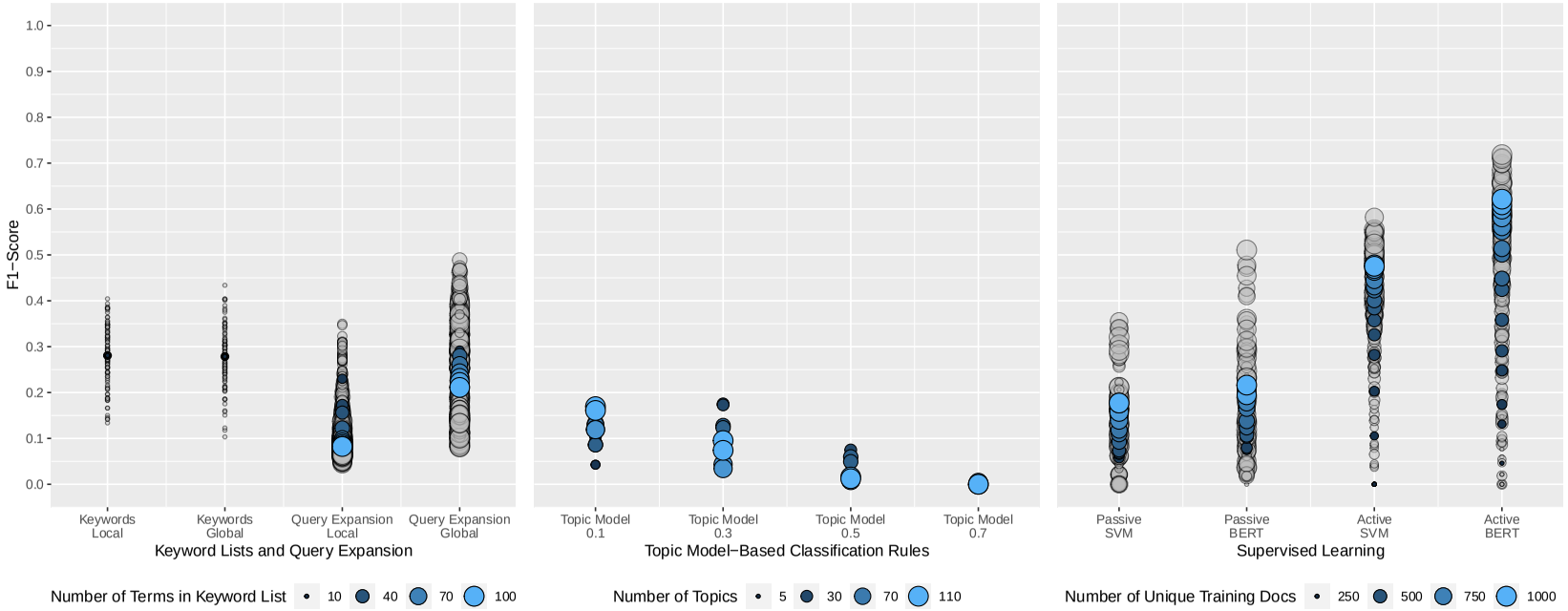


Figure 3.8: **SBIC: Comparison of Retrieval Methods.** This plot summarizes the retrieval performances of the here evaluated approaches on the retrieval task associated with the SBIC. **Left panel:** F_1-Scores for the 100 lists of 10 predictive keywords that then are expanded in the local and global embedding spaces. A gray transparent dot marks the F_1-Score reached by a single (expanded) keyword list. A blue opaque dot marks the mean of the F_1-Scores across 100 (expanded) keyword lists that contain the same number of search terms. The larger the number of terms in an (expanded) keyword list, the larger the size and the lighter the color of the printed dots. **Panel at the center:** F_1-Scores of the topic model-based classification rules with different values for threshold ξ. The larger the number of topics in an estimated CTM, the larger the size and the lighter the color of the printed dots. (From all CTMs with a given topic number that have been estimated here, the best two performing combinations regarding the question of how many and which topics are considered relevant are presented.) **Right panel:** F_1-Scores for active as well as passive supervised learning with SVM and BERT. A gray transparent dot marks the F_1-Score reached by a model that has been trained by a given number of unique training documents and is evaluated on one set aside test set g. A blue opaque dot marks the mean of the F_1-Scores of 10 such models that have been trained on the same number of training documents. Each of the 10 models is evaluated on another test set and together the 10 test sets constitute the entire SBIC. The larger the number of unique labeled training instances, the larger the size and the lighter the color of the printed dots.

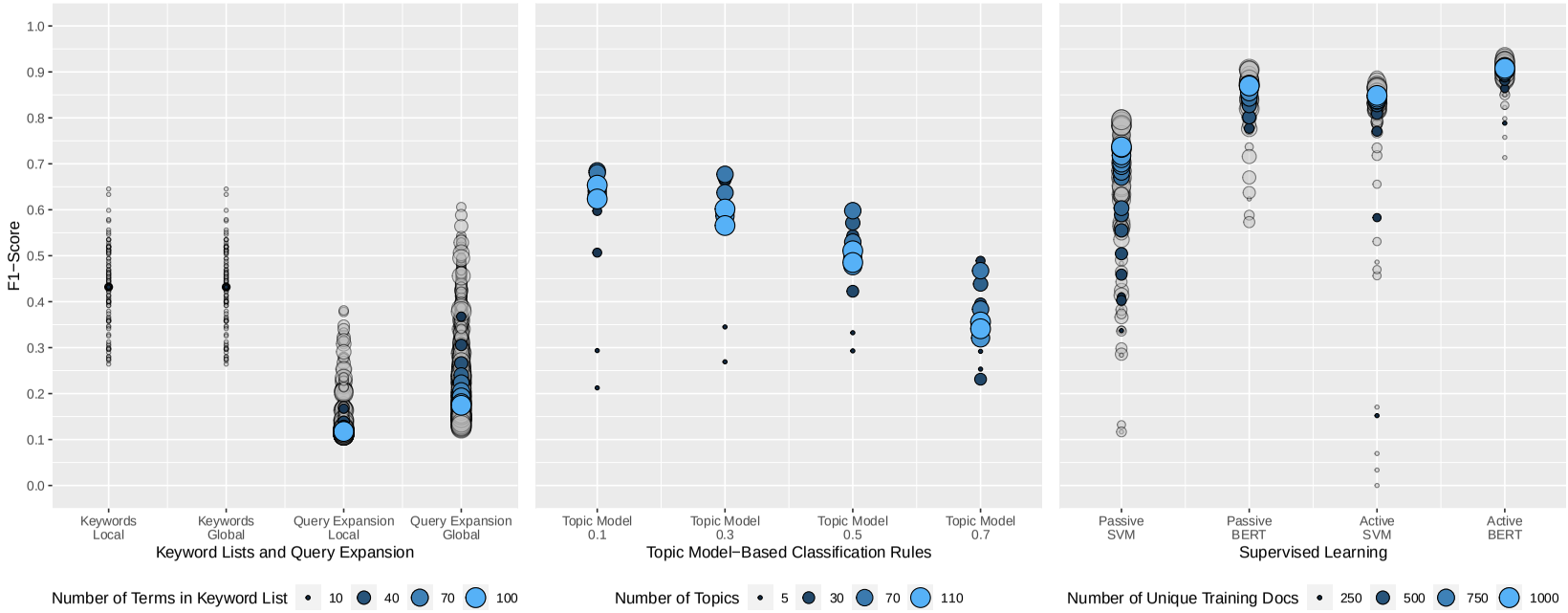


Figure 3.9: **Reuters: Comparison of Retrieval Methods.** This plot summarizes the retrieval performances of the here evaluated approaches on the retrieval task associated with the Reuters-21578 corpus. **Left panel:** F_1-Scores for the 100 lists of 10 predictive keywords that then are expanded in the local and global embedding spaces. A gray transparent dot marks the F_1-Score reached by a single (expanded) keyword list. A blue opaque dot marks the mean of the F_1-Scores across 100 (expanded) keyword lists that contain the same number of search terms. The larger the number of terms in an (expanded) keyword list, the larger the size and the lighter the color of the printed dots. **Panel at the center:** F_1-Scores of the topic model-based classification rules with different values for threshold ξ. The larger the number of topics in an estimated CTM, the larger the size and the lighter the color of the printed dots. (From all CTMs with a given topic number that have been estimated here, the best two performing combinations regarding the question of how many and which topics are considered relevant are presented.) **Right panel:** F_1-Scores for active as well as passive supervised learning with SVM and BERT. A gray transparent dot marks the F_1-Score reached by a model that has been trained by a given number of unique training documents and is evaluated on one set aside test set g. A blue opaque dot marks the mean of the F_1-Scores of 5 such models that have been trained on the same number of training documents. Each of the 5 models is evaluated on another test set and together the 5 test sets constitute the entire Reuters-21578 corpus. The larger the number of unique labeled training instances, the larger the size and the lighter the color of the printed dots.

Whereas query expansion techniques and topic model-based classification rules show no or small improvements, supervised learning—especially if conducted in an active learning mode—has the potential to yield a substantively higher retrieval performance than a list of 10 predictive keywords. The prerequisite for this, however, is that not too few training instances are used. The larger the number of training instances, the higher the F_1-Score tends to be. Yet, as has been established above, especially for BERT this relationship is not monotonic and can exhibit considerable variability. What number of training documents is required to produce acceptable retrieval results that are better than what could be achieved with a keyword list, depends on the specifics of the retrieval task at hand and the employed learning mode and model. In the Twitter application, for example, it is likely to achieve an acceptable to good retrieval performance that improves upon keyword lists when applying active learning with BERT using ≥ 350 training documents or applying passive learning with BERT on ≥ 800 unique training documents (see again Figure 3.4).

In general, active learning is to be preferred over passive learning, as across applications and learning models, active learning tends to reach a higher retrieval performance than passive learning with the same number of training documents. Moreover, the pretrained deep neural network BERT tends to yield a higher F_1-Score compared to an SVM operating on bag-of-words-based document representations. BERT produces more unstable results, but this behavior can be monitored with a set aside test set.

Across applications, applying active learning with BERT until 1,000 training instances have been labeled here produces a good separation of relevant and irrelevant documents that considerably improves upon the separation achieved by applying a keyword list. The mean F_1-Scores of BERT applied in an active learning mode with a training budget of 1,000 labeled instances are 0.712 (Twitter), 0.622 (SBIC), and 0.908 (Reuters), whereas the *maximum* F_1-Scores reached by the empirically constructed initial keyword lists are 0.417 (Twitter), 0.404 (SBIC), and 0.645 (Reuters). Hence, the improvements in the F_1-Scores that are achieved by applying active learning with BERT rather than the best performing keyword list are 0.295 (Twitter), 0.218 (SBIC), and 0.263 (Reuters). Active learning with a conventional machine learning model as an SVM still is a good and viable alternative. The mean F_1-Scores of active learning with SVM trained with 1,000 labeled documents are 0.538 (Twitter), 0.475 (SBIC), and 0.849 (Reuters). This still yields enhancements of the F_1-Score by 0.121 (Twitter), 0.071 (SBIC), and 0.204 (Reuters).

Note that the performance enhancements of active learning here are observed across applications. Irrespective of document length, textual style, the type of the entity of interest, and the homogeneity or heterogeneity of the corpus from which the documents are retrieved, active learning with 1,000 training documents shows superior performance to keyword lists and the other approaches.

3.5 Conclusion

In text-based analyses, researchers are typically interested to study documents referring to a particular entity. Yet textual references to specific entities are often contained within multi-thematic corpora. In consequence, documents that contain references to the entities of interest have to be separated from those that do not.

A very common approach in social science to retrieve relevant documents is to apply a list of keywords. Keyword lists are inexpensive and easy to apply, but they can result in biased inferences if they systematically fail to identify relevant documents. Query expansion techniques, topic model-based classification rules, and active as well as passive supervised learning constitute alternative, more expensive, more complex, and in social science rarely applied procedures for the retrieval of relevant documents. These more complex procedures theoretically have the potential to reach a higher retrieval performance than keyword lists and thus could reduce the potential size of selection biases. So far, a systematic comparison of these approaches was lacking and therefore it was unclear, whether the employment of any of these more expensive methods would yield any improvements of retrieval performance, and, if they did, how large and consistent across contexts the improvement would be.

This study closed this gap. The comparison of the approaches on the basis of retrieval tasks associated with a data set of German tweets (Linder, 2017), the Social Bias Inference Corpus (SBIC) (Sap et al., 2020), and the Reuters-21578 corpus (Lewis, 1997) shows that neither of the applied more complex approaches necessarily enhances the retrieval performance, as measured by the F_1-Score, over the application of a keyword list containing 10 empirically predictive terms. However, whereas across all settings and combinations evaluated for query expansion techniques and topic model-based classification rules at the very best small increases in the F_1-Score can be observed, active supervised learning with the Transformer-based language representation model BERT increases the F_1-Scores across application contexts substantively if the number of labeled training documents used in the active learning process is not too small.

Thus, in terms of retrieval performance, supervised learning in an active learning mode—preferably with a pretrained deep neural network—is the procedure to be preferred to all other approaches. However, this procedure is also the most expensive of the evaluated methods. Supervised learning implies human, financial, and time resources for annotating the training documents. A training data set comprising 1,000 instances is very small for usual supervised learning settings, but the coding process also has to be monitored and coordinated. Moreover, active learning involves a dynamic labeling process in which after each iteration those documents are annotated for which a label is requested by the model. While active learning reduces the overall number of training instances for which a label is required, the dynamic labeling process may increase coordination costs or the time coders spend on coding as they wait for the model to request the next labels.

The precise separation of documents that refer to the entity of interest and thus are relevant for the planned study at hand from documents that are irrelevant is an essential analytic step. This step defines the set of documents on which all the following analyses are conducted. Selection biases induced by the applied retrieval method ultimately bias the study's results. Therefore, attention and care should be taken when it comes to extracting relevant documents. Compared to the creation of a set of keywords, active learning requires substantive amounts of additional resources. But given the observed considerably higher retrieval performances achieved by active learning compared to keyword lists, spending these resources is likely to be worthwhile for the quality of the study.

The aim of this study was to compare different learning approaches: keyword lists, query expansion techniques, topic model-based classification rules, and active as well as passive supervised learning. One problem that naturally arises if one seeks to compare learning approaches is that different approaches can only be compared on the basis of specific models that follow specific procedures (e.g. for query expansion), that have specific hyperparameter settings, that are trained on a specific finite set of training documents, that are evaluated on a specific finite set of test set documents, and that are initialized by specific random seed values (Reimers & Gurevych, 2018). Here, care was taken to have a broad range of several specific models with different settings for each approach. With regard to keyword lists and query expansion, 100 different keyword lists were expanded in local as well as global embedding spaces (where the number of expansion terms was varied from 1 to 9). For topic-model-based classification rules, four different values for threshold ξ for each of 426,725 combinations were evaluated. With regard to active and passive supervised learning, two different types of models (SVM and BERT) were applied 10 or 5 times with different random initializations. In each of the 10 or 5 runs, a different initial training set was used that then was enlarged by passive random sampling or active selection in 15 iterations. This broad evaluation setting makes it more likely that the conclusions drawn here on the set of models evaluated for each approach hold and that active learning indeed is superior to keyword lists for the studied tasks.

Nevertheless, future studies might inspect the effect of the chosen model settings of the evaluated approaches on the obtained results: Here, for each of the evaluated approaches, the most simple setting or version was used. For query expansion, GloVe embeddings that represent each term by a single vector were employed and a simple Boolean query using the OR operator was conducted. More complex procedures from the field of information retrieval that make use of contextualized embeddings and pseudo-relevant feedback (e.g. Zheng et al., 2020) were not applied. For the estimation of the topics, the CTM rather than, for example, a neural topic model was used (for an overview of neural topic models see Zhao et al., 2021). For passive supervised learning, simple random oversampling was employed, and for active learning, uncertainty sampling was applied as a query strategy rather than more complex procedures such as query-by-committee or expected model change (see e.g. Settles, 2010). These simplest versions applied here present the core idea of each approach often most clearly and also are most easy to implement for scientists that seek to use one of the approaches as a first step in their analysis. Nevertheless, future studies might explore

whether applying more complex procedures changes the substantive conclusions reached here.

Finally, there is the question of in how far the conclusions drawn here on the basis of three applications travel to further contexts. The selected data sets and tasks differ with regard to textual length and style, the heterogeneity of the corpora, the characteristics of the entities of interest, and the share of relevant documents in the corpora. The finding that active supervised learning—if applied with a not too small number of training instances—considerably increases the F_1-Score compared to keyword lists holds across these applications' differences. But further studies could look more closely at which contextual factors lead to which effects on retrieval performance for which procedures.

Appendix to A Comparison of Approaches for Imbalanced Classification Problems in the Context of Retrieving Relevant Documents for an Analysis

3.A Most Predictive Terms

Note on Tables 3.A.1 to 3.A.3: The keyword lists comprising empirically highly predictive terms are not only applied on the corpora to evaluate the retrieval performance of keyword lists but also form the basis for query expansion (see Section 3.4.2.2). The query expansion technique makes use of GloVe word embeddings (Pennington et al., 2014) trained on the local corpora at hand and also makes use of externally obtained GloVe word embeddings trained on large global corpora. In the case of the locally trained word embeddings, there is a learned word embedding for each predictive term. Thus, the set of extracted highly predictive terms can be directly used as starting terms for query expansion. In the case of the globally pretrained word embeddings, however, not all of the highly predictive terms have a corresponding global word embedding. Hence, for the globally pretrained embeddings, the 50 most predictive terms *for which a globally pretrained word embedding is available* are extracted. If a predictive term has no corresponding global embedding, the set of extracted predictive terms is enlarged with the next most predictive term until there are 50 extracted terms. In consequence, below for each corpus two lists of the most predictive features are shown.

A	B
#flüchtlinge	migranten
migranten	asylanten
#refugeeswelcome	flüchtlingen
#refugee	asylrecht
asylanten	asylunterkunft
flüchtlingen	asyl
asylrecht	flüchtlinge
#migration	asylbewerber
#fluechtlinge	ausländer
asylunterkunft	flüchtlingsheim
asyl	refugees
flüchtlinge	flüchtling
migrationshintergrund	asylpolitik
asylbewerber	ungarn
ausländer	refugee
flüchtlingsheim	mittelmeer
flüchtlingsheime	kritisiert
flüchtlingskrise	syrer
#schauhin	syrischen
refugees	bamberg
flüchtling	brandanschlag
asylpolitik	behandelt
ungarn	merkels
#refugeecamp	ermittelt
#bloggerfuerfluechtlinge	zusammenhang
#refugees	innenminister
#flüchtlingen	kundgebung
flüchtlingsheimen	pegida
#asyl	welcome
refugee	unterbringung
mittelmeer	migration
kritisiert	benötigt
syrer	erfahrungen
proasyl	sollen
syrischen	heimat
bamberg	tja
brandanschlag	balkanroute
behandelt	rechte
merkels	dort
ermittelt	bürger
zusammenhang	merkel
innenminister	demo
kundgebung	mehrheit
pegida	letztes
welcome	geplante
#deutschland	recht
refugeeswlcm_le	hilfe
unterbringung	europa
ndaktuell	afghanistan
migration	islam

Table 3.A.1: **Most Predictive Features in the Twitter Data Set.** **A**: This is a list of the 50 terms that are extracted as the most predictive features for the relevant class of documents that refer to the refugee topic (most predictive term at the top). From this list of terms, 100 samples of 10 keywords are sampled to construct initial keyword lists that then are extended via query expansion using the locally trained GloVe word embeddings. **B**: This is a list of the 50 terms *for which a globally pretrained GloVe word embedding is available* that are extracted as the most predictive features for the relevant class of documents that refer to the refugee topic (most predictive term at the top). From this list of terms, 100 samples of 10 keywords are sampled to construct initial keyword lists that then are extended via query expansion using the globally trained GloVe word embeddings.

A	B
retard	retard
retards	retards
retarded	retarded
quadriplegic	quadriplegic
autistic	autistic
paralyzed	paralyzed
schizophrenic	schizophrenic
vegetables	vegetables
wheelchair	wheelchair
epileptic	epileptic
parkinson's	disabled
disabled	anorexic
anorexic	stevie
stevie	cripples
cripples	paralympics
paralympics	adhd
adhd	paraplegic
paraplegic	syndrome
syndrome	paralysed
paralysed	midget
midget	leper
leper	amputee
amputee	cripple
cripple	tons
tons	handicapped
handicapped	bipolar
bipolar	wheelchairs
wheelchairs	dyslexic
dyslexic	chromosome
chromosome	blind
blind	suicidal
suicidal	crippled
crippled	chromosomes
chromosomes	vegetable
vegetable	challenged
alzheimer's	special
challenged	veggie
special	cancer
veggie	spade
cancer	helen
spade	jenga
helen	autism
jenga	medication
autism	deaf
medication	logan
deaf	depressed
logan	christopher
depressed	mentally
christopher	potato
mentally	shouted

Table 3.A.2: **Most Predictive Features in the SBIC.** **A**: This is a list of the 50 terms that are extracted as the most predictive features for the relevant class of documents that are offensive toward disabled people (most predictive term at the top). From this list of terms, 100 samples of 10 keywords are sampled to construct initial keyword lists that then are extended via query expansion using the locally trained GloVe word embeddings. **B**: This is a list of the 50 terms *for which a globally pretrained GloVe word embedding is available* that are extracted as the most predictive features for the relevant class of documents that are offensive toward disabled people (most predictive term at the top). From this list of terms, 100 samples of 10 keywords are sampled to construct initial keyword lists that then are extended via query expansion using the globally trained GloVe word embeddings.

A	B
oil	oil
crude	crude
barrels	barrels
barrel	barrel
exploration	exploration
energy	energy
petroleum	petroleum
production	production
drilling	drilling
bpd	bpd
refinery	refinery
opec	opec
gulf	gulf
tanker	tanker
texas	texas
offshore	offshore
canada	canada
resources	resources
sea	sea
rigs	rigs
out	out
petrobras	petrobras
rig	rig
refineries	refineries
agency	agency
along	along
depressed	depressed
conoco	conoco
raise	raise
shelf	shelf
iranian	iranian
platform	platform
day	day
maintain	maintain
drill	drill
total	total
well	well
deal	deal
16	16
fiscal	fiscal
upon	upon
postings	postings
light	light
blocks	blocks
tuesday	tuesday
about	about
meters	meters
daily	daily
future	future
equivalent	equivalent

Table 3.A.3: **Most Predictive Features in Reuters-21578 Corpus.** **A**: This is a list of the 50 terms that are extracted as the most predictive features for the relevant class of documents that are about the crude oil topic (most predictive term at the top). From this list of terms, 100 samples of 10 keywords are sampled to construct initial keyword lists that then are extended via query expansion using the locally trained GloVe word embeddings. **B**: This is a list of the 50 terms *for which a globally pretrained GloVe word embedding is available* that are extracted as the most predictive features for the relevant class of documents that are about the crude oil topic (most predictive term at the top). From this list of terms, 100 samples of 10 keywords are sampled to construct initial keyword lists that then are extended via query expansion using the globally trained GloVe word embeddings.

3.B Recall and Precision of Keyword Lists and Query Expansion

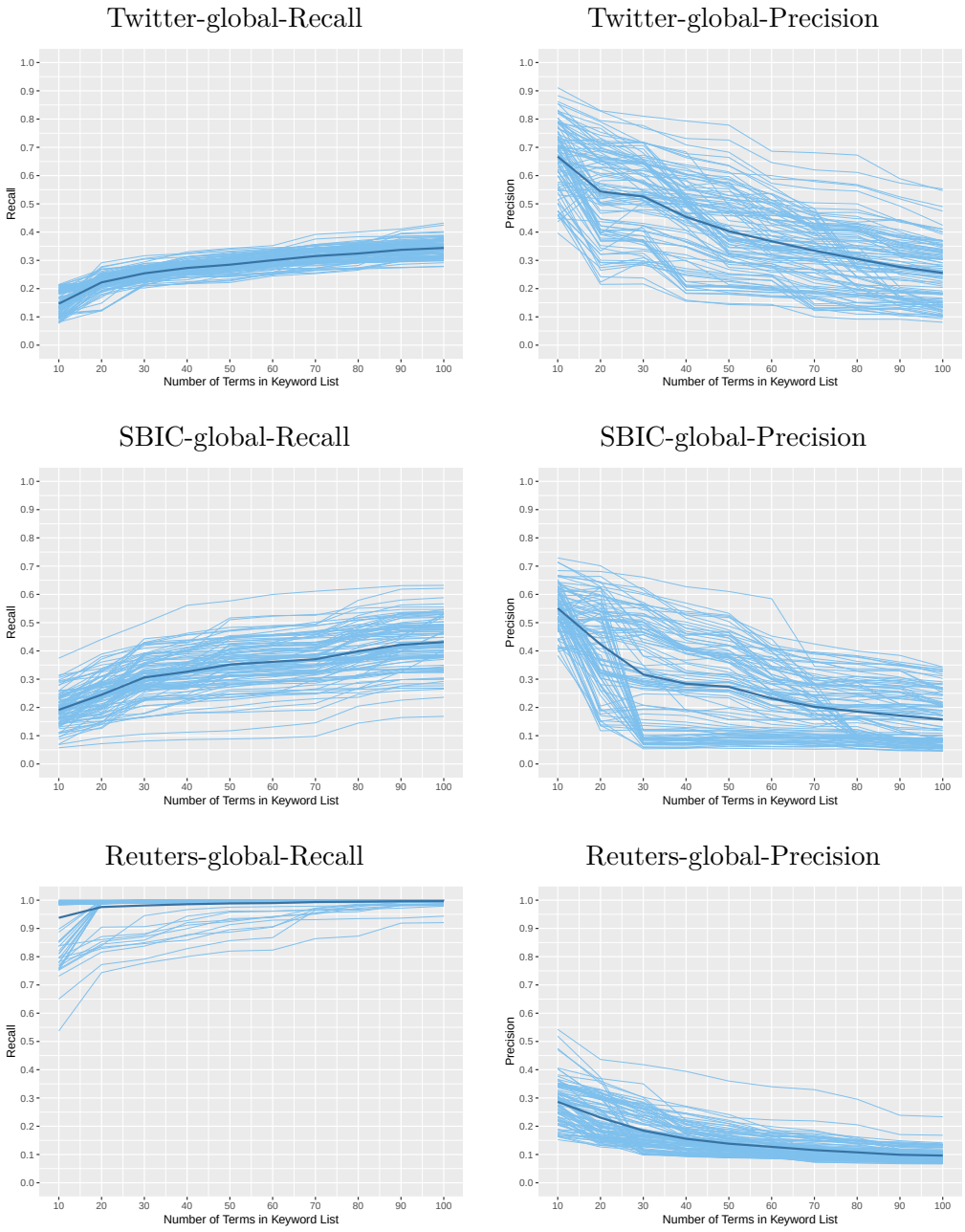


Figure 3.B.1: **Recall and Precision for Retrieving Relevant Documents with Keyword Lists and Global Query Expansion.** This plot shows recall and precision scores resulting from the application of the keyword lists of 10 highly predictive terms as well as the evolution of the recall and precision scores across the query expansion procedure based on globally trained GloVe embeddings. For each of the sampled 100 keyword lists that then are expanded, one light blue line is plotted. The thick dark blue line gives the mean over the 100 lists.

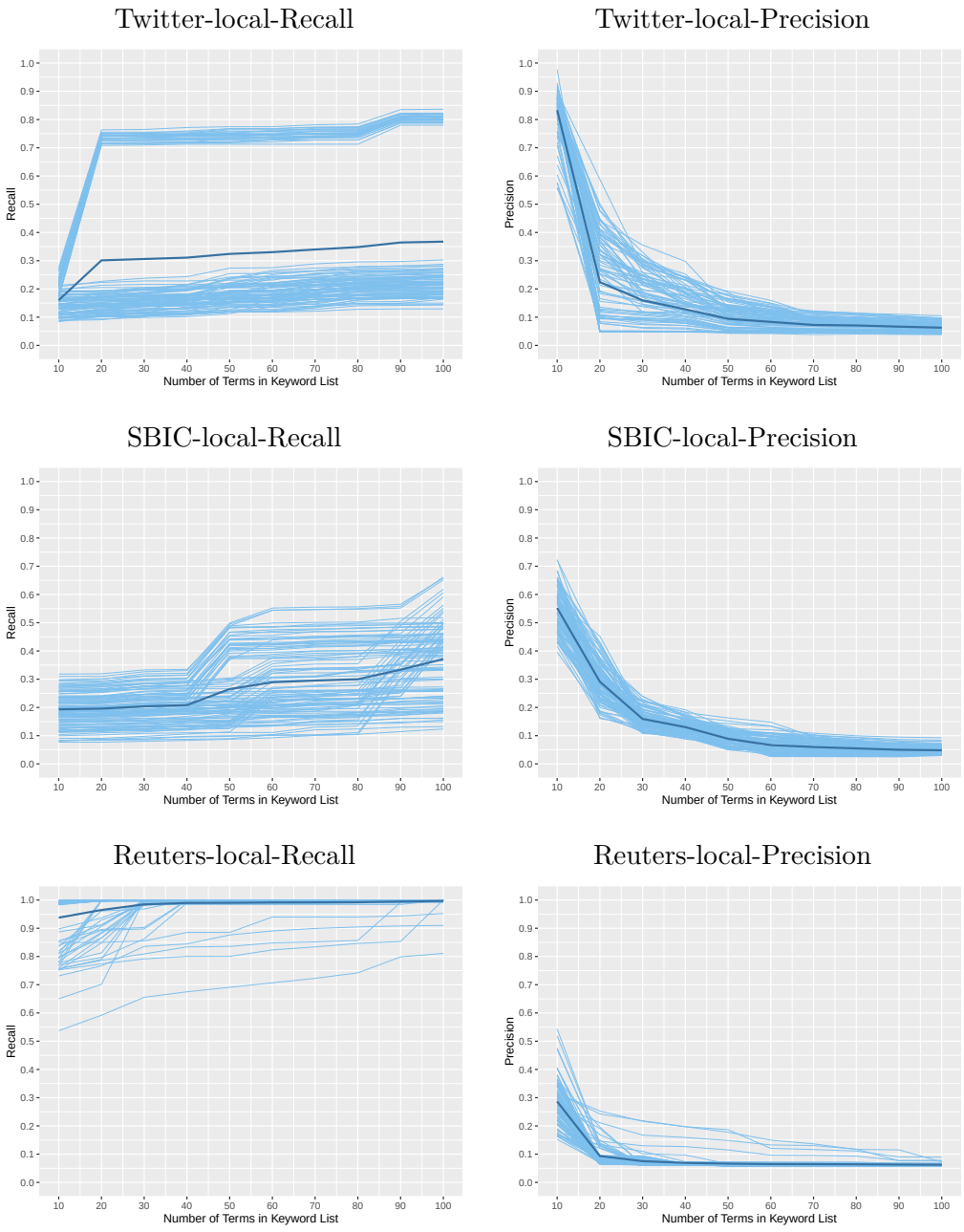


Figure 3.B.2: **Recall and Precision for Retrieving Relevant Documents with Keyword Lists and Local Query Expansion.** This plot shows recall and precision scores resulting from the application of the keyword lists of 10 highly predictive terms as well as the evolution of the recall and precision scores across the query expansion procedure based on locally trained GloVe embeddings. For each of the sampled 100 keyword lists that then are expanded, one light blue line is plotted. The thick dark blue line gives the mean over the 100 lists. Note that the strong increase in recall for some keyword lists in the Twitter data set is due to the fact that the textual feature with the highest cosine similarity to the highly predictive initial term *'flüchtlinge'* (translation: *'refugees'*) is the colon *':'*.

3.C Recall and Precision of Topic Model-Based Classification Rules

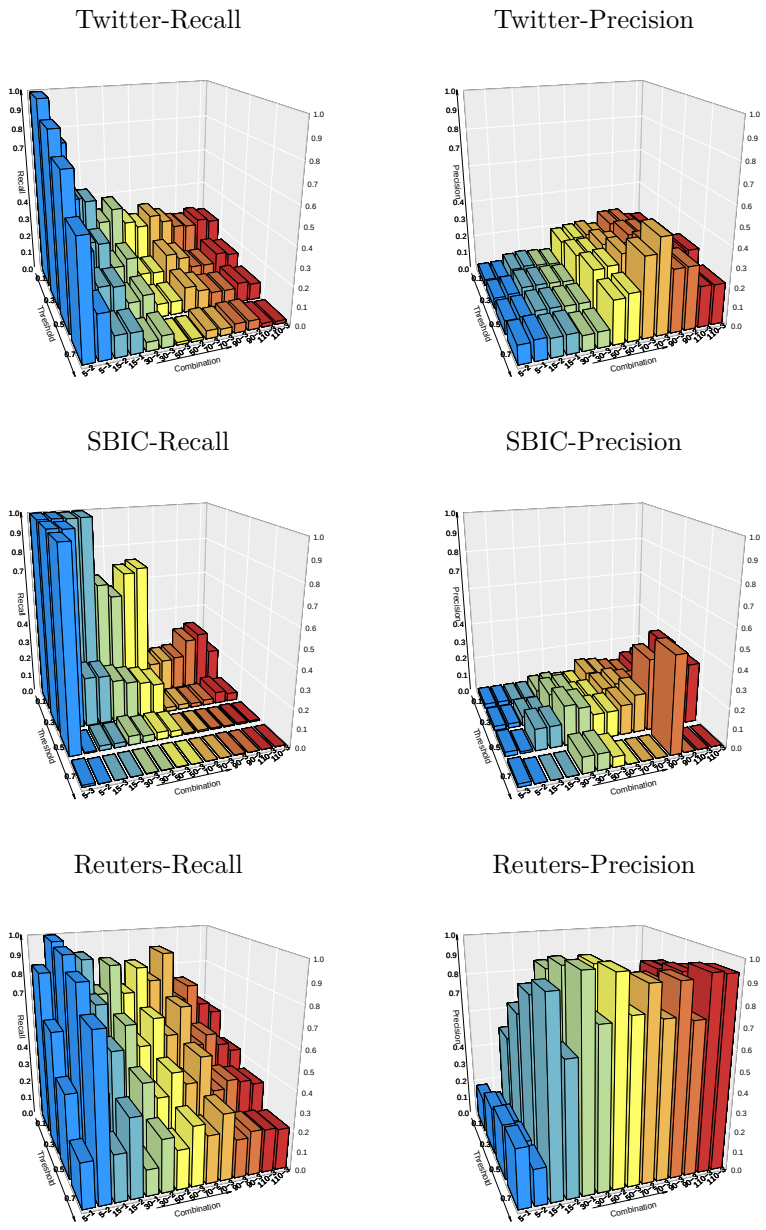


Figure 3.C.1: **Recall and Precision of Topic Model-Based Classification Rules.** The height of a bar indicates the recall values (left column) and precision values (right column) resulting from the application of a topic model-based classification rule. For each number of topics $K \in \{5, 15, 30, 50, 70, 90, 110\}$, those two combinations out of all explored combinations regarding the question how many and which topics are considered relevant are shown that reach the highest F_1-Score for the given topic number. For each combination, recall and precision values for each threshold value $\xi \in \{0.1, 0.3, 0.5, 0.7\}$ is given. Classification rules that assign none of the documents to the positive class have a recall value of 0 and an undefined value for precision and the F_1-Score. Undefined values here are visualized by the value 0.

3.D Recall and Precision of Active and Passive Supervised Learning

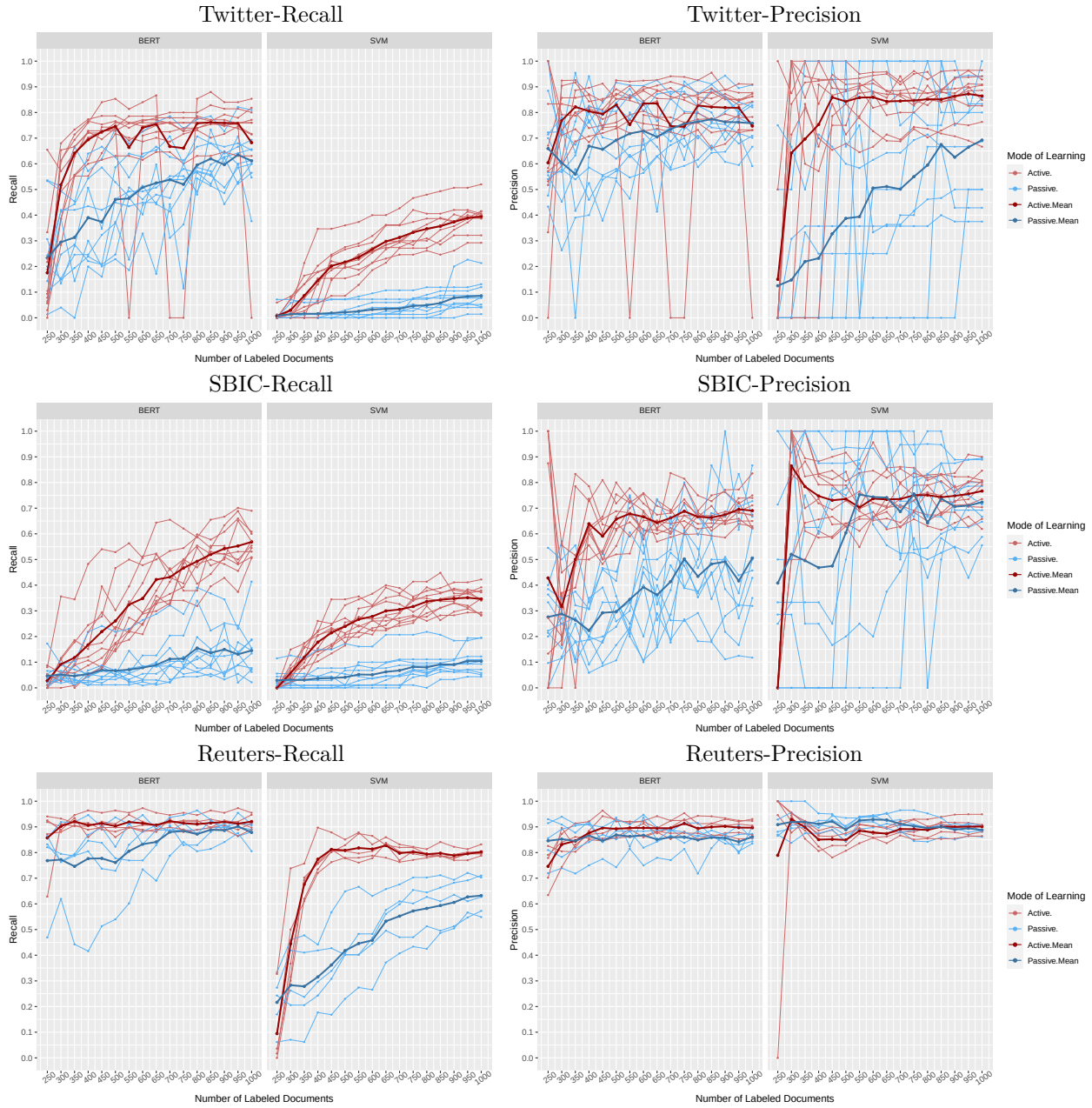


Figure 3.D.1: **Recall and Precision of Active and Passive Supervised Learning.** Recall values (left column) and precision values (right column) achieved on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. Passive supervised learning results are visualized by blue lines, active learning results are given in red. For each of the 10 (Twitter, SBIC) or 5 (Reuters) conducted iterations, one light colored line is plotted. The thick and dark lines give the means across the iterations. If a trained model assigns none of the documents to the positive relevant class, then it has a recall value of 0 and an undefined value for precision and the F_1-Score. Undefined values here are visualized by the value 0.

3.E Comparing BERT and SVM for Active and Passive Supervised Learning

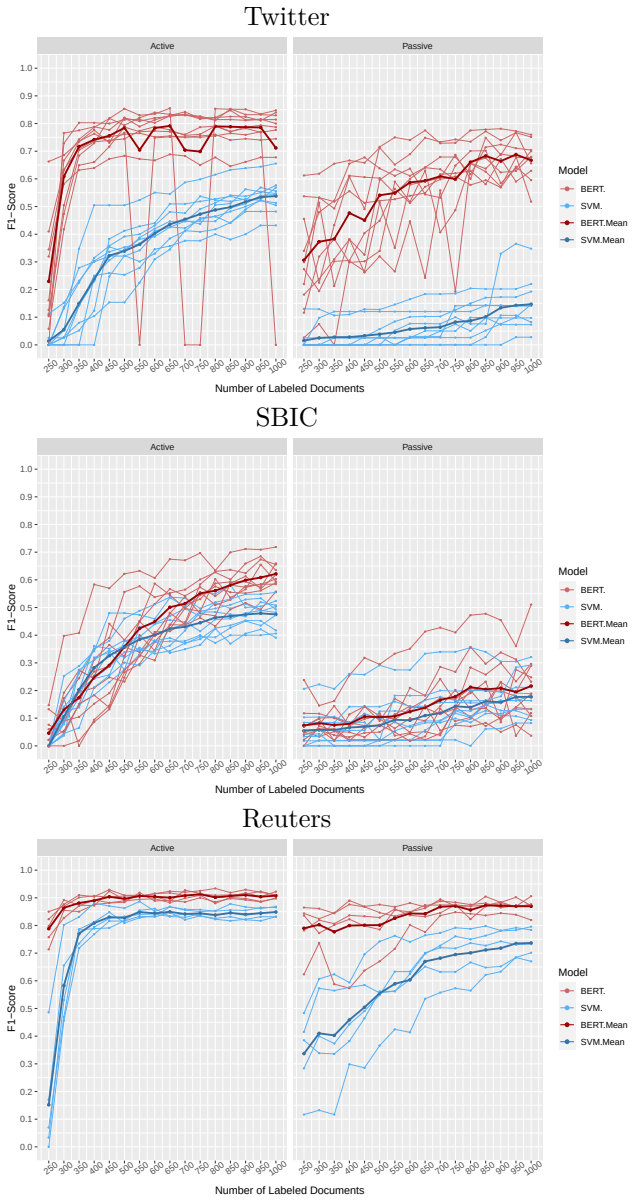


Figure 3.E.1: **Comparing BERT and SVM for Active and Passive Supervised Learning I.** F_1-Scores achieved on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. Active learning results are visualized in the left panels, passive learning results are given in the right panels. F_1-Scores of the SVMs are visualized by blue lines, BERT performances are given in red. For each of the 10 (Twitter, SBIC) or 5 (Reuters) conducted iterations, one light colored line is plotted. The thick and dark lines give the mean across the iterations. If a trained model assigns none of the documents to the positive relevant class, then it has a recall value of 0 and an undefined value for precision and the F_1-Score. Undefined values here are visualized by the value 0.

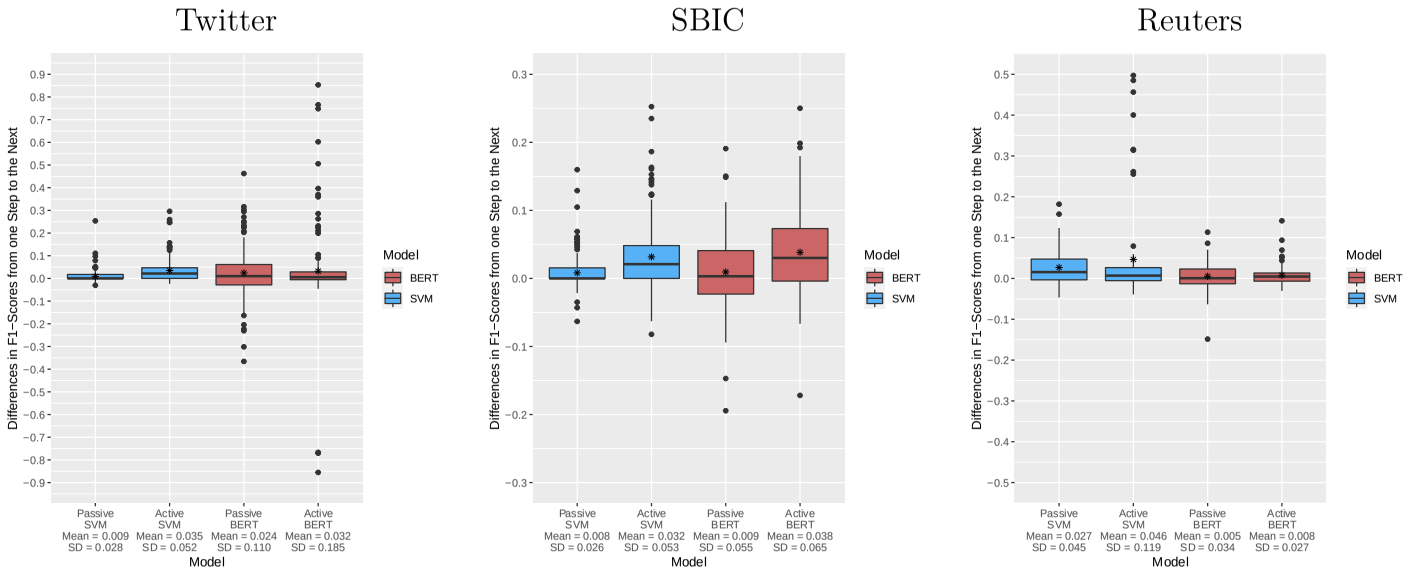


Figure 3.E.2: **Comparing BERT and SVM for Active and Passive Supervised Learning II.** Distribution of the differences in the F_1-Scores achieved on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from one training step to the next by a batch of 50 documents. Boxplots visualizing the distribution of differences in F_1-Scores of the SVMs are presented in blue. F_1-Score differences for BERT are given in red. The mean is visualized by a star dot. The value of the mean as well as the standard deviation (SD) are given below the respective boxplots.

3.F Comparing BERT with Different Hyperparameter Values

The hyperparameter values for BERT models trained with hyperparameter values in conventional value ranges are determined via hyperparameter tuning. As for the SVMs, a grid search across sets of hyperparameter values is implemented via stratified 5-fold cross-validation using one of the folds of the data. The AdamW algorithm (Loshchilov & Hutter, 2019) with a warmup period lasting 6% of the training steps is used. Dropout is set to 0.1. The batch size is set to 16. The inspected hyperparameter values for the global learning rate are {2e-05, 3e-05}, and for the number of epochs are $\{2, 3, 4, 5\}$. Stratification ensures that the share of training instances of the relevant minority class is the same across all folds. Moreover, random oversampling of the minority class is performed to increase the number of relevant minority class documents by a factor of 5. The evaluated hyperparameter setting that has the highest F_1-Score and does not display extreme overfitting is selected.

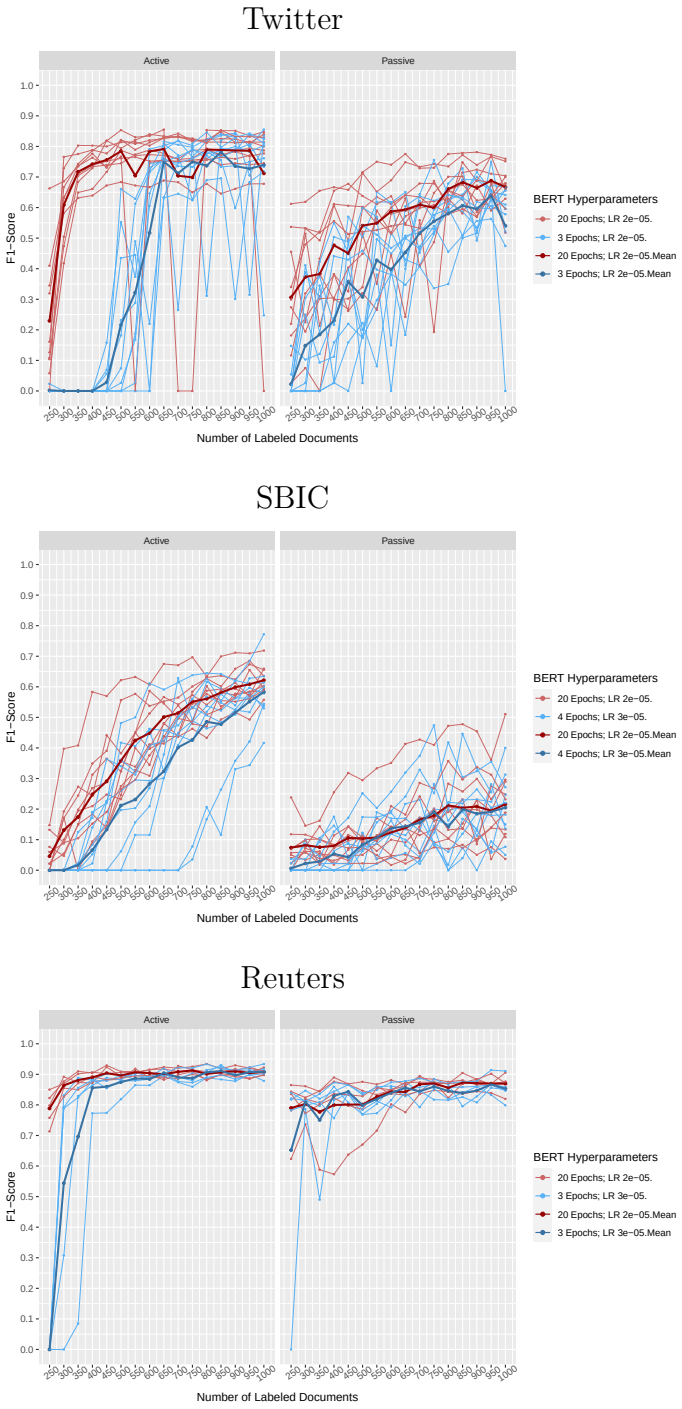


Figure 3.F.1: **Comparing BERT with Different Hyperparameter Values I.** F_1-Scores achieved by BERT models on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from 250 to 1,000. Active learning results are visualized in the left panels, passive learning results are given in the right panels. F_1-Scores of BERT models trained with a global learning rate of 2e-05 for 20 epochs are given in red, performances of BERT models trained with hyperparameter values in conventional value ranges are visualized by blue lines. The precise values for the number of epochs and the learning rates are specified in the legends beside the plots. The thick and dark lines give the means across the iterations. If a trained model assigns none of the documents to the positive relevant class, then it has a recall value of 0 and an undefined value for precision and the F_1-Score. Undefined values here are visualized by the value 0.

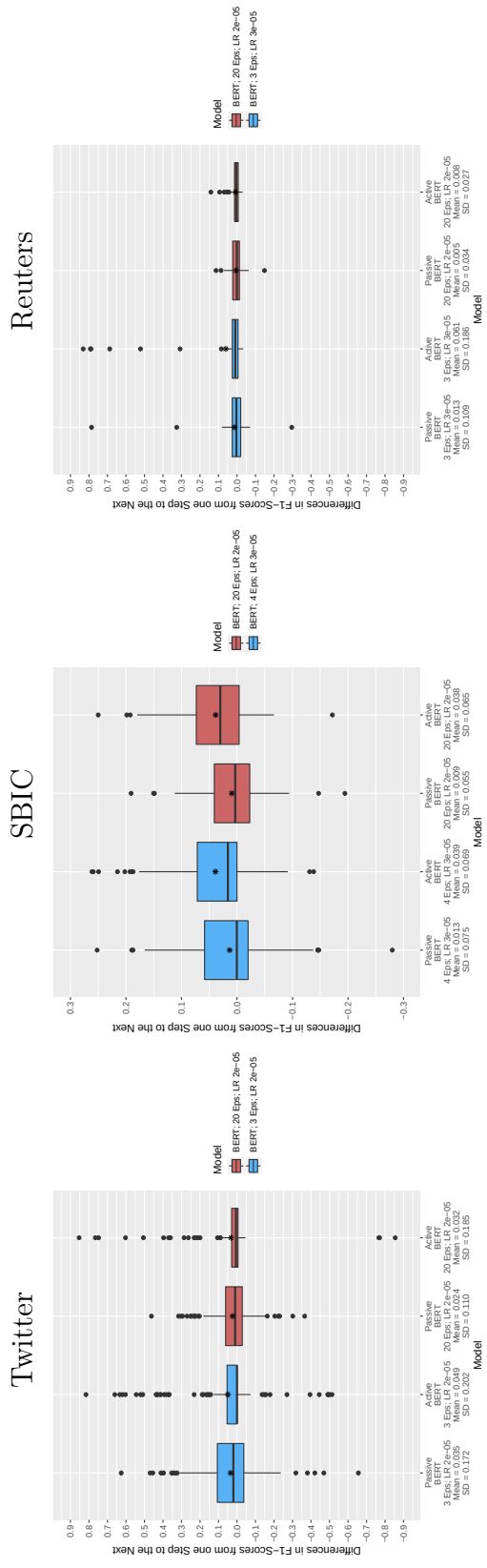


Figure 3.F.2: **Comparing BERT with Different Hyperparameter Values II.** Distributions of the differences in the F_1-Scores achieved on the set aside test set as the number of unique labeled documents in set $\mathcal{I}$ increases from one training step to the next by a batch of 50 documents. Boxplots visualizing the distribution of differences in F_1-Scores of BERT models trained with hyperparameter values in conventional value ranges are presented in blue. F_1-Score differences for BERT trained with a global learning rate (LR) of 2e-05 for 20 epochs (Eps) are given in red. The mean is visualized by a star dot. The value of the mean as well as the standard deviation (SD) are given below the respective boxplots.

Bibliography

Abdul Reda, A., Sinanoglu, S., & Abdalla, M. (2021). Mobilizing the masses: Measuring resource mobilization on Twitter. *Sociological Methods & Research*, (pp. 1–40). https://doi.org/10.1177/0049124120986197

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In J.-F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.), *Machine Learning: ECML 2004* (pp. 39–50). Springer. https://doi.org/10.1007/978-3-540-30115-8_7

Allaire, J. J., Francois, R., Ushey, K., Vandenbrouck, G., Geelnard, M., & Intel (2020). *RcppParallel: Parallel programming tools for 'Rcpp'*. (Version 5.0.2) [R Package]. CRAN. https://CRAN.R-project.org/package=RcppParallel

ALMasri, M., Berrut, C., & Chevallet, J.-P. (2013). Wikipedia-based semantic query enrichment. In P. N. Bennett, E. Gabrilovich, J. Kamps, & J. Karlgren (Eds.), *Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR '13)* (pp. 5–8). Association for Computing Machinery. https://doi.org/10.1145/2513204.2513209

Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Wu, Y., & Zhu, M. (2013). A practical algorithm for topic modeling with provable guarantees. *Proceedings of Machine Learning Research*, *28*(2), 280–288. https://proceedings.mlr.press/v28/arora13.html

Azad, H. K. & Deepak, A. (2019). Query expansion techniques for information retrieval: A survey. *Information Processing and Management*, *56*(5), 1698–1735. https://doi.org/10.1016/j.ipm.2019.05.009

Azar, E. E. (2009). *Conflict and Peace Data Bank (COPDAB), 1948-1978*. [Data set]. Inter-University Consortium for Political and Social Research. https://doi.org/10.3886/ICPSR07767.v4

Baden, C., Kligler-Vilenchik, N., & Yarchi, M. (2020). Hybrid content analysis: Toward a strategy for the theory-driven, computer-assisted classification of large text corpora. *Communication Methods and Measures*, *14*(3), 165–183. https://doi.org/10.1080/19312458.2020.1803247

Baerg, N. & Lowe, W. (2020). A textual Taylor rule: Estimating central bank preferences combining topic and scaling methods. *Political Science Research and Methods*, *8*(1), 106–122. https://doi.org/10.1017/psrm.2018.31

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. https://arxiv.org/abs/1409.0473

Barberá, P. (2016). Less is more? How demographic sample weights

can improve public opinion estimates based on Twitter data. Manuscript. http://pablobarbera.com/static/less-is-more.pdf

Bauer, P. C., Barberá, P., Ackermann, K., & Venetz, A. (2017). Is the left-right scale a valid measure of ideology? *Political Behavior*, *39*(3), 553–583. https://doi.org/10.1007/s11109-016-9368-2

Baum, M., Cohen, D. K., & Zhukov, Y. M. (2018). Does rape culture predict rape? Evidence from U.S. newspapers, 2000-2013. *Quarterly Journal of Political Science*, *13*(3), 263–289. http://dx.doi.org/10.1561/100.00016124

Bäuml, K.-H. (2007). Making memories unavailable: The inhibitory power of retrieval. *Journal of Psychology*, *215*(1), 4–11. https://doi.org/10.1027/0044-3409.215.1.4

Beauchamp, N. (2017). Predicting and interpolating state-level polls using Twitter textual data. *American Journal of Political Science*, *61*(2), 490–503. https://doi.org/10.1111/ajps.12274

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document Transformer. arXiv. https://arxiv.org/abs/2004.05150

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155. https://www.jmlr.org/papers/v3/bengio03a.html

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Blei, D. M. & Lafferty, J. D. (2007). A Correlated Topic Model of science. *The Annals of Applied Statistics*, *1*(1), 17–35. https://doi.org/10.1214/07-AOAS114

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. https://doi.org/10.1162/tacl_a_00051

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)* (pp. 144–152). Association for Computing Machinery. https://doi.org/10.1145/130385.130401

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, *49*(2), 1–50. https://doi.org/10.1145/2907070

Brownlee, J. (2020). Cost-sensitive learning for imbalanced classification. *Machine Learning*

Mastery. https://machinelearningmastery.com/cost-sensitive-learning-for-imbalanced-classification/

Brownlee, J. (2021a). Random oversampling and undersampling for imbalanced classification. *Machine Learning Mastery*. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Brownlee, J. (2021b). SMOTE for imbalanced classification with Python. *Machine Learning Mastery*. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Burnap, P., Gibson, R., Sloan, L., Southern, R., & Williams, M. (2016). 140 characters to victory?: Using Twitter to predict the UK 2015 general election. *Electoral Studies*, 41, 230–233. https://doi.org/10.1016/j.electstud.2015.11.017

Chawla, N. V. (2005). Data mining for imbalanced datasets: An overview. In O. Maimon & L. Rokach (Eds.), *The Data Mining and Knowledge Discovery Handbook* (pp. 853–867). Springer.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*(1), 321–357. https://doi.org/10.1613/jair.953

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., & Swinton, J. (2019). *xtable: Export tables to LaTeX or HTML*. (Version 1.8-4) [R Package]. CRAN. https://cran.r-project.org/web/packages/xtable/index.html

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Diaz, F., Mitra, B., & Craswell, N. (2016). Query expansion with locally-trained word embeddings. In K. Erk & N. A. Smith (Eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 367–377). Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1035

Diermeier, D., Godbout, J.-F., Yu, B., & Kaufmann, S. (2011). Language and ideology in Congress. *British Journal of Political Science*, *42*(1), 31–55. https://doi.org/10.1017/S0007123411000160

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020).

Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv. https://arxiv.org/abs/2002.06305v1

D'Orazio, V., Landis, S. T., Palmer, G., & Schrodt, P. (2014). Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Political Analysis*, *22*(2), 224–242. https://doi.org/10.1093/pan/mpt030

Dowle, M. & Srinivasan, A. (2020). *data.table: Extension of 'data.frame'*. (Version 1.13.0) [R Package]. CRAN. https://CRAN.R-project.org/package=data.table

Durrell, M. (2008). Linguistic variable - linguistic variant. In U. Ammon, N. Dittmar, K. J. Mattheier, & P. Trudgill (Eds.), *Sociolinguistics* (pp. 195–200). De Gruyter Mouton. https://doi.org/10.1515/9783110141894.1.2.195

Ein-Dor, L., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., & Slonim, N. (2020). Active learning for BERT: An empirical study. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7949–7962). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.638

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)* (pp. 973–978). Morgan Kaufmann Publishers Inc.

Erlich, A., Dantas, S. G., Bagozzi, B. E., Berliner, D., & Palmer-Rubin, B. (2021). Multi-label prediction for political text-as-data. *Political Analysis*, (pp. 1–18). https://doi.org/10.1017/pan.2021.15

Ertekin, S., Huang, J., Bottou, L., & Giles, L. (2007). Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM '07)* (pp. 127–136). Association for Computing Machinery. https://doi.org/10.1145/1321440.1321461

Eshima, S., Imai, K., & Sasaki, T. (2021). Keyword assisted topic models. arXiv. https://arxiv.org/abs/2004.05964v2

Firth, J. R. (1957). *Studies in linguistic analysis*. Publications of the Philological Society. Blackwell.

Fogel-Dror, Y., Shenhav, S. R., Sheafer, T., & Atteveldt, W. V. (2019). Role-based association of verbs, actions, and sentiments with entities in political discourse. *Communication Methods and Measures*, *13*(2), 69–82. https://doi.org/10.1080/19312458.2018.1536973

Gessler, T. & Hunger, S. (2021). How the refugee crisis and radical right parties shape party competition on immigration. *Political Science Research and Methods*, (pp. 1–21). https://doi.org/10.1017/psrm.2021.64

Google Colaboratory (2020). Google Colaboratory Frequently Asked Questions. *research.google.com/colaboratory*. https://research.google.com/colaboratory/faq.html

Grimmer, J. (2013). Appropriators not position takers: The distorting effects of electoral incentives on Congressional representation. *American Journal of Political Science*, *57*(3), 624–642. https://doi.org/10.1111/ajps.12000

Grün, B. & Hornik, K. (2011). topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, *40*(13), 1–30. https://doi.org/10.18637/jss.v040.i13

Howard, J. & Ruder, S. (2018). Universal language model fine-tuning for text classification. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (pp. 328–339). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1031

Hugging Face (2021). Dataset card for reuters21578. *huggingface.co*. https://huggingface.co/datasets/reuters21578

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Jungherr, A., Schoen, H., & Jürgens, P. (2016). The mediation of politics through Twitter: An analysis of messages posted during the campaign for the German federal election 2013. *Journal of Computer-Mediated Communication*, *21*(1), 50–68. https://doi.org/10.1111/jcc4.12143

Katagiri, A. & Min, E. (2019). The credibility of public and private signals: A document-based approach. *American Political Science Review*, *113*(1), 156–172. https://doi.org/10.1017/S0003055418000643

Kentaro, W. (2020). *gdown: Download a large file from Google Drive*. [Python package]. github.com/wkentaro. https://github.com/wkentaro/gdown

King, G., Lam, P., & Roberts, M. E. (2017). Computer-assisted keyword and document set discovery from unstructured text. *American Journal of Political Science*, *61*(4), 971–988. https://doi.org/10.1111/ajps.12291

King, G., Pan, J., & Roberts, M. E. (2013). How censorship in China allows government criticism but silences collective expression. *American Political Science Review*, *107*(2), 326–343. https://doi.org/10.1017/S0003055413000014

Kouw, W. M. & Loog, M. (2019). A review of domain adaptation without target labels. arXiv. https://arxiv.org/abs/1901.05335

Krippendorff, K. (2013). *Content analysis: An introduction to its methodology (3rd ed.)*. SAGE Publications.

Kuzi, S., Shtok, A., & Kurland, O. (2016). Query expansion using word embeddings. In S. Mukhopadhyay & C. Zhai (Eds.), *Proceedings of the 25th ACM International on*

Conference on Information and Knowledge Management (CIKM '16) (pp. 1929–1932). Association for Computing Machinery. https://doi.org/10.1145/2983323.2983876

Lavrenko, V. & Croft, W. B. (2001). Relevance based language models. In D. H. Kraft, W. B. Croft, D. J. Harper, & J. Zobel (Eds.), *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)* (pp. 120–127). Association for Computing Machinery. https://doi.org/10.1145/383952.383972

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1–5. http://jmlr.org/papers/v18/16-365.html

Lewis, D. D. (1997). *Reuters-21578 (Distribution 1.0)*. [Data set]. http://www.daviddlewis.com/resources/testcollections/reuters21578/

Lewis, D. D. & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In B. W. Croft & C. J. van Rijsbergen (Eds.), *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)* (pp. 3–12). Springer. https://dl.acm.org/doi/10.5555/188490.188495

Linder, F. (2017). Reducing bias in online text datasets: Query expansion and active learning for better data from keyword searches. SSRN. http://dx.doi.org/10.2139/ssrn.3026393

Loshchilov, I. & Hutter, F. (2019). Decoupled weight decay regularization. In T. Sainath (Ed.), *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net. https://openreview.net/forum?id=Bkg6RiCqY7

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., Pfetsch, B., Heyer, G., Reber, U., Häussler, T., Schmid-Petri, H., & Adam, S. (2018). Applying LDA topic modeling in communication research: Toward a valid and reliable methodology. *Communication Methods and Measures*, *12*(2-3), 93–118. https://doi.org/10.1080/19312458.2018.1430754

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press. https://nlp.stanford.edu/fsnlp/

McKinney, W. (2010). Data structures for statistical computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference (SciPy 2010)* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Michael Waskom and Team (2020). *Seaborn*. [Python package]. Zenodo. https://zenodo.org/record/4379347

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv. https://arxiv.org/abs/1301.3781

Mikolov, T., Yih, W.-t., & Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In L. Vanderwende, H. Daumé III, & K. Kirchhoff (Eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 746–751). Association for Computational Linguistics. https://www.aclweb.org/anthology/N13-1090

Miller, B., Linder, F., & Mebane, W. R. (2020). Active learning approaches for labeling text: Review and assessment of the performance of active learning approaches. *Political Analysis*, *28*(4), 532–551. https://doi.org/10.1017/pan.2020.4

Moore, W. H. & Siegel, D. A. (2013). *A mathematics course for political and social research*. Princeton University Press.

Mosbach, M., Andriushchenko, M., & Klakow, D. (2021). On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*. https://openreview.net/forum?id=nzpLWnVAyah

Muchlinski, D., Yang, X., Birch, S., Macdonald, C., & Ounis, I. (2021). We need to go deeper: Measuring electoral violence using convolutional neural networks and social media. *Political Science Research and Methods*, *9*(1), 122–139. https://doi.org/10.1017/psrm.2020.32

Münchener Digitalisierungszentrum der Bayerischen Staatsbibliothek (dbmdz) (2021). Model card for bert-base-german-uncased from dbmdz. *huggingface.co*. https://huggingface.co/dbmdz/bert-base-german-uncased

Neelakantan, A., Shankar, J., Passos, A., & McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1059–1069). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1113

Oliphant, T. E. (2006). *A Guide to NumPy*. Trelgol Publishing.

Oller Moreno, S. (2021). *facetscales: facet grid with different scales per facet*. (Version 0.1.0.9000) [R Package]. github.com/zeehio. https://github.com/zeehio/facetscales

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, (pp. 8024–8035). Curran Associates, Inc.

http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Phang, J., Févry, T., & Bowman, S. R. (2019). Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. arXiv. https://arxiv.org/abs/1811.01088v2

Pilehvar, M. T. & Camacho-Collados, J. (2020). *Embeddings in natural language processing: Theory and advances in vector representations of meaning*. Morgan & Claypool Publishers. https://doi.org/10.2200/S01057ED1V01Y202009HLT047

Pilny, A., McAninch, K., Slone, A., & Moore, K. (2019). Using supervised machine learning in automated content analysis: An example using relational uncertainty. *Communication Methods and Measures*, *13*(4), 287–304. https://doi.org/10.1080/19312458.2019.1650166

Puglisi, R. & Snyder, J. M. (2011). Newspaper coverage of political scandals. *The Journal of Politics*, *73*(3), 931–950. https://doi.org/10.1017/s0022381611000569

Quinn, K. M., Monroe, B. L., Colaresi, M., Crespin, M. H., & Radev, D. R. (2010). How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science*, *54*(1), 209–228. https://doi.org/10.1111/j.1540-5907.2009.00427.x

R Core Team (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Raschka, S. (2020). *watermark*. [Python package]. github.com/rasbt. https://github.com/rasbt/watermark

Rauh, C., Bes, B. J., & Schoonvelde, M. (2020). Undermining, defusing or defending European integration? Assessing public communication of European executives in times of EU politicisation. *European Journal of Political Research*, 59, 397–423. https://doi.org/10.1111/1475-6765.12350

Reimers, N. & Gurevych, I. (2018). Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. arXiv. http://arxiv.org/abs/1803.09578

Richardson, L. (2020). *Beautiful Soup 4.* [Python library]. Crummy. https://www.crummy.com/software/BeautifulSoup/

Roberts, M. E., Stewart, B. M., & Airoldi, E. M. (2016a). A model of text for experimentation in the social sciences. *Journal of the American Statistical Association*, *111*(515), 988–1003. https://doi.org/10.1080/01621459.2016.1141684

Roberts, M. E., Stewart, B. M., & Tingley, D. (2016b). Navigating the local modes of big data: The case of topic models. In R. M. Alvarez (Ed.), *Computational Social Science: Discovery and Prediction*, (pp. 51–97). Cambridge University Press, New York.

Roberts, M. E., Stewart, B. M., & Tingley, D. (2019). stm: An R package for Structural Topic Models. *Journal of Statistical Software*, *91*(2), 1–40. https://doi.org/10.18637/jss.v091.i02

Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Luis, J. L., Gadarian, S. K., Albertson, B., & Rand, D. G. (2014). Structural Topic Models for open-ended survey responses. *American Journal of Political Science*, *58*(4), 1064–1082. https://doi.org/10.1111/ajps.12103

Rodriguez, P. L. & Spirling, A. (2022). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. *The Journal of Politics*, *84*(1), 101–115. https://doi.org/10.1086/715162

Ruder, S. (2019). *Neural transfer learning for natural language processing.* PhD thesis, National University of Ireland, Galway. https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf

Ruder, S. (2020). NLP-Progress. *nlpprogress.com.* https://nlpprogress.com/english/text_classification.html

Sap, M., Gabriel, S., Qin, L., Jurafsky, D., Smith, N. A., & Choi, Y. (2020). Social bias frames: Reasoning about social and power implications of language. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 5477–5490). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.486

Schulze, P., Wiegrebe, S., Thurner, P. W., Heumann, C., Aßenmacher, M., & Wankmüller, S. (2021). Exploring topic-metadata relationships with the STM: A Bayesian approach. arXiv. https://arxiv.org/abs/2104.02496

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123. https://aclanthology.org/J98-1004

scikit-learn Developers (2020a). 1.4. Support Vector Machines. *scikit-learn.* https://scikit-learn.org/stable/modules/svm.html

scikit-learn Developers (2020b). RBF SVM Parameters. *scikit-learn.* https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Sebők, M. & Kacsuk, Z. (2021). The multiclass classification of newspaper articles with machine learning: The hybrid binary snowball approach. *Political Analysis*, *29*(2), 236–249. https://doi.org/10.1017/pan.2020.27

Selivanov, D., Bickel, M., & Wang, Q. (2020). *text2vec: Modern text mining framework for R.* [R package]. CRAN. https://CRAN.R-project.org/package=text2vec

Settles, B. (2010). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. http://burrsettles.com/pub/settles.activelearning.pdf

Silva, A. & Mendoza, M. (2020). Improving query expansion strategies with word embeddings. In *Proceedings of the ACM Symposium on Document Engineering 2020 (DocEng '20)* (pp. 1–4). Association for Computing Machinery. https://doi.org/10.1145/3395027.3419601

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1631–1642). Association for Computational Linguistics. https://aclanthology.org/D13-1170

Soetaert, K. (2019). *plot3D: Plotting multi-dimensional data.* (Version 1.3) [R package]. CRAN. https://CRAN.R-project.org/package=plot3D

Stier, S., Bleier, A., Bonart, M., Mörsheim, F., Bohlouli, M., Nizhegorodov, M., Posch, L., Maier, J., Rothmund, T., & Staab, S. (2018). *Systematically monitoring social media: the case of the German federal election 2017.* GESIS - Leibniz-Institut für Sozialwissenschaften, Köln. https://doi.org/10.21241/ssoar.56149

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune BERT for text classification? arXiv. https://arxiv.org/abs/1905.05583v3

Tong, S. & Koller, D. (2001). Support Vector Machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2, 45–66. https://www.jmlr.org/papers/v2/tong01a.html

Turney, P. D. & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37, 141–188. https://doi.org/10.1613/jair.2934

Ushey, K., Allaire, J. J., Wickham, H., & Ritchie, G. (2020). *rstudioapi: Safely access the RStudio API.* (Version 0.11) [R package]. CRAN. https://CRAN.R-project.org/package=rstudioapi

Uyheng, J. & Carley, K. M. (2020). Bots and online hate during the COVID-19 pandemic:

Case studies in the United States and the Philippines. *Journal of Computational Social Science*, 3, 445–468. https://doi.org/10.1007/s42001-020-00087-4

van Atteveldt, W., Sheafer, T., Shenhav, S. R., & Fogel-Dror, Y. (2017). Clause analysis: Using syntactic information to automatically extract source, subject, and predicate from texts with an application to the 2008–2009 Gaza War. *Political Analysis*, *25*(2), 207–222. https://doi.org/10.1017/pan.2016.12

van Rijsbergen, C. J. (2000). *Information retrieval. Session 1: Introduction to information retrieval* [Lecture Notes]. Universität Duisburg Essen. https://www.is.inf.uni-due.de/courses/dortmund/lectures/ir_ws00-01/folien/keith_intro.ps

van Rossum, G. & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of Support Vector Machines. In T. Dean (Ed.), *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '99)* (pp. 55–60).

Wang, H. (2020). Logistic regression for massive data with rare events. *Proceedings of Machine Learning Research*, 119, 9829–9836. http://proceedings.mlr.press/v119/wang20a.html

Wankmüller, S. (2021). Neural transfer learning with Transformers for social science text analysis. arXiv. https://arxiv.org/abs/2102.02111

Watanabe, K. (2021). Latent semantic scaling: A semisupervised text analysis technique for new domains and languages. *Communication Methods and Measures*, *15*(2), 81–102. https://doi.org/10.1080/19312458.2020.1832976

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer. https://ggplot2.tidyverse.org/

Wickham, H. (2019). *stringr: Simple, consistent wrappers for common string operations*. (Version 1.4.0) [R package]. CRAN. https://CRAN.R-project.org/package=stringr

Wickham, H., François, R., Henry, L., & Müller, K. (2021). *dplyr: A grammar of data manipulation*. (Version 1.0.6) [R package]. CRAN. https://CRAN.R-project.org/package=dplyr

Wild, F. (2020). *lsa: Latent Semantic Analysis*. (Version 0.73.2) [R package]. CRAN. https://CRAN.R-project.org/package=lsa

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). Hugging Face's Transformers: State-of-the-art natural language processing. arXiv. https://arxiv.org/abs/1910.03771v5

Zhang, H. & Pan, J. (2019). CASM: A deep-learning approach for identifying collective action events with text and image data from social media. *Sociological Methodology*, *49*(1), 1–57. https://doi.org/10.1177/0081175019860244

Zhao, H., Phung, D., Huynh, V., Jin, Y., Du, L., & Buntine, W. (2021). Topic modelling meets deep neural networks: A survey. In Z.-H. Zhou (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (pp. 4713–4720). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2021/638

Zheng, Z., Hui, K., He, B., Han, X., Sun, L., & Yates, A. (2020). BERT-QE: Contextualized query expansion for document re-ranking. In T. Cohn, Y. He, & Y. Liu (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 4718–4728). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.424

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)* (pp. 19–27). IEEE. https://doi.org/10.1109/ICCV.2015.11

Chapter 4

How to Estimate Continuous Sentiments From Texts Using Binary Training Data

Article. Wankmüller, S. & Heumann, C. (2021). How to estimate continuous sentiments from texts using binary training data. In Evang, K., Kallmeyer, L., Osswald, R., Waszczuk, J., & Zesch, T. (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 182–192). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.16/

Author Contributions. Sandra Wankmüller had the basic idea for developing the method. She performed the analyses and wrote the article. Christian Heumann had the idea to aggregate predicted probabilities with a beta mixed model (instead of aggregating binary predictions with an item response theory model). This significantly increased the performance of the method. He also helped to revise the manuscript.

Source Code. https://doi.org/10.6084/m9.figshare.14381825

How to Estimate Continuous Sentiments From Texts Using Binary Training Data

Sandra Wankmüller
Ludwig-Maximilians-Universität
Munich, Germany
https://orcid.org/0000-0002-4003-1704
sandra.wankmueller@gsi.lmu.de

Christian Heumann
Ludwig-Maximilians-Universität
Munich, Germany
chris@stat.uni-muenchen.de

Abstract

Although sentiment is conceptualized as a continuous variable, most text-based sentiment analyses categorize texts into discrete sentiment categories. Compared to discrete categorizations, continuous sentiment estimates provide much more detailed information which can be used for more fine-grained analyses by researchers and practitioners alike. Yet, existing approaches that estimate continuous sentiments either require detailed knowledge about context and compositionality effects or require granular training labels, that are created in resource intensive annotation processes. Thus, existing approaches are too costly to be applied for each potentially interesting application. To overcome this problem, this work introduces CBMM (standing for *c*lassifier-based *b*eta *m*ixed *m*odeling procedure). CBMM aggregates the predicted probabilities of an ensemble of binary classifiers via a beta mixed model and thereby generates continuous, real-valued output based on mere binary training input. CBMM is evaluated on the Stanford Sentiment Treebank (SST) (Socher et al., 2013), the V-reg data set (Mohammad et al., 2018), and data from the 2008 American National Election Studies (ANES) (The American National Election Studies, 2015). The results show that CBMM produces continuous sentiment estimates that are acceptably close to the truth and not far from what could be obtained if highly fine-grained training data were available.

1 Introduction

In natural language processing and computer science, the term *sentiment* typically refers to a loosely defined, broad umbrella concept: Feeling, emotion, judgement, evaluation, and opinion all fall under the term sentiment or are used synonymously with it (Pang and Lee, 2008; Liu, 2015). Interestingly, the broad notion of sentiment is very well captured by the psychological concept of an attitude (Liu, 2015). In psychology, scholars agree that an attitude is a summary evaluation of an entity (Banaji and Heiphetz, 2010; Albarracin et al., 2019). An attitude is the aggregated evaluative response resulting from a multitude of different (and potentially conflicting) information bases relating to the attitude entity (Fabringar et al., 2019). When putting the definition of an attitude as an evaluative summary into mathematical terms, an attitude is a unidimensional, continuous variable ranging from highly negative to highly positive (Cacioppo et al., 1997). This notion that attitudes are continuous is also mirrored in the sentiment analysis literature in which sentiments are devised to vary in their levels of intensity (Liu, 2015).

Despite this conceptualization, in an overwhelming majority of studies textual sentiment expressions are measured as instances of discrete classes. Sentiment analysis often implies a binary or multiclass classification task in which texts are assigned into two or three classes, thereby distinguishing positive from negative sentiments and sometimes a third neutral category (e.g. Pang et al., 2002; Turney, 2002; Maas et al., 2011). Other studies pursue ordinal sentiment classification (e.g. Pang and Lee, 2005; Thelwall et al., 2010; Socher et al., 2013; Kim, 2014; Zhang et al., 2015; Cheang et al., 2020). Here, texts fall into one out of several discrete and ordered categories.

If researchers would generate continuous—rather than discrete—sentiment estimates, this would not only align the theoretical conceptualization of sentiment with the way it is measured but also would provide much more detailed information that in turn can be used by researchers and practitioners for more fine-grained analyses and more fine-tuned responses.

For example, in the plot on the right hand side in Figure 1, the distribution of the binarized sentiment values of the tweets in the V-reg data set (Mo-

hammad et al., 2018) is shown. If researchers and practitioners would operate only on this discrete sentiment categorization, the shape of the underlying continuous sentiment distribution would be unknown. In fact, all distributions shown on the left hand side in Figure 1 produce the plot on the right hand side in Figure 1 if the sentiment values are binarized in such way that tweets with a sentiment value of ≥ 0.5 are assigned to the positive class and otherwise are assigned to the negative class. Imagine that a team of researchers would be interested in the sentiments expressed toward a policy issue and they would only know the binarized sentiment values on the right hand side in Figure 1. The researchers would not be able to conclude whether the expressions toward the policy issue are polarized into a supporting and an opposing side, whether a large share of sentiment expressions is positioned in the neutral middle, or whether the sentiments are evenly spread out. Knowing the continuous sentiment values, however, would allow them to differentiate between these scenarios.

As will be elaborated in Section 2, existing approaches that estimate continuous sentiment values for texts rely on (1) the availability of a comprehensive, context-matching sentiment lexicon and the researcher's knowledge regarding how to accurately model compositionality effects, or (2) highly costly processes to create fine-grained training data.

Sentiment analysis thus would benefit from a technique that generates continuous sentiment predictions for texts and is less demanding concerning the required information or resources. To meet this need, this work explores in how far the here proposed *c*lassifier-based *b*eta *m*ixed *m*odeling approach (CBMM) can produce valid continuous (i.e. real-valued) sentiment estimates on the basis of mere binary training data. The method comprises three steps. First, for each training set document a binary class label indicating whether the document is closer to the negative or the positive extreme of the sentiment variable has to be created or acquired. Second, an ensemble of J classifiers is trained on the binary class labels to produce for each of N test set documents J predicted probabilities to belong to the positive class. Third, a beta mixed model with N document random intercepts and J classifier random intercepts is estimated on the predicted probabilities. The N document random intercepts are the documents' continuous sentiment estimates.

In the following section, existing approaches

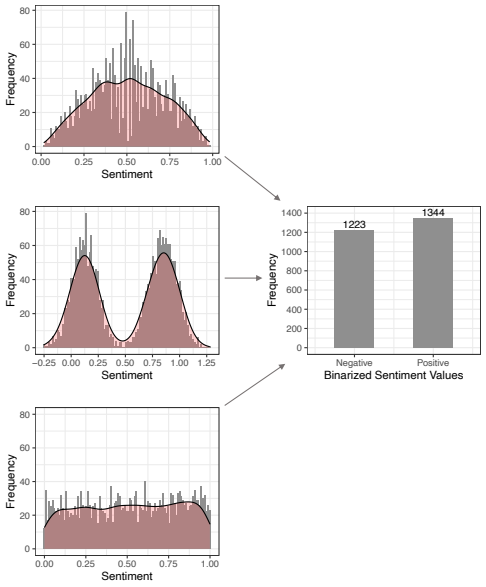


Figure 1: Continuous and Discrete Sentiment Distributions. Right plot: Binarized sentiment values of the tweets in the V-reg data set (Mohammad et al., 2018). Left plots: Histograms and kernel density estimates for three continuous distributions of sentiments that produce the plot on the right hand side if the continuous sentiment values are binarized such that tweets with values of ≥ 0.5 are assigned to the positive class and otherwise are assigned to the negative class. The unimodal distribution at the top is the true distribution of sentiment values but the other two distributions would generate the same binary separation of tweets into positive and negative.

that generate continuous sentiments are reviewed (Section 2). Then, CBMM is introduced in detail (Section 3) before it is evaluated on the basis of the Stanford Sentiment Treebank (SST) (Socher et al., 2013), the V-reg data set (Mohammad et al., 2018), and data from the 2008 American National Election Studies (ANES) (The American National Election Studies, 2015) (Section 4). A concluding discussion follows in Section 5.

2 Related Work

This work is concerned with the estimation of continuous values for texts in applications in which the underlying, unidimensional, continuous variable (e.g. sentiment) is well defined and the researcher seeks to position the texts along exactly this variable. Hence, this work does not consider unsupervised approaches (e.g. Slapin and Proksch,

2008) and only considers methods in which information on the definition of the underlying variable explicitly enters the estimation of the texts' values. Among these methods, one can distinguish two major approaches: lexicon-based procedures and regression models that operate on fine-grained training data.[1]

2.1 Lexicon-Based Approaches

An ideal sentiment lexicon covers all features in the corpus of an application and precisely assigns each feature to the sentiment value the feature has in the thematic context of the application (Grimmer and Stewart, 2013; Gatti et al., 2016). A major difficulty of lexicon-based approaches, however, is that even such an ideal sentiment lexicon will not guarantee highly accurate sentiment estimates. The reason is that sentiment builds up through complex compositional effects (Socher et al., 2013). These compositional effects either can be modeled via human-created rules or can be learned by supervised machine learning algorithms. Approaches that try to model compositionality via human-created rules range from simple formulas (e.g. Paltoglou et al., 2013; Gatti et al., 2016) to elaborate procedures (e.g. Moilanen and Pulman, 2007; Thet et al., 2010). Human-coded compositionality rules, however, tend to be outperformed by supervised machine learning algorithms (compare e.g. Gatti et al., 2016, Table 12 and Socher et al., 2013, Table 1). In the latter case, sentiment lexicons serve the purpose of creating the feature inputs to regression approaches—which are discussed next.

2.2 Regression Approaches

The second major set of approaches that generate real-valued sentiment estimates makes use of highly granular training data (e.g. as in the SST data set where each text is assigned to one out of 25 distinct values (Socher et al., 2013)). In these approaches, the fine-grained annotations are treated as if they were continuous and a regression model is applied.[2] Typically, the mean squared error (MSE) between the true granular labels and the real-valued predictions from the regression model is minimized. Regression approaches have shown to be able to generate continuous sentiment predictions that are quite close to the true fine-grained labels (Mohammad et al., 2018; Wang et al., 2018). Yet, the prerequisite for implementing such an approach is that fine-grained labels for the training data are available. Generating such granular annotations, however, is difficult and costly: Categorizing a training text into few ordinal categories is arguably a more easy task than assigning a text into one out of a large number of ordered values or even rating a text on a real-valued scale. As the number of distinct values increases, the number of inter- and intra-rater disagreements is likely to increase (Krippendorff, 2004). Hence, to produce reliable text annotations, it is advantageous to have each document rated several times by independent raters. The independent ratings then can be aggregated by taking the median or the mean of the ratings to obtain the final value (see e.g. Kiritchenko and Mohammad, 2017). The larger the number of raters for a document, the more reliable the final value assigned to the document. For this reason, generating reliable fine-grained labels for training documents via rating scale annotations requires a resource intensive annotation process.

The best-worst scaling (BWS) method in which coders have to identify the most positive and the most negative document among tuples of documents (typically 4-tuples), alleviates the problems of inter- and intra-rater inconsistencies (Kiritchenko and Mohammad, 2017). Yet, in order for the rankings among document tuples to generate valid real-valued ratings via the counting procedure implemented in BWS, it is essential that each document occurs in many different tuples such that each document is compared to many different other documents. This implies that a substantive number of unique tuples have to be annotated—which, in turn, demands respective human coding resources.

An alternative to the labeling of texts by human coders is the usage of already available information (e.g. if product reviews additionally come with numerical star ratings). The problem here, however, is that such information—if available at all—often comes in the form of discrete variables with only few distinct values (e.g. 5-star rating systems).

[1] Techniques for estimating continuous document positions on an a priori defined unidimensional latent variable also have been developed in political science. These methods either are at their core lexicon-based approaches (Watanabe, 2021) or require continuous values for the training documents (Laver et al., 2003)—and thus have the same shortcomings as either lexicon-based or regression approaches.

[2] Note that here, in correspondence with machine learning terminology, regression refers to statistical models and algorithms that model a real-valued response variable—which typically is assumed to follow a normal distribution.

To conclude, it is difficult and resource intensive to create or acquire fine-grained training data that is so detailed that it can be treated as if it were continuous. Not each team of researchers or practitioners will have the resources to create detailed training annotations and thus regression models cannot be applied to each substantive application of interest. Hence, the question that this work addresses is: Can one generate continuous sentiments with fewer costs in a setting where inter- and intra-rater inconsistencies are likely to be small? For example based on a simple binary coding of the training data?

3 Procedure

In the following, the three steps of the proposed CBMM procedure—(1) generating binary class labels, (2) training and applying an ensemble of classifiers, as well as (3) estimating a beta mixed model—are explicated. CBMM assumes that the documents to be analyzed are positioned on a latent, unidimensional, continuous sentiment variable. The aim is to estimate the test set documents' real-valued sentiment positions. The test set documents are indexed as $i \in \{1 \dots N\}$ and their sentiment positions are denoted as $\boldsymbol{\theta} = [\theta_1 \dots \theta_i \dots \theta_N]^\top$.

3.1 Generating Binary Class Labels

The CBMM procedure starts by generating binary class labels for the training set documents, e.g. via human coding. The coders classify the training documents into two classes such that the binary class label of each training set document indicates whether the document is closer to the negative (0) or the positive (1) extreme of the sentiment variable. Alternatively to human coding, binarized external information (such as star ratings associated with texts) can be used as class label indicators.

3.2 Training and Applying an Ensemble of Classifiers

In the second step, an ensemble of classification algorithms, indexed as $j \in \{1 \dots J\}$, is trained on the binary training data. The classifiers in the ensemble may differ regarding the type of algorithm, hyperparameter settings, or merely the seed values initializing the optimization process. After training, each classifier produces predictions for the N documents in the test set and each classifier's predicted probabilities for the test set documents to belong to the positive class are extracted. Thus, for each document i, a predicted probability to belong to class 1 is obtained from each classifier j, such that there are J predicted probabilities for each document: $\hat{\boldsymbol{y}}_i = [\hat{y}_{i1} \dots \hat{y}_{ij} \dots \hat{y}_{iJ}]$; whereby $\hat{y}_{ij}$ is classifier j's predicted probability for document i to belong to class 1.

3.3 Estimating a Beta Mixed Model

In step three, the aim is to infer the unobserved documents' continuous values on the latent sentiment variable from the observed predicted probabilities that have been generated by the set of classifiers. The approach taken here is similar to item response theory (IRT) in which unobserved subjects' values on a latent variable of interest (e.g. intelligence) are inferred from the observed subjects' responses to a set of question items (Hambleton et al., 1991). Central to IRT is the assumption that a subject's value on the latent variable of interest *affects* the subject's responses to the set of question items (Hambleton et al., 1991). For example, a subject's level of intelligence is postulated to influence his/her answers in an intelligence test. In correspondence with this assumption, the consistent mathematical element across all types of IRT models is that the observed subjects' responses are regressed on the unobserved subjects' latent levels of ability.

Here, there are documents rather than subjects and classifiers rather than question items. Yet, the aim is the same: to infer unobserved latent positions from what is observed. As in IRT, the idea here is that a document's value on the latent sentiment variable *affects* the predicted probabilities the document obtains from the classifiers. For example, a document with a highly positive sentiment is assumed to get rather high predicted probabilities from the classifiers. Consequently, the predicted probabilities are regressed on the documents' latent sentiment positions.

In doing so, it has to be accounted for that the predicted probabilities are grouped in a crossed non-nested design: In step 2, for each of the N documents, J predicted probabilities (one from each classifier) are produced such that there are $N \times J$ predicted probabilities. These predicted probabilities cannot be assumed to be independent. The J predicted probabilities for one document are likely to be correlated because they are repeated measurements on the same document. Additionally, the N predicted probabilities produced by one classifier also are generated by a common source. They come

from the same classifier that might systematically differ from the others, e.g. produce systematically lower predicted probabilities.

Moreover, the data generating process is such that the documents are drawn from a larger population of documents. The population distribution of the probability to belong to class 1 might inform the probabilities obtained by individual documents. Similarly, the classifiers are sampled from a population of classifiers with a population distribution in the generated predicted probabilities that may inform an individual classifier's predicted probabilities. To account for this data generating process, a mixed model with N document random intercepts and J classifier random intercepts seems the adequate model of choice. (On mixed models see for example Fahrmeir et al. (2013, chapter 7)).

As the predicted probabilities, $\hat{y}_{ij}$, are in the unit interval [0,1], it is assumed that the $\hat{y}_{ij}$ are beta distributed. Following the parameterization of the beta density employed by Ferrari and Cribari-Neto (2004) the beta mixed model is:

$$\hat{y}_{ij} \sim B(\mu_{ij}, \phi) \tag{1}$$

$$g(\mu_{ij}) = \beta_0 + \theta_i + \gamma_j \tag{2}$$

$$\theta_i \sim N(0, \tau_\theta^2) \tag{3}$$

$$\gamma_j \sim N(0, \tau_\gamma^2) \tag{4}$$

In the model described here, $\hat{y}_{ij}$ (the probability for document i to belong to class 1 as predicted by classifier j) is assumed to be drawn from a beta distribution with conditional mean μ_{ij}. μ_{ij} assumes values in the range (0,1) and $\phi > 0$ is a precision parameter (Cribari-Neto and Zeileis, 2010). μ_{ij} is determined by the fixed global population intercept β_0, the document-specific deviation θ_i from this population intercept, and the classifier-specific deviation γ_j from the population intercept. As the documents are assumed to be sampled from a larger population, the document-specific θ_i are modeled to be drawn from a shared distribution (see equation 3).[3] The same is true for the classifier-specific γ_j. To ensure that the results from the linear predictor in equation 2 are kept between 0 and 1, the logit link is chosen as the link function $g(\cdot)$.[4]

Note that in the beta distribution $Var(\hat{y}_{ij}) = \mu_{ij}(1-\mu_{ij})/(1+\phi)$ (Cribari-Neto and Zeileis, 2010). This means that the variance of $\hat{y}_{ij}$ not only depends on precision parameter ϕ but also depends on μ_{ij}, which implies that the model naturally exhibits heteroscedasticity (Cribari-Neto and Zeileis, 2010). In the given data structure, documents that express very positive (or very negative) sentiments are likely to be easy cases for the classifiers and it is likely that all classifiers will predict very high (or very low) values. Documents that express less extreme sentiments, in contrast, are likely to be more difficult cases and the classifiers are likely to differ more in their predicted probabilities. This is, predicted probabilities are likely to exhibit a higher variance for documents positioned in the middle of the sentiment value range. To additionally account for this effect, the beta mixed model described in equations 1 to 4 can be extended with a dispersion formula describing the precision parameter ϕ as a function of document-specific fixed effects:[5]

$$h(\phi_i) = \delta_i \tag{5}$$

To keep $\phi_i > 0$, $h(\cdot)$ here is the log link (Brooks et al., 2017).[6] In the following, CBMM is implemented with and without the dispersion formula in equation 5. The variant of CBMM that includes equation 5 is denoted CBMMd.

With or without a dispersion formula, the θ_i describe the document-specific deviations from the fixed population mean β_0. Hence, the θ_i —in linear relation to β_0—position the documents on the real line and thus are taken as the CBMM and CBMMd estimates for the continuous sentiment values.

4 Applications

4.1 Data

The effectiveness of CBMM in generating continuous sentiments using binary training data is evaluated on the basis of four data sets:

The Stanford Sentiment Treebank (SST) (Socher et al., 2013) contains sentiment labels for 11,855 sentences [train: 9,645; test: 2,210] taken from movie reviews. Each of the sentences was assigned one out of 25 sentiment score values ranging from highly negative (0) to highly positive (1) by three independent human annotators.

[3] Note that the usually employed assumption is that the random effects are independent and identically distributed according to a normal distribution (Fahrmeir et al., 2013).

[4] Thus, equation 2 is $log(\mu_{ij}/(1-\mu_{ij})) = \beta_0 + \theta_i + \gamma_j$.

[5] Note that the document-specific δ_i are fixed effects that are not modeled to be sampled from a shared population distribution. The reason is that current software implementations of mixed models that use maximum likelihood estimation only allow for inserting fixed effects but no random effects in the dispersion model formula (Brooks et al., 2017).

[6] Thus, equation 5 here is $log(\phi_i) = \delta_i$.

The V-reg data set from the SemEval-2018 Task 1 on "Affect in Tweets" (Mohammad et al., 2018) contains 2,567 tweets [train: 1,630; test: 937] that are likely to be rich in emotion. The tweets' real-valued valence scores are in the range (0,1) and were generated via BWS, whereby each 4-tuple was ranked by four independent coders.

Furthermore, *two data sets from the 2008 American National Election Studies (ANES)* (The American National Election Studies, 2015) are used. The feeling thermometer question, in which participants have to rate on an integer scale ranging from 0 to 100 in how far they feel favorable and warm vs. unfavorable and cold toward parties, is posed regularly in ANES surveys. In the 2008 pre-election survey, participants were additionally asked in open-ended questions to specify what they specifically like and dislike about the Democratic and the Republican Party.[7] Here, the aim is to generate continuous estimates of the sentiments expressed in the answers based on the binarized feeling thermometer scores. For the Democrats there are 1,646 answers [train: 1,097; test: 549]. This data set is named ANES-D. For the Republicans there are 1,523 answers [train: 1,015; test: 508] that make up data set ANES-R. For comparison with the other applications, the true scores from ANES are rescaled by min-max normalization from range [0,100] to [0,1].

To create binary training labels for the CBMM procedure, in all training data sets the fine-grained sentiment values are dichotomized such that the class label for a document is 1 if its score is $\geq$ 0.5 and is 0 otherwise. CBMM's continuous sentiment estimates for the test set documents then are compared to the original fine-grained values. Note that these four data sets are selected for evaluation precisely because they provide fine-grained sentiment scores against which the CBMM estimates can be compared to. In each of the four data sets, the detailed training annotations are the result of resourceful coding processes or—in the case of ANES—lucky coincidences. For example, around 50,000 annotations were made for the V-reg data set that comprises 2,567 tweets (Mohammad et al., 2018). Such resources or coincidences, however, are unlikely to be available for each potentially interesting research question. Thus, whilst these data sets are selected because they come with fine-grained labels that can be used for evaluating CBMM, the settings in which CBMM will be especially valuable are those in which external information that may serve as a granular training input is unavailable and the available amounts of resources are not sufficient for a granular coding of texts.

[7]The survey contains one question asking what the participant likes and a separate question asking what the participant dislikes about a party. For each respondent, the answers to these two questions are concatenated into a single answer.

4.2 Generating Continuous Sentiment Estimates via CBMM

Step 2 of the CBMM procedure consists in training an ensemble of classifiers on the binary training data to then obtain predicted probabilities for the test set documents. Here, for all four applications, a set of 10 pretrained language representation models with the RoBERTa architecture (Liu et al., 2019) are fine-tuned to the binary classification task. The 10 models within one ensemble merely differ regarding their seed value that initializes the optimization process and governs batch allocation.[8] As the seed values are randomly generated, this neatly fits with the assumption encoded in the specified mixed models that classifiers are randomly sampled from a larger population of classifiers. As a Transformer-based model for transfer learning, RoBERTa is likely to yield relatively high prediction performances in text-based supervised learning tasks also if—as is the case for the selected applications—training data sets are small.

In step 3 of CBMM, two different beta mixed models as presented in equations 1 to 5—one model with and the other without a dispersion formula—are estimated. In each mixed model, the estimate for θ_i is taken as the sentiment value predicted for document i.

Steps 1 and 3 of the CBMM procedure are conducted in R (R Core Team, 2020). The beta mixed models are estimated with the R package glmmTMB (Brooks et al., 2017). In step 2, fine-tuning is conducted in Python 3 (Van Rossum and Drake, 2009) making use of PyTorch (Paszke et al., 2019). Pretrained RoBERTa models are accessed via the open-source library provided by HuggingFace's Transformers (Wolf et al., 2020). The source code to replicate the findings is available at https://doi.org/10.6084/m9.figshare.14381825.v1.

[8]The 10 models applied for one application also have the same hyperparameter settings. In all four applications, a grid search across sets of different values for the batch size, the learning rate and the number of epochs is conducted via a 5-fold cross-validation. The hyperparameter setting that exhibits the lowest mean loss across the validation folds and does not suffer from too strong overfitting is selected.

4.3 Evaluation

4.3.1 Comparisons to Other Methods

The sentiment estimates from CBMM and CBMMd are compared to the following methods.

Mean of Predicted Probabilities [Pred-Prob-Mean]. For each document, this procedure simply takes the mean of the predicted probabilities across the ensemble of classifiers: $\hat{\theta}_i = \frac{1}{J}\sum_{j=1}^{J}\hat{y}_{ij}$.

Lexicon-Based Approaches. Two lexicons are made use of. First, the SST provides for each textual feature in the SST corpus a fine-grained human annotated sentiment value that indicates the feature's sentiment in the context of movie reviews. Hence, the SST constitutes an all-encompassing and perfectly tailored lexicon for the SST application and is employed as a lexicon here. Second, the SentiWords lexicon (Gatti et al., 2016), that is based on SentiWordNet (Esuli and Sebastiani, 2006) and contains prior polarity sentiment values for around 155,287 English lemmas, is used. For the SST and the SentiWords lexicons, the sentiment value estimates are generated by computing the arithmetic mean of a document's matched features' values. The procedures here are named SST-Mean and SentiWords-Mean.

Regression approaches, that make use of the true fine-grained sentiment values rather than the binary training data, are also applied. Note that the evaluation results for the regression-based procedures signify the levels of performance that can be achieved *if* one is in the ideal situation and possesses fine-grained training annotations. Hence, the regression approaches constitute a reference point against which the other approaches' performances can be related to.

Here, in all four applications, J = 10 RoBERTa regression models are trained on the training set and then make real-valued predictions for the documents in the test set such that there are J = 10 predictions for each test set document: $\hat{\boldsymbol{z}}_i = [\hat{z}_{i1} \dots \hat{z}_{ij} \dots \hat{z}_{iJ}]$; whereby $\hat{z}_{ij}$ is the real-valued prediction of regression model j for document i. To have a fair comparison to CBMM, the same procedures for aggregating the predicted values are explored. Thus, there are three different aggregation methods. First, the mean of the 10 models' predictions is taken such that the sentiment estimate is: $\hat{\theta}_i = \frac{1}{J}\sum_{j=1}^{J}\hat{z}_{ij}$ [Regr-Mean]. Second and third, a mixed model with and without a dispersion formula is estimated on the basis of the $\hat{z}_{ij}$. The estimates for the θ_i are extracted as the continuous sentiment predictions. Yet, to account for the data generating process of the $\hat{z}_{ij}$, a linear mixed model (LMM)—instead of a beta mixed model—is estimated:

$$\hat{z}_{ij} \sim N(\mu_{ij}, \sigma^2) \tag{6}$$

$$\mu_{ij} = \beta_0 + \theta_i + \gamma_j \tag{7}$$

$$\theta_i \sim N(0, \tau_\theta^2) \tag{8}$$

$$\gamma_j \sim N(0, \tau_\gamma^2) \tag{9}$$

This approach is named Regr-LMM. The LMM with a dispersion formula, Regr-LMMd, additionally has: $h(\sigma_i^2) = \delta_i$; with $h(\cdot)$ being the log link.

4.3.2 Evaluation Metrics

The generated continuous sentiment estimates are evaluated by comparing them to the original granular sentiment labels. Three evaluation metrics are used: the mean absolute error (MAE), the Pearson correlation coefficient r, and Spearman's rank correlation coefficient ρ. The evaluation metrics are selected such that there is a measure of the average absolute distance (MAE) as well as a measure of the linear correlation (r) between the original true sentiment values and the estimated values. Note that Spearman's ρ assesses the correlation between the ranks of the true and the ranks of the estimated values and thus evaluates in how far the order of documents from negative to positive sentiment as produced by the evaluated approaches reflects the order of documents according to the true scores.

4.4 Results

Table 1 presents the evaluation results across all applied data sets. Figure 2 visualizes distributions of the true and estimated sentiment values for the SST data. Across the four employed data sets (each with a different shape of the to be approximated distribution of the true sentiment values) the performance levels vary for all approaches. Yet, the main result remains consistent: the continuous sentiment estimates generated by CBMM correlate similarly with the truth and get only slightly less closer to the truth as the predictions generated by regression approaches that operate on fine-grained training data. At times, CBMM estimates even slightly outperform regression predictions. Hence, researchers that seek to get continuous sentiment estimates but do not have the resources to produce highly detailed training annotations can apply CBMM on binary training labels and thereby obtain estimated continuous sentiments whose performance is likely

	SST			V-reg			ANES-D			ANES-R		
	MAE	r	ρ	MAE	r	ρ	MAE	r	ρ	MAE	r	ρ
SST-Mean	0.190	0.554	0.574	0.171	0.437	0.487	0.242	−0.013	−0.033	0.252	0.059	0.058
SentiWords-Mean	0.201	0.428	0.429	0.177	0.429	0.475	0.254	0.067	0.079	0.289	−0.009	0.005
Regr-Mean	0.099	0.892	0.876	0.090	0.871	0.869	0.195	0.655	0.653	0.191	0.618	0.627
Regr-LMM	0.099	0.892	0.876	0.090	0.871	0.869	0.195	0.655	0.653	0.191	0.618	0.627
Regr-LMMd	0.099	0.892	0.876	0.090	0.872	0.870	0.195	0.655	0.653	0.192	0.618	0.627
Pred-Prob-Mean	0.216	0.859	0.856	0.198	0.804	0.844	0.202	0.646	0.649	0.218	0.624	0.613
CBMM	0.161	0.874	0.856	0.164	0.819	0.842	0.191	0.667	0.648	0.207	0.621	0.613
CBMMd	0.137	0.877	0.856	0.133	0.835	0.844	0.200	0.668	0.649	0.205	0.620	0.612

Table 1: Evaluation Results. For the SST, V-reg, ANES-D, and ANES-R test data sets, the mean absolute error (MAE), the Pearson correlation coefficient r, and Spearman's rank correlation coefficient ρ between the true and the estimated sentiment values are presented. The shading of the cells is a linear function of the approaches' level of performance. The darker the shading, the higher the performance. For computing the MAE, the predicted sentiment values are rescaled via min-max normalization to the range of the true sentiment values.

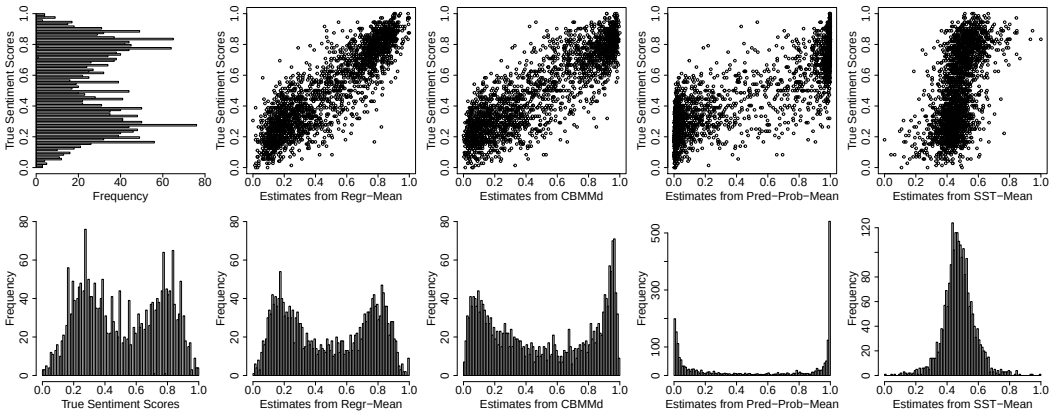


Figure 2: True and Estimated Sentiment Values for the SST Data. First column: Histograms of the true sentiment scores. Remaining columns, top row: Estimates from Regr-Mean, CBMMd, Pred-Prob-Mean, and SST-Mean plotted against the true sentiment values. Remaining columns, bottom row: Histograms of the estimates from Regr-Mean, CBMMd, Pred-Prob-Mean, and SST-Mean.

to be only slightly lower compared to predictions from regression models. Beside this main finding, the following aspects are revealed:

Lexicon-based approaches do not perform very well. The predicted sentiments are centered in the middle of the sentiment value range and changes in a document's sentiment are not strongly reflected in changes in the sentiment values predicted by the lexicons. (As an example see the most right column of Figure 2.) Consequently, the lexicon generated sentiment estimates exhibit relatively low levels of correlation with the true sentiment values. Especially the case of the SST lexicon for the SST data shows that it is not sufficient to have a lexicon that has a coverage of 100% and is perfectly tailored to the context it is applied to. In order to get valid sentiment estimates, one requires an aggregation procedure that accounts for the complex building up of sentiment in texts.

Regression Approaches. The continuous sentiment predictions generated by regression approaches tend to have the smallest distances to and the highest correlations with the true sentiment scores. Hence, the results demonstrate that *if* one has detailed training annotations available that can be treated as if they were continuous, regression approaches constitute an effective way to bring sentiment estimates as close as possible to the true sentiment values.

Across applications, the estimates obtained from Regr-Mean, Regr-LMM, and Regr-LMMd are highly similar. The reason is that the variance for the document-specific intercepts, τ_θ^2, is high relative to the error variance σ^2, and the classifier-

specific variance τ^2_γ.[9] Thus, the LMM estimator is close to a fully unpooled solution in which a separate model for each document is estimated (Fahrmeir et al., 2013, p. 355-356). The sentiment predictions from Regr-LMM are therefore highly correlated with Regr-Mean that computes a separate mean for each document. Furthermore, adding a dispersion formula does not strongly affect the predictions from Regr-LMM.

Pred-Prob-Mean leads to acceptable results. Yet, the estimates from Pred-Prob-Mean still strongly mirror the binary coding structure (see the fourth column of Figure 2). Moreover, MAE tends to decrease and r tends to increase further if the predicted probabilities are aggregated via beta mixed models in CBMM.

CBMM produces continuous sentiment estimates that exhibit performance levels that are relatively close to those of the regression-based procedures. When considering the MAE and r, CBMMd tends to slightly outperform CBMM. As the predicted probabilities across all four data sets are characterized by a high degree of heteroskedasticity[10] additionally accounting for heteroskedasticity via the dispersion formula thus tends to further improve the estimates.

Interestingly, across the three approaches based on predicted probabilities (Pred-Prob-Mean, CBMM, CBMMd) Spearman's ρ nearly remains unchanged. This implies that the predicted order of documents on the latent sentiment variable is largely determined by the predicted probabilities from the ensemble of classifiers. Thus, whilst Pred-Prob-Mean, CBMM and CBMMd operate on the same order of documents,[11] it is the aggregation of the predicted probabilities by a beta mixed model—and the accounting for heteroskedasticity—that enables CBMM and CBMMd to alter the distances between the documents' positions on the sentiment variable such that the distribution of true sentiment values can be approximated more closely. (Compare the histograms of the values predicted by CBMMd and Pred-Prob-Mean in Figure 2.)

[9] Yet, across all evaluated data sets, a Restricted Likelihood-Ratio-Test (based on the approximation presented by Scheipl et al. (2008) as implemented in the RLRsim R-package) testing the null hypothesis that $\tau^2_\gamma = 0$, reveals that this null hypothesis can be rejected at a significance level of 0.01.

[10] To assess heteroskedasticity, Breusch-Pagan Tests (Breusch and Pagan, 1979) are conducted. For all applications and tested linear models, the Breusch-Pagan Test suggests that the null hypothesis of homoskedasticity can be rejected at a significance level of 0.01.

[11] Spearman's ρ between the estimates from Pred-Prob-Mean and CBMMd equals 0.999 across all applications.

5 Conclusion

This work introduced CBMM—a *c*lassifier-based *b*eta *m*ixed *m*odeling technique that generates continuous estimates for texts by estimating a beta mixed model based on predicted probabilities from a set of classifiers. CBMM's central contribution is that it produces continuous output based on binary training input, thereby dispensing the requirement of regression approaches to have (possibly prohibitively costly to create) fine-grained training data. Evaluation results demonstrate that CBMM's continuous estimates perform well and are not far from regression predictions.

CBMM here is applied in the context of sentiment analysis. Yet, it can be applied to any context in which the aim is to have continuous predictions but the resources only allow for creating binary training annotations.

References

Dolores Albarracin, Aashna Sunderrajan, Sophie Lohmann, Man pui Sally Chan, and Duo Jiang. 2019. The psychology of attitudes, motivation, and persuasion. In Dolores Albarracin and Blair T. Johnson, editors, *The Handbook of Attitudes*, pages 3–44. Routledge, New York, NY.

Mahzarin R. Banaji and Larisa Heiphetz. 2010. Attitudes. In Susan T. Fiske, Daniel T. Gilbert, and Gardner Lindzey, editors, *Handbook of Social Psychology*, pages 348–388. John Wiley & Sons, New York, NY.

Trevor S. Breusch and Adrian R. Pagan. 1979. A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47(5):1287–1294.

Mollie E. Brooks, Kasper Kristensen, Koen J. van Benthem, Arni Magnusson, Casper W. Berg, Anders Nielsen, Hans J. Skaug, Martin Mächler, and Benjamin M. Bolker. 2017. glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal*, 9(2):378–400.

John T. Cacioppo, Wendi L. Gardner, and Gary G. Berntson. 1997. Beyond bipolar conceptualizations and measures: The case of attitudes and evaluative space. *Personality and Social Psychology Review*, 1(1):3–25.

Brian Cheang, Bailey Wei, David Kogan, Howey Qiu, and Masud Ahmed. 2020. Language representation models for fine-grained sentiment classification. *arXiv preprint*. arXiv:2005.13619v1 [cs.CL].

Francisco Cribari-Neto and Achim Zeileis. 2010. Beta regression in R. *Journal of Statistical Software*, 34(2):1–24.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Leandre R. Fabringar, Tara K. MacDonald, and Duane T. Wegener. 2019. The origins and structure of attitudes. In Dolores Albarracin and Blair T. Johnson, editors, *The Handbook of Attitudes*, pages 109–157. Routledge, New York, NY.

Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx. 2013. *Regression*. Springer-Verlag, Berlin.

Silvia Ferrari and Francisco Cribari-Neto. 2004. Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31(7):799–815.

Lorenzo Gatti, Marco Guerini, and Marco Turchi. 2016. SentiWords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, 7(4):409–421.

Justin Grimmer and Brandon M. Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21(3):267–297.

Ronald K. Hambleton, Hariharan Swaminathan, and H. Jane Rogers. 1991. *Fundamentals of Item Response Theory*. Sage, Newbury Park, California.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 465–470, Vancouver, Canada. Association for Computational Linguistics.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology*, 2nd edition. Sage Publications, Thousand Oaks.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331.

Bing Liu. 2015. *Sentiment Analysis*. Cambridge University Press, New York.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint*. arXiv:1907.11692v1 [cs.CL].

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 378–382, Borovets, Bulgaria.

Georgios Paltoglou, Mathias Theunis, Arvid Kappas, and Mike Thelwall. 2013. Predicting emotional responses to long informal text. *IEEE Transactions on Affective Computing*, 4(1):106–115.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 115–124, Ann Arbor, Michigan, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Fabian Scheipl, Sonja Greven, and Helmut Küchenhoff. 2008. Size and power of tests for a zero random effect variance or polynomial regression in additive and linear mixed models. *Computational Statistics & Data Analysis*, 52(7):3283–3299.

Jonathan B. Slapin and Sven-Oliver Proksch. 2008. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52(3):705–722.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

The American National Election Studies. 2015. *ANES 2008 Time Series Study*. Inter-University Consortium for Political and Social Research, Ann Arbor, MI. https://electionstudies.org/data-center/2008-time-series-study/.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Tun Thura Thet, Jin-Cheon Na, and Christopher S.G. Khoo. 2010. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 36(6):823–848.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Jin Wang, Bo Peng, and Xuejie Zhang. 2018. Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*, 322:93–101.

Kohei Watanabe. 2021. Latent Semantic Scaling: A semisupervised text analysis technique for new domains and languages. *Communication Methods and Measures*, 15(2):81–102.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint*. arXiv:1910.03771v5 [cs.CL].

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 649–657, Montreal, Canada. MIT Press.

Chapter 5

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv. https://arxiv.org/abs/1603.04467

Abdul Reda, A., Sinanoglu, S., & Abdalla, M. (2021). Mobilizing the masses: Measuring resource mobilization on Twitter. *Sociological Methods & Research*, (pp. 1–40). https://doi.org/10.1177/0049124120986197

Agarwal, S., Jabbari, S., Agarwal, C., Upadhyay, S., Wu, S., & Lakkaraju, H. (2021). Towards the unification and robustness of perturbation and gradient based explanations. *Proceedings of Machine Learning Research*, 139, 110–119. https://proceedings.mlr.press/v139/agarwal21c.html

Agrawal, H., Desai, K., Wang, Y., Chen, X., Jain, R., Johnson, M., Batra, D., Parikh, D., Lee, S., & Anderson, P. (2019). nocaps: Novel object captioning at scale. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, 8947–8956. https://doi.org/10.1109/ICCV.2019.00904

Ahlquist, J. S. & Breunig, C. (2012). Model-based clustering and typologies in the social sciences. *Political Analysis*, *20*(1), 92–112. https://doi.org/10.1093/pan/mpr039

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211. https://doi.org/10.1016/0749-5978(91)90020-T

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In J.-F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi

(Eds.), *Machine Learning: ECML 2004* (pp. 39–50). Springer. https://doi.org/10.1007/978-3-540-30115-8_7

Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In E. M. Bender, L. Derczynski, & P. Isabelle (Eds.), *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 1638–1649). Association for Computational Linguistics. https://aclanthology.org/C18-1139

Alammar, J. (2018a). The illustrated BERT, ELMo, and co. (How NLP cracked transfer learning). *Jay Alammar*. http://jalammar.github.io/illustrated-bert/

Alammar, J. (2018b). The illustrated Transformer. *Jay Alammar*. http://jalammar.github.io/illustrated-transformer/

Alammar, J. (2018c). Visualizing a neural machine translation model (mechanics of seq2seq models with attention). *Jay Alammar*. https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Albarracin, D., Sunderrajan, A., Lohmann, S., Chan, M.-p. S., & Jiang, D. (2019). The psychology of attitudes, motivation, and persuasion. In D. Albarracin & B. T. Johnson (Eds.), *The Handbook of Attitudes* (pp. 3–44). Routledge. https://doi.org/10.4324/9781315178103

ALDayel, A. & Magdy, W. (2021). Stance detection on social media: State of the art and trends. *Information Processing & Management*, *58*(4), 102597. https://doi.org/10.1016/j.ipm.2021.102597

Allaire, J. J., Francois, R., Ushey, K., Vandenbrouck, G., Geelnard, M., & Intel (2020). *RcppParallel: Parallel programming tools for 'Rcpp'*. (Version 5.0.2) [R Package]. CRAN. https://CRAN.R-project.org/package=RcppParallel

ALMasri, M., Berrut, C., & Chevallet, J.-P. (2013). Wikipedia-based semantic query enrichment. In P. N. Bennett, E. Gabrilovich, J. Kamps, & J. Karlgren (Eds.), *Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR '13)* (pp. 5–8). Association for Computing Machinery. https://doi.org/10.1145/2513204.2513209

Alrababa'h, A. & Blaydes, L. (2021). Authoritarian media and diversionary threats: Lessons from 30 years of Syrian state discourse. *Political Science Research and Methods*, *9*(4), 693–708. https://doi.org/10.1017/psrm.2020.28

Amador, J., Collignon-Delmar, S., Benoit, K., & Matsuo, A. (2017). Predicting the Brexit vote by tracking and classifying public opinion using Twitter data. *Statistics, Politics and Policy*, *8*(1), 85–104. https://doi.org/10.1515/spp-2017-0006

Amidi, A. & Amidi, S. (2019). *Recurrent neural networks cheatsheet* [Lecture Notes]. Stanford University. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Amsalem, E., Fogel-Dror, Y., Shenhav, S. R., & Sheafer, T. (2020). Fine-grained analysis of diversity levels in the news. *Communication Methods and Measures*, *14*(4), 266–284. https://doi.org/10.1080/19312458.2020.1825659

Anastasopoulos, L. J. & Bertelli, A. M. (2020). Understanding delegation through machine learning: A method and application to the European Union. *American Political Science Review*, *114*(1), 291–301. https://doi.org/10.1017/S0003055419000522

Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. (2018). Towards better understanding of gradient-based attribution methods for deep neural networks. In Y. Bengio & Y. LeCun (Eds.), *6th International Conference on Learning Representations (ICLR 2018)*. https://openreview.net/pdf?id=Sy21R9JAW

Aribandi, V., Tay, Y., Schuster, T., Rao, J., Zheng, H. S., Mehta, S. V., Zhuang, H., Tran, V. Q., Bahri, D., Ni, J., Gupta, J., Hui, K., Ruder, S., & Metzler, D. (2022). Ext5: Towards extreme multi-task scaling for transfer learning. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=Vzh1BFUCiIX

Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Wu, Y., & Zhu, M. (2013). A practical algorithm for topic modeling with provable guarantees. *Proceedings of Machine Learning Research*, *28*(2), 280–288. https://proceedings.mlr.press/v28/arora13.html

Aßenmacher, M. & Heumann, C. (2020). On the comparability of pre-trained language models. In S. Ebling, D. Tuggener, M. Hürlimann, M. Cieliebak, & M. Volk (Eds.), *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS 2020)*. CEUR-WS.org. http://ceur-ws.org/Vol-2624/paper2.pdf

Aßenmacher, M., Schulze, P., & Heumann, C. (2021). Benchmarking down-scaled (not so large) pre-trained language models. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 14–27). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.2

Azad, H. K. & Deepak, A. (2019). Query expansion techniques for information retrieval: A survey. *Information Processing and Management*, *56*(5), 1698–1735. https://doi.org/10.1016/j.ipm.2019.05.009

Azar, E. E. (2009). *Conflict and Peace Data Bank (COPDAB), 1948-1978*. [Data set]. Inter-University Consortium for Political and Social Research. https://doi.org/10.3886/ICPSR07767.v4

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. arXiv. https://arxiv.org/abs/1607.06450

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., & Auli, M. (2021). XLS-R: Self-supervised cross-lingual speech representation learning at scale. arXiv. https://arxiv.org/abs/2111.09296

Baden, C., Kligler-Vilenchik, N., & Yarchi, M. (2020). Hybrid content analysis: Toward a strategy for the theory-driven, computer-assisted classification of large text corpora. *Communication Methods and Measures*, *14*(3), 165–183. https://doi.org/10.1080/19312458.2020.1803247

Baerg, N. & Lowe, W. (2020). A textual Taylor rule: Estimating central bank preferences combining topic and scaling methods. *Political Science Research and Methods*, *8*(1), 106–122. https://doi.org/10.1017/psrm.2018.31

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 12449–12460). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf

Bagozzi, B. E. & Berliner, D. (2018). The politics of scrutiny in human rights monitoring: Evidence from Structural Topic Models of US State Department human rights reports. *Political Science Research and Methods*, *6*(4), 661–677. https://doi.org/10.1017/psrm.2016.44

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. https://arxiv.org/abs/1409.0473

Banaji, M. R. & Heiphetz, L. (2010). Attitudes. In S. T. Fiske, D. T. Gilbert, & G. Lindzey (Eds.), *Handbook of Social Psychology* (pp. 348–388). John Wiley & Sons. https://doi.org/10.1002/9780470561119

Bapna, A., Chung, Y.-a., Wu, N., Gulati, A., Jia, Y., Clark, J. H., Johnson, M., Riesa, J., Conneau, A., & Zhang, Y. (2021). SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training. arXiv. https://arxiv.org/abs/2110.10329

Barberá, P. (2015). Birds of the same feather tweet together: Bayesian ideal point estimation using Twitter data. *Political Analysis*, *23*(1), 76–91. https://doi.org/10.1093/pan/mpu011

Barberá, P. (2016). Less is more? How demographic sample weights can improve public opinion estimates based on Twitter data. Manuscript. http://pablobarbera.com/static/less-is-more.pdf

Barberá, P., Boydstun, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated text classification of news articles: A practical guide. *Political Analysis*, *29*(1), 19–42. https://doi.org/10.1017/pan.2020.8

Barberá, P., Casas, A., Nagler, J., Egan, P. J., Bonneau, R., Jost, J. T., & Tucker, J. A. (2019). Who leads? Who follows? Measuring issue attention and agenda setting by legislators and the mass public using social media data. *American Political Science Review*, *113*(4), 883–901. https://doi.org/10.1017/S0003055419000352

Barberá, P., Jost, J. T., Nagler, J., Tucker, J. A., & Bonneau, R. (2015a). Tweeting from left to right: Is online political communication more than an echo chamber? *Psychological Science*, *26*(10), 1531–1542. https://doi.org/10.1177/0956797615594620

Barberá, P. & Steinert-Threlkeld, Z. C. (2020). How to use social media data for political science research. In L. Curini & R. Franzese (Eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations* (pp. 404–423). SAGE Publications. https://www.doi.org/10.4135/9781526486387.n26

Barberá, P., Wang, N., Bonneau, R., Jost, J. T., Nagler, J., Tucker, J., & González-Bailón, S. (2015b). The critical periphery in the growth of social protests. *PLoS ONE*, *10*(11), e0143611. https://doi.org/10.1371/journal.pone.0143611

Bauer, P. C., Barberá, P., Ackermann, K., & Venetz, A. (2017). Is the left-right scale a valid measure of ideology? *Political Behavior*, *39*(3), 553–583. https://doi.org/10.1007/s11109-016-9368-2

Baum, M., Cohen, D. K., & Zhukov, Y. M. (2018). Does rape culture predict rape? Evidence from U.S. newspapers, 2000-2013. *Quarterly Journal of Political Science*, *13*(3), 263–289. http://dx.doi.org/10.1561/100.00016124

Bäuml, K.-H. (2007). Making memories unavailable: The inhibitory power of retrieval. *Journal of Psychology*, *215*(1), 4–11. https://doi.org/10.1027/0044-3409.215.1.4

Beauchamp, N. (2017). Predicting and interpolating state-level polls using Twitter textual data. *American Journal of Political Science*, *61*(2), 490–503. https://doi.org/10.1111/ajps.12274

Belinkov, Y. & Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7, 49–72. https://doi.org/10.1162/tacl_a_00254

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document Transformer. arXiv. https://arxiv.org/abs/2004.05150

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155. https://www.jmlr.org/papers/v3/bengio03a.html

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Benoit, K. (2020). Text as data: An overview. In L. Curini & R. Franzese (Eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations* (pp. 461–497). SAGE Publications. https://www.doi.org/10.4135/9781526486387.n29

Benoit, K. & Laver, M. (2006). *Party policy in modern democracies*. Routledge.

Benoit, K., Laver, M., & Mikhaylov, S. (2009). Treating words as data with error: Uncertainty in text statements of policy positions. *American Journal of Political Science*, *53*(2), 495–513. https://doi.org/10.1111/j.1540-5907.2009.00383.x

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Bischl, B., Mersmann, O., Trautmann, H., & Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, *20*(2), 249–275. https://doi.org/10.1162/EVCO_a_00069

Bischl, B. & Molnar, C. (2018). *Introduction to machine learning* [Lecture Notes]. Ludwig-Maximilians-Universität München.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blaydes, L., Grimmer, J., & McQueen, A. (2018). Mirrors for princes and sultans: Advice on the art of governance in the medieval christian and islamic worlds. *The Journal of Politics*, *80*(4), 1150–1167. https://doi.org/10.1086/699246

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1145/2133806.2133826

Blei, D. M. & Lafferty, J. D. (2007). A Correlated Topic Model of science. *The Annals of Applied Statistics*, *1*(1), 17–35. https://doi.org/10.1214/07-AOAS114

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. https://doi.org/10.1162/tacl_a_00051

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., & et al. (2021). On the opportunities and risks of foundation models. arXiv. https://arxiv.org/abs/2108.07258

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)* (pp. 144–152). Association for Computing Machinery. https://doi.org/10.1145/130385.130401

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 632–642). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1075

Branco, P., Torgo, L., & Ribeiro, R. P. (2016a). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, *49*(2), 1–50. https://doi.org/10.1145/2907070

Branco, P., Torgo, L., & Ribeiro, R. P. (2016b). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, *49*(2), 1–50. https://doi.org/10.1145/2907070

Bräuninger, T., Müller, J., & Stecker, C. (2016). Modeling preferences using roll call votes in parliamentary systems. *Political Analysis*, *24*(2), 189–210. https://doi.org/10.1093/pan/mpw006

Breiman, L. (2001a). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L. (2001b). Statistical modeling: The two cultures. *Statistical Science*, *16*(3), 199–215.

Breusch, T. S. & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, *47*(5), 1287–1294. https://doi.org/10.2307/1911963

Brockman, G., Murati, M., Welinder, P., & OpenAI (2020). OpenAI API. *OpenAI*. https://openai.com/blog/openai-api/

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Mächler, M., & Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal*, *9*(2), 378–400. https://doi.org/10.32614/RJ-2017-066

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. arXiv. https://arxiv.org/abs/2005.14165

Brownlee, J. (2017). What is the difference between a parameter and a hyperparameter? *Machine Learning Mastery*. https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/

Brownlee, J. (2019a). A gentle introduction to cross-entropy for machine learning. *Machine Learning Mastery*. https://machinelearningmastery.com/cross-entropy-for-machine-learning/

Brownlee, J. (2019b). How to control the stability of training neural networks with the batch size. *Machine Learning Mastery*. https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/

Brownlee, J. (2020a). Cost-sensitive learning for imbalanced classification. *Machine Learning Mastery*. https://machinelearningmastery.com/cost-sensitive-learning-for-imbalanced-classification/

Brownlee, J. (2020b). Difference between algorithm and model in machine learning. *Machine Learning Mastery*. https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/

Brownlee, J. (2020c). Random oversampling and undersampling for imbalanced classification. *Machine Learning Mastery*. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Brownlee, J. (2021a). Random oversampling and undersampling for imbalanced classification. *Machine Learning Mastery*. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Brownlee, J. (2021b). SMOTE for imbalanced classification with Python. *Machine Learning Mastery*. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Budhwar, A., Kuboi, T., Dekhtyar, A., & Khosmood, F. (2018). Predicting the vote using legislative speech. In A. Zuiderwijk & C. C. Hinnant (Eds.), *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age* (pp. 1–10). Association for Computing Machinery. https://doi.org/10.1145/3209281.3209374

Burnap, P., Gibson, R., Sloan, L., Southern, R., & Williams, M. (2016). 140 characters to victory?: Using Twitter to predict the UK 2015 general election. *Electoral Studies*, 41, 230–233. https://doi.org/10.1016/j.electstud.2015.11.017

Bussell, J. (2020). Shadowing as a tool for studying political elites. *Political Analysis*, *28*(4), 469–486. https://doi.org/10.1017/pan.2020.14

Busso, C., Bulut, M., Lee, C.-C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J. N., Lee, S., & Narayanan, S. S. (2008). IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources & Evaluation*, 42, 335. https://doi.org/10.1007/s10579-008-9076-6

Bustikova, L., Siroky, D. S., Alashri, S., & Alzahrani, S. (2020). Predicting partisan

responsiveness: A probabilistic text mining time-series approach. *Political Analysis*, *28*(1), 47–64. https://doi.org/10.1017/pan.2019.18

Cabrio, E. & Villata, S. (Eds.) (2020). *Proceedings of the 7th Workshop on Argument Mining*. Association for Computational Linguistics. https://aclanthology.org/2020.argmining-1.0

Cacioppo, J. T. & Berntson, G. G. (1994). Relationship between attitudes and evaluative space: A critical review, with emphasis on the separability of positive and negative substrates. *Psychological Bulletin*, *115*(3), 401–423. https://doi.org/10.1037/0033-2909.115.3.401

Cacioppo, J. T., Gardner, W. L., & Berntson, G. G. (1997). Beyond bipolar conceptualizations and measures: The case of attitudes and evaluative space. *Personality and Social Psychology Review*, *1*(1), 3–25. https://doi.org/10.1207/s15327957pspr0101_2

Camacho-Collados, J. & Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, *63*(1), 743–788. https://doi.org/10.1613/jair.1.11259

Card, D., Tan, C., & Smith, N. A. (2018). Neural models for documents with metadata. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2031–2040). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1189

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In P. E. Utgoff (Ed.), *Proceedings of the Tenth International Conference on Machine Learning* (pp. 41–48). Morgan Kaufmann.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28, 41–75. https://doi.org/10.1023/A:1007379606734

Caulfield, B. (2009). What's the difference between a CPU and a GPU? *blogs.nvidia*. https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

Ceron, A., Curini, L., & Iacus, S. M. (2015). Using sentiment analysis to monitor electoral campaigns: Method matters—Evidence from the United States and Italy. *Social Science Computer Review*, *33*(1), 3–20. https://doi.org/10.1177/0894439314521983

Ceron, A., Curini, L., Iacus, S. M., & Porro, G. (2014). Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. *New Media & Society*, *16*(2), 340–358. https://doi.org/10.1177/1461444813480466

Chang, C. & Masterson, M. (2020). Using word order in political text classification with long short-term memory models. *Political Analysis*, *28*(3), 395–411. https://doi.org/10.1017/pan.2019.46

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems 22*. Curran Associates Inc. https://proceedings.neurips.cc/paper/2009/file/f92586a25b b3145facd64ab20fd554ff-Paper.pdf

Chawla, N. V. (2005). Data mining for imbalanced datasets: An overview. In O. Maimon & L. Rokach (Eds.), *The Data Mining and Knowledge Discovery Handbook* (pp. 853–867). Springer.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*(1), 321–357. https://doi.org/10.1613/jair.953

Cheang, B., Wei, B., Kogan, D., Qiu, H., & Ahmed, M. (2020). Language representation models for fine-grained sentiment classification. arXiv. https://arxiv.org/abs/2005.13619

Chen, D. & Manning, C. (2014). A fast and accurate dependency parser using neural networks. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 740–750). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1082

Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In B. Krishnapuram & M. Shah (Eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse Transformers. arXiv. https://arxiv.org/abs/1904.10509

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1179

Chollet, F. (2021). *Deep learning with Python (2nd ed.)*. Manning Publications. https://www.manning.com/books/deep-learning-with-python-second-edition

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

Chomsky, N. (2017). The Galilean challenge: Architecture and evolution of language. *Journal of Physics: Conference Series*, 880, 012015. https://doi.org/10.1088/1742-6596/880/1/012015

Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10, 73–91. https://doi.org/10.1162/tacl_a_00448

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. In T. Linzen, G. Chrupała, Y. Belinkov, & D. Hupkes (Eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (pp. 276–286). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4828

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. In A. Rush (Ed.), *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net. https://openreview.net/forum?id=r1xMH1BtvB

Clark, K. & Luong, T. (2020). More efficient NLP model pre-training with ELECTRA. *Google AI Blog*. https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? Try ARC, the AI2 reasoning challenge. arXiv. https://arxiv.org/abs/1803.05457

Clark, T. S. & Lauderdale, B. (2010). Locating Supreme Court opinions in doctrine space. *American Journal of Political Science*, *54*(4), 871–890. https://doi.org/10.2307/20788775

Clevert, D., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by Exponential Linear Units (ELUs). In Y. Bengio & Y. LeCun (Eds.), *4th International Conference on Learning Representations (ICLR 2016)*. https://arxiv.org/abs/1511.07289

Clinton, J., Jackman, S., & Rivers, D. (2004). The statistical analysis of roll call data. *American Political Science Review*, *98*(2), 355–370. https://doi.org/10.1017/S0003055404001194

Colleoni, E., Rozza, A., & Arvidsson, A. (2014). Echo chamber or public sphere? Predicting political orientation and measuring political homophily in Twitter using big data. *Journal of Communication*, *64*(2), 317–332. https://doi.org/10.1111/jcom.12084

Collobert, R. & Weston, J. (2007). Fast semantic extraction using a novel neural network architecture. In A. Zaenen & A. van den Bosch (Eds.), *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* (pp. 560–567). Association for Computational Linguistics. https://aclanthology.org/P07-1071

Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International*

Conference on Machine Learning (pp. 160–167). Association for Computing Machinery. https://doi.org/10.1145/1390156.1390177

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*(76), 2493–2537. http://jmlr.org/papers/v12/collobert11a.html

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440–8451). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747

Connor, R. J. & Mosimann, J. E. (1969). Concepts of independence for proportions with a generalization of the Dirichlet distribution. *Journal of the American Statistical Association*, *64*(325), 194–206.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Cribari-Neto, F. & Zeileis, A. (2010). Beta regression in R. *Journal of Statistical Software*, *34*(2), 1–24. https://doi.org/10.18637/jss.v034.i02

Csereklyei, Z., Thurner, P. W., Langer, J., & Küchenhoff, H. (2017). Energy paths in the European Union: A model-based clustering approach. *Energy Economics*, 65, 442–457. https://doi.org/10.1016/j.eneco.2017.05.014

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., & Swinton, J. (2019). *xtable: Export tables to LaTeX or HTML*. (Version 1.8-4) [R Package]. CRAN. https://cran.r-project.org/web/packages/xtable/index.html

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. arXiv. https://arxiv.org/abs/1901.02860

Däubler, T. & Benoit, K. (2017). Estimating better left-right positions through statistical scaling of manual content analysis. Manuscript. https://kenbenoit.net/pdfs/text_in_context_2017.pdf

Däubler, T. & Benoit, K. (2021). Scaling hand-coded political texts to learn more about left-right policy content. *Party Politics*, (pp. 1–11). https://doi.org/10.1177/13540688211026076

Davison, J. (2020a). New pipeline for zero-shot text classification. *discuss.huggingface.co*. Retrieved December 28, 2021 from https://discuss.huggingface.co/t/new-pipeline-for-zero-shot-text-classification/681

Davison, J. (2020b). Zero-shot learning in modern NLP. *Joe Davison Blog*. https://joeddav.github.io/blog/2020/05/29/ZSL.html

Denny, M. J. & Spirling, A. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, *26*(2), 168–189. https://doi.org/10.1017/pan.2017.44

Deshwal, A., Doppa, J. R., & Roth, D. (2019). Learning and inference for structured prediction: A unifying perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (pp. 6291–6299). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2019/878

Devlin, J. (2019). Multilingual BERT. *github.com/google-research*. https://github.com/google-research/bert/blob/master/multilingual.md

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. arXiv. https://arxiv.org/abs/1810.04805v1

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Di Cocco, J. & Monechi, B. (2021). How populist are parties? Measuring degrees of populism in party manifestos using supervised machine learning. *Political Analysis*, (pp. 1–17). https://doi.org/10.1017/pan.2021.29

Diaz, F., Mitra, B., & Craswell, N. (2016). Query expansion with locally-trained word embeddings. In K. Erk & N. A. Smith (Eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 367–377). Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1035

Diekmann, A. (2007). *Empirische Sozialforschung (11th ed.)*. Rowohlt.

Diermeier, D., Godbout, J.-F., Yu, B., & Kaufmann, S. (2011). Language and ideology in Congress. *British Journal of Political Science*, *42*(1), 31–55. https://doi.org/10.1017/S0007123411000160

Dietrich, B. J., Hayes, M., & O'Brien, D. Z. (2019). Pitch perfect: Vocal pitch and the emotional intensity of congressional speech. *American Political Science Review*, *113*(4), 941–962. https://doi.org/10.1017/S0003055419000467

Dixon, L. (2017). Hi! and welcome to our first toxicity classification challenge. *kaggle.com*.

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/46064

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv. https://arxiv.org/abs/2002.06305v1

D'Orazio, V., Landis, S. T., Palmer, G., & Schrodt, P. (2014). Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Political Analysis*, *22*(2), 224–242. https://doi.org/10.1093/pan/mpt030

Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv. https://arxiv.org/abs/1702.08608

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*.

Dowle, M. & Srinivasan, A. (2020). *data.table: Extension of 'data.frame'*. (Version 1.13.0) [R Package]. CRAN. https://CRAN.R-project.org/package=data.table

Druckman, J. N., Kifer, M. J., & Parkin, M. (2009). Campaign communications in U.S. congressional elections. *American Political Science Review*, *103*(3), 343–366. https://doi.org/10.1017/S0003055409990037

Druckman, J. N. & Lupia, A. (2000). Preference formation. *Annual Review of Political Science*, 3, 1–24. https://doi.org/10.1146/annurev.polisci.3.1.1

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159. https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

Duong, L., Cohn, T., Bird, S., & Cook, P. (2015). Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 845–850). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-2139

Durrell, M. (2008). Linguistic variable - linguistic variant. In U. Ammon, N. Dittmar, K. J. Mattheier, & P. Trudgill (Eds.), *Sociolinguistics* (pp. 195–200). De Gruyter Mouton. https://doi.org/10.1515/9783110141894.1.2.195

Duthie, R. & Budzynska, K. (2018). A deep modular RNN approach for ethos mining. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* (pp. 4041–4047). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2018/562

Eagly, A. H. & Chaiken, S. (1993). *The psychology of attitudes*. Harcourt Brace Jovanovich College Publishers.

Eagly, A. H. & Chaiken, S. (2007). The advantages of an inclusive definition of attitude. *Social Cognition*, *25*(5), 582–602. https://doi.org/10.1521/soco.2007.25.5.582

Efron, B. & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall.

Egami, N., Fong, C. J., Grimmer, J., Roberts, M. E., & Stewart, B. M. (2018). How to make causal inferences using texts. arXiv. https://arxiv.org/abs/1802.02163

Ein-Dor, L., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., & Slonim, N. (2020). Active learning for BERT: An empirical study. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7949–7962). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.638

Eisenstein, J., O'Connor, B., Smith, N. A., & Xing, E. P. (2010). A latent variable model for geographic lexical variation. In H. Li & L. Màrquez (Eds.), *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1277–1287). Association for Computational Linguistics. https://aclanthology.org/D10-1124

Elff, M. (2013). A dynamic state-space model of coded political texts. *Political Analysis*, *21*(2), 217–232. https://doi.org/10.1093/pan/mps042

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)* (pp. 973–978). Morgan Kaufmann Publishers Inc.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1016/0364-0213(90)90002-E

Enamorado, T., López-Moctezuma, G., & Ratkovic, M. (2021). Scaling data from multiple sources. *Political Analysis*, *29*(2), 212–235. https://doi.org/10.1017/pan.2020.24

Ennser-Jedenastik, L. & Meyer, T. M. (2018). The impact of party cues on manual coding of political texts. *Political Science Research and Methods*, *6*(3), 625–633. https://doi.org/10.1017/psrm.2017.29

Erlich, A., Dantas, S. G., Bagozzi, B. E., Berliner, D., & Palmer-Rubin, B. (2021). Multi-label prediction for political text-as-data. *Political Analysis*, (pp. 1–18). https://doi.org/10.1017/pan.2021.15

Ertekin, S., Huang, J., Bottou, L., & Giles, L. (2007). Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM '07)* (pp. 127–136). Association for Computing Machinery. https://doi.org/10.1145/1321440.1321461

Eshima, S., Imai, K., & Sasaki, T. (2021). Keyword assisted topic models. arXiv. https://arxiv.org/abs/2004.05964v2

Esuli, A. & Sebastiani, F. (2006). SentiWordNet: A publicly available lexical resource for opinion mining. In N. Calzolari, A. Gangemi, B. Maegaard, J. Mariani, J. Odijk, & D. Tapias (Eds.), *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2006/pdf/384_pdf.pdf

Fabringar, L. R., MacDonald, T. K., & Wegener, D. T. (2019). The origins and structure of attitudes. In D. Albarracin & B. T. Johnson (Eds.), *The Handbook of Attitudes* (pp. 109–157). Routledge. https://doi.org/10.4324/9781315178103

Fahrmeir, L., Kneib, T., Lang, S., & Marx, B. (2013). *Regression*. Springer-Verlag. https://doi.org/10.1007/978-3-642-34333-9

Ferrari, S. & Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, *31*(7), 799–815. https://doi.org/10.1080/02664760420002 14501

Firat, A. (2016). Unified framework for quantification. arXiv. https://arxiv.org/abs/1606.00868

Firth, J. R. (1957). *Studies in linguistic analysis*. Publications of the Philological Society. Blackwell.

Fogel-Dror, Y., Shenhav, S. R., Sheafer, T., & Atteveldt, W. V. (2019). Role-based association of verbs, actions, and sentiments with entities in political discourse. *Communication Methods and Measures*, *13*(2), 69–82. https://doi.org/10.1080/19312458.2018.1536973

Fong, C. & Tyler, M. (2021). Machine learning predictions as regression covariates. *Political Analysis*, *29*(4), 467–484. https://doi.org/10.1017/pan.2020.38

Fowler, E. F., Franz, M. M., Martin, G. J., Peskowitz, Z., & Ridout, T. N. (2021). Political advertising online and offline. *American Political Science Review*, *115*(1), 130–149. https://doi.org/10.1017/S0003055420000696

Freidank, M. (2020). Reproducibility issue with Transformers (BERT) and TF2.2. *github.com/NVIDIA*. Retrieved January 20, 2022 from https://github.com/NVIDIA/framework-determinism/issues/19

Fu, T.-J., Li, L., Gan, Z., Lin, K., Wang, W. Y., Wang, L., & Liu, Z. (2021). VIOLET: End-to-end video-language Transformers with masked visual-token modeling. arXiv. https://arxiv.org/abs/2111.12681

Gabel, M. J. & Huber, J. D. (2000). Putting parties in their place: Inferring party left-right ideological positions from party manifestos data. *American Journal of Political Science*, *44*(1), 94–103. https://doi.org/10.2307/2669295

Galassi, A., Lippi, M., & Torroni, P. (2021). Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(10), 4291–4308. https://doi.org/10.1109/TNNLS.2020.3019893

Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 3816–3830). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.295

Gatt, A. & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61, 65–170. https://doi.org/10.1613/jair.5477

Gatti, L., Guerini, M., & Turchi, M. (2016). SentiWords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, *7*(4), 409–421. https://doi.org/10.1109/TAFFC.2015.2476456

Gawronski, B. & Brannon, S. M. (2019). Attitudes and the implicit-explicit dualism. In D. Albarracin & B. T. Johnson (Eds.), *The Handbook of Attitudes* (pp. 158–196). Routledge. https://doi.org/10.4324/9781315178103

Gessler, T. & Hunger, S. (2021). How the refugee crisis and radical right parties shape party competition on immigration. *Political Science Research and Methods*, (pp. 1–21). https://doi.org/10.1017/psrm.2021.64

Ghahramani, Z. (2015). *The Machine Learning Summer School. Lecture: Bayesian inference* [Lecture Notes]. Max Planck Institute for Intelligent Systems. http://mlss.tuebingen.mpg.de/2015/slides/ghahramani/gp-neural-nets15.pdf

Glavaš, G., Nanni, F., & Ponzetto, S. P. (2017). Cross-lingual classification of topics in political texts. In D. Hovy, S. Volkova, D. Bamman, D. Jurgens, B. O'Connor, O. Tsur, & A. S. Doğruöz (Eds.), *Proceedings of the Second Workshop on NLP and Computational Social Science* (pp. 42–46). Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-2906

Glavaš, G., Nanni, F., & Ponzetto, S. P. (2019). Computational analysis of political texts: Bridging research efforts across communities. In P. Nakov & A. Palmer (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts* (pp. 18–23). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-4004

Göbel, S. & Munzert, S. (2018). Political advertising on the Wikipedia marketplace of information. *Social Science Computer Review*, *36*(2), 157–175. https://doi.org/10.1177/0894439317703579

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*(1), 345–420. https://jair.org/index.php/jair/article/download/11030/26198/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org

Google Colaboratory (2020). Google Colaboratory Frequently Asked Questions. *research.google.com/colaboratory*. https://research.google.com/colaboratory/faq.html

Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A., & Bojanowski, P. (2021). Self-supervised pretraining of visual features in the wild. arXiv. https://arxiv.org/abs/2103.01988

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., & Parikh, D. (2017). Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, (pp. 6904–6913). https://openaccess.thecvf.com/content_cvpr_2017/papers/Goyal_Making_the_v_CVPR_2017_paper.pdf

Graves, A. (2008). *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technische Universität München. https://mediatum.ub.tum.de/doc/1289309/401810.pdf

Graves, A. (2014). Generating sequences with recurrent neural networks. arXiv. https://arxiv.org/abs/1308.0850

Greene, K. T., Park, B., & Colaresi, M. (2019). Machine learning human rights and wrongs: How the successes and failures of supervised learning algorithms can inform the debate about information effects. *Political Analysis*, *27*(2), 223–230. https://doi.org/10.1017/pan.2018.11

Griffiths, T. L. & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(suppl 1), 5228–5235. https://doi.org/10.1073/pnas.0307752101

Grimmer, J. (2010). A Bayesian hierarchical topic model for political texts: Measuring expressed agendas in Senate press releases. *Political Analysis*, *18*(1), 1–35. https://doi.org/10.1093/pan/mpp034

Grimmer, J. (2013). Appropriators not position takers: The distorting effects of electoral incentives on Congressional representation. *American Journal of Political Science*, *57*(3), 624–642. https://doi.org/10.1111/ajps.12000

Grimmer, J. & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Grosse, R. (2020a). *CSC321 Neural networks and machine learning (UTM). Lecture 3: Linear classification* [Lecture Notes]. University of Toronto. https://www.cs.toronto.edu/~lczhang/321/notes/notes03.pdf

Grosse, R. (2020b). *CSC321 Neural networks and machine learning (UTM). Lecture 4: Training a classifier* [Lecture Notes]. University of Toronto. https://www.cs.toronto.edu/~lczhang/321/notes/notes04.pdf

Grosse, R. (2020c). *CSC321 Neural networks and machine learning (UTM). Lecture 9: Generalization* [Lecture Notes]. University of Toronto. https://www.cs.toronto.edu/~lczhang/321/notes/notes09.pdf

Grün, B. & Hornik, K. (2011). topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, *40*(13), 1–30. https://doi.org/10.18637/jss.v040.i13

Habernal, I., Wachsmuth, H., Gurevych, I., & Stein, B. (2018). The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1930–1940). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1175

Haim, M. & Hoven, E. (2022). Hate speech's double damage: A semi-automated approach toward direct and indirect targets. *Journal of Quantitative Description: Digital Media*, 2. https://doi.org/10.51685/jqd.2022.009

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. SAGE Publications.

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques (3rd ed.)*. Elsevier.

Han, R., Gill, M., Spirling, A., & Cho, K. (2018). Conditional word embedding and hypothesis testing via Bayes-by-backprop. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 4890–4895). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1527

Hansen, C. (2019). Activation functions explained - GELU, SELU, ELU, ReLU and more. *Machine Learning From Scratch*. https://mlfromscratch.com/activation-functions-explained/

Hare, C. & Poole, K. T. (2014). The polarization of contemporary American politics. *Polity*, *46*(3), 411–429. https://doi.org/10.1057/pol.2014.10

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction (2nd ed.)*. Springer. https://doi.org/10.1007/978-0-387-84858-7

Hayes, A. F. & Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, *1*(1), 77–89. https://doi.org/10.1080/19312450709336664

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. arXiv. https://arxiv.org/abs/1512.03385

Henderson, M., Budzianowski, P., Casanueva, I., Coope, S., Gerz, D., Kumar, G., Mrkšić, N., Spithourakis, G., Su, P.-H., Vulić, I., & Wen, T.-H. (2019). A repository of conversational datasets. In Y.-N. Chen, T. Bedrax-Weiss, D. Hakkani-Tur, A. Kumar, M. Lewis, T.-M. Luong, P.-H. Su, & T.-H. Wen (Eds.), *Proceedings of the First Workshop on NLP for Conversational AI* (pp. 1–10). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4101

Henderson, M., Thomson, B., & Williams, J. D. (2014). The second dialog state tracking challenge. In K. Georgila, M. Stone, H. Hastie, & A. Nenkova (Eds.), *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)* (pp. 263–272). Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-4337

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020). Measuring massive multitask language understanding. arXiv. https://arxiv.org/abs/2009.03300

Hendrycks, D. & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). arXiv. https://arxiv.org/abs/1606.08415

Herrmann, M. & Döring, H. (2021). Party positions from Wikipedia classifications of party ideology. *Political Analysis*, (pp. 1–20). https://doi.org/10.1017/pan.2021.28

Herzog, A. & Benoit, K. (2015). The most unkindest cuts: Speaker selection and expressed government dissent during economic crisis. *The Journal of Politics*, *77*(4), 1157–1175. https://doi.org/10.1086/682670

Heumann, C., Schomaker, M., & Shalabh (2016). *Introduction to statistics and data analysis*. Springer. https://doi.org/10.1007/978-3-319-46162-5

Hewitt, J. & Manning, C. D. (2019). A structural probe for finding syntax in word representations. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4129–4138). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1419

Hinton, G., Srivastava, N., & Swerky, K. (2012). *Neural networks for machine learning. Lecture 6a: Overview of mini-batch gradient descent* [Lecture Notes]. Coursera. http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf

Hix, S., Noury, A., & Roland, G. (2006). Dimensions of politics in the European Parliament.

American Journal of Political Science, *50*(2), 494–511. https://doi.org/10.1111/j.1540-5907.2006.00198.x

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, *81*(396), 945–960. https://doi.org/10.2307/2289064

Hopkins, D. J. & King, G. (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, *54*(1), 229–247. https://doi.org/10.1111/j.1540-5907.2009.00428.x

Howard, J. & Ruder, S. (2018). Universal language model fine-tuning for text classification. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (pp. 328–339). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1031

Hu, M., Peng, Y., Huang, Z., Li, D., & Lv, Y. (2019). Open-domain targeted sentiment analysis via span-based extraction and classification. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 537–546). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1051

Huang, J., Chang, Y., & Cheng, X. (Eds.) (2020a). *SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery. https://doi.org/10.1145/3397271

Huang, L., Perry, P. O., & Spirling, A. (2020b). A general model of author "style" with application to the UK House of Commons, 1935–2018. *Political Analysis*, *28*(3), 412–434. https://doi.org/10.1017/pan.2019.49

Hugging Face (2018). PyTorch BERT model. *github.com/huggingface*. https://github.com/huggingface/transformers/blob/v4.15.0/src/transformers/models/bert/modeling_bert.py

Hugging Face (2020a). Preprocessing. *huggingface.co*. https://huggingface.co/transformers/preprocessing.html

Hugging Face (2020b). Summary of the models. *huggingface.co*. https://huggingface.co/transformers/model_summary.html

Hugging Face (2020c). Tokenizer summary. *huggingface.co*. https://huggingface.co/transformers/tokenizer_summary.html

Hugging Face (2021). Dataset card for reuters21578. *huggingface.co*. https://huggingface.co/datasets/reuters21578

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hutchins, W. J. (2004). The Georgetown-IBM experiment demonstrated in January 1954. In R. E. Frederking & K. B. Taylor (Eds.), *Machine Translation: From Real Users to Research* (pp. 102–114). Springer. https://doi.org/10.1007/978-3-540-30194-3_12

İrsoy, O. & Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 720–728). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1080

Iyyer, M., Enns, P., Boyd-Graber, J., & Resnik, P. (2014). Political ideology detection using recursive neural networks. In K. Toutanova & H. Wu (Eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1113–1122). Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1105

Jackman, S. (2008). Measurement. In J. M. Box-Steffensmeier, H. E. Brady, & D. Collier (Eds.), *The Oxford Handbook of Political Methodology* (pp. 119–151). Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199286546.001.0001

Jacovi, A. & Goldberg, Y. (2020). Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4198–4205). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.386

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., Henaff, O. J., Botvinick, M., Zisserman, A., Vinyals, O., & Carreira, J. (2022). Perceiver IO: A general architecture for structured inputs & outputs. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=fILj7WpI-g

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., & Carreira, J. (2021). Perceiver: General perception with iterative attention. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 4651–4664). PMLR. https://proceedings.mlr.press/v139/jaegle21a.html

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning with applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7

Jawahar, G., Sagot, B., & Seddah, D. (2019). What does BERT learn about the structure of language? In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3651–3657). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1356

Jerzak, C. T., King, G., & Strezhnev, A. (2022). An improved method of automated nonparametric content analysis for social science. *Political Analysis*, (pp. 1–17). https://doi.org/10.1017/pan.2021.36

Jiang, J. (2008). A literature survey on domain adaptation of statistical classifiers. Manuscript. http://www.mysmu.edu/faculty/jingjiang/papers/da_survey.pdf

Jigsaw/Conversation AI (2018). Toxic comment classification challenge. *kaggle.com*. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

Jin, W. & Ho, H. H. (2009). A novel lexicalized HMM-based learning framework for web opinion mining. In A. Danyluk (Ed.), *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 465–472). Association for Computing Machinery. https://doi.org/10.1145/1553374.1553435

Johnson, J. (2017). *CS231n: Convolutional neural networks for visual recognition. Derivatives, backpropagation, and vectorization* [Lecture Notes]. http://cs231n.stanford.edu/handouts/derivatives.pdf

Joshi, P., Santy, S., Budhiraja, A., Bali, K., & Choudhury, M. (2020). The state and fate of linguistic diversity and inclusion in the NLP world. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 6282–6293). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.560

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., & Bennamoun, M. (2022). Hands-on Bayesian neural networks – A tutorial for deep learning users. arXiv. https://arxiv.org/abs/2007.06823

Jungherr, A., Posegga, O., & An, J. (2022). Populist supporters on reddit: A comparison of content and behavioral patterns within publics of supporters of Donald Trump and Hillary Clinton. *Social Science Computer Review*, *40*(3), 809–830. https://doi.org/10.1177/0894439321996130

Jungherr, A., Schoen, H., & Jürgens, P. (2016). The mediation of politics through Twitter: An analysis of messages posted during the campaign for the German federal election 2013. *Journal of Computer-Mediated Communication*, *21*(1), 50–68. https://doi.org/10.1111/jcc4.12143

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In K. Toutanova & H. Wu (Eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 655–665). Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1062

Katagiri, A. & Min, E. (2019). The credibility of public and private signals: A document-based approach. *American Political Science Review*, *113*(1), 156–172. https://doi.org/10.1017/S0003055418000643

Kaufman, A. R. & Klevs, A. (2021). Adaptive fuzzy string matching: How to merge datasets with only one (messy) identifying field. *Political Analysis*, (pp. 1–7). https://doi.org/10.1017/pan.2021.38

Kavlakoglu, E. (2020). AI vs. machine learning vs. deep learning vs. neural networks: What's the difference? *IBM Cloud Blog*. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Kellstedt, P. M. & Whitten, G. D. (2009). *The fundamentals of political science research*. Cambridge University Press. https://doi.org/10.1017/9781108131704

Kentaro, W. (2020). *gdown: Download a large file from Google Drive*. [Python package]. github.com/wkentaro. https://github.com/wkentaro/gdown

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv. https://arxiv.org/abs/1609.04836

Kim, I. S. (2017). Political cleavages within industry: Firm-level lobbying for trade liberalization. *American Political Science Review*, *111*(1), 1–20. https://doi.org/10.1017/S0003055416000654

Kim, I. S. & Kunisky, D. (2021). Mapping political communities: A statistical analysis of lobbying networks in legislative politics. *Political Analysis*, *29*(3), 317–336. https://doi.org/10.1017/pan.2020.29

Kim, J., Griggs, E., Kim, I. S., & Oh, A. (2021). Learning bill similarity with annotated and augmented corpora of bills. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 10048–10064). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.787

Kim, Y. (2014). Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181

King, G. (1998). *Unifying political methodology: The likelihood theory of statistical inference*. The University of Michigan Press. https://doi.org/10.3998/mpub.23784

King, G., Lam, P., & Roberts, M. E. (2017). Computer-assisted keyword and document set discovery from unstructured text. *American Journal of Political Science*, *61*(4), 971–988. https://doi.org/10.1111/ajps.12291

King, G., Pan, J., & Roberts, M. E. (2013). How censorship in China allows government criticism but silences collective expression. *American Political Science Review*, *107*(2), 326–343. https://doi.org/10.1017/S0003055413000014

King, K., Keohane, R. O., & Verba, S. (1994). *Designing social inquiry: Scientific inference in qualitative research*. Princeton University Press.

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations (ICLR 2015)*. http://arxiv.org/abs/1412.6980

Kingma, D. P. & Welling, M. (2014). Auto-encoding variational bayes. In Y. Bengio & Y. LeCun (Eds.), *International Conference on Learning Representations (ICLR 2014)*. https://arxiv.org/abs/1312.6114

Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Y. Bengio & Y. LeCun (Eds.), *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

Kiritchenko, S. & Mohammad, S. (2017). Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In R. Barzilay & M.-Y. Kan (Eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 465–470). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-2074

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient Transformer. arXiv. https://arxiv.org/abs/2001.04451

Klüver, H. (2009). Measuring interest group influence using quantitative text analysis. *European Union Politics*, *10*(4), 535–549. https://doi.org/10.1177/1465116509346782

Klüver, H. & Bäck, H. (2019). Coalition agreements, issue attention, and cabinet governance. *Comparative Political Studies*, *52*(13–14), 1995–2031. https://doi.org/10.1177/0010414019830726

Knight, K., Badarau, B., Baranescu, L., Bonial, C., Bardocz, M., Griffitt, K., Hermjakob, U., Marcu, D., Palmer, M., O'Gorman, T., & Schneider, N. (2020). *Abstract meaning representation (AMR) Annotation release 3.0*. [Data set]. Linguistic Data Consortium. https://doi.org/10.35111/44cy-bp51

Kobayashi, G., Kuribayashi, T., Yokoi, S., & Inui, K. (2020). Attention is not only a weight: Analyzing Transformers with vector norms. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7057–7075). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.574

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., & Grefenstette, E. (2018). The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6, 317–328. https://doi.org/10.1162/tacl_a_00023

Koh, A. & Boey, D. K. S. (2021). Predicting policy domains and preferences with BERT and convolutional neural networks. Manuscript. https://hertie-data-science-lab.github.io/student-projects/posts/predicting-policy-domains-and-preferences-with-bert-and-convolutional-neural-networks/

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., & Reblitz-Richardson, O. (2020). Captum: A unified and generic model interpretability library for PyTorch. arXiv. https://arxiv.org/abs/2009.07896

Koo, T., Carreras, X., & Collins, M. (2008). Simple semi-supervised dependency parsing. In J. D. Moore, S. Teufel, J. Allan, & S. Furui (Eds.), *Proceedings of the Annual Meeting of the Association for Computational Linguistics 2008: Human Language Technologies* (pp. 595–603). Association for Computational Linguistics. https://aclanthology.org/P08-1068

Kouw, W. M. & Loog, M. (2019). A review of domain adaptation without target labels. arXiv. https://arxiv.org/abs/1901.05335

Kozareva, Z., Ravi, S., Vlachos, A., Agrawal, P., & Martins, A. (Eds.) (2021). *Proceedings of the 5th Workshop on Structured Prediction for NLP (SPNLP 2021)*. Association for Computational Linguistics. https://aclanthology.org/2021.spnlp-1.0

Kozlowski, A. C., Taddy, M., & Evans, J. A. (2019). The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review*, *84*(5), 905–949. https://doi.org/10.1177/0003122419877135

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology (2nd ed.)*. SAGE Publications.

Krippendorff, K. (2013). *Content analysis: An introduction to its methodology (3rd ed.)*. SAGE Publications.

Krippendorff, K. & Craggs, R. (2016). The reliability of multi-valued coding of data. *Communication Methods and Measures*, *10*(4), 181–198. https://doi.org/10.1080/19312458.2016.1228863

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (pp. 1097–1105). Curran Associates Inc. https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Küçük, D. & Can, F. (2020). Stance detection: A survey. *ACM Computing Surveys*, *53*(1), 1–37. https://doi.org/10.1145/3369026

Kuhn, M. & Johnson, K. (2013). *Applied predictive modeling.* Springer. https://doi.org/10.1007/978-1-4614-6849-3

Kumar, E. (2011). *Natural language processing.* I. K. International Publishing House Pvt. Ltd.

Kumar, S., Jat, S., Saxena, K., & Talukdar, P. (2019). Zero-shot word sense disambiguation using sense definition embeddings. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 5670–5681). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1568

Kuzi, S., Shtok, A., & Kurland, O. (2016). Query expansion using word embeddings. In S. Mukhopadhyay & C. Zhai (Eds.), *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)* (pp. 1929–1932). Association for Computing Machinery. https://doi.org/10.1145/2983323.2983876

Kwon, K. H., Priniski, J. H., & Chadha, M. (2018). Disentangling user samples: A supervised machine learning approach to proxy-population mismatch in Twitter research. *Communication Methods and Measures*, *12*(2-3), 216–237. https://doi.org/10.1080/19312458.2018.1430755

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C. E. Brodley & A. Danyluk (Eds.), *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 282–289). Morgan Kaufmann Publishers Inc. https://doi.org/10.5555/645530.655813

Lai, G., Xie, Q., Liu, H., Yang, Y., & Hovy, E. (2017). RACE: Large-scale reading comprehension dataset from examinations. arXiv. https://arxiv.org/abs/1704.04683

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In K. Knight, A. Nenkova, & O. Rambow (Eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 260–270). Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1030

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In A. Rush (Ed.), *8th International Conference on Learning Representations (ICLR 2020).* https://openreview.net/pdf?id=H1eA7AEtvS

Lauderdale, B. E. & Herzog, A. (2016). Measuring political positions from legislative speech. *Political Analysis*, *24*(3), 374–394. https://doi.org/10.1093/pan/mpw017

Laver, M., Benoit, K., & Garry, J. (2003). Extracting policy positions from political texts using words as data. *American Political Science Review*, *97*(2), 311–331. https://doi.org/10.1017/S0003055403000698

Lavrenko, V. & Croft, W. B. (2001). Relevance based language models. In D. H. Kraft, W. B. Croft, D. J. Harper, & J. Zobel (Eds.), *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)* (pp. 120–127). Association for Computing Machinery. https://doi.org/10.1145/383952.383972

Lawrence, J. & Reed, C. (2020). Argument mining: A survey. *Computational Linguistics*, *45*(4), 765–818. https://doi.org/10.1162/coli_a_00364

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. *Proceedings of Machine Learning Research*, *32*(2), 1188–1196. http://proceedings.mlr.press/v32/le14.html

LeCun, Y. & Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*. MIT Press.

Lei, T. (2021). When attention meets fast recurrence: Training language models with reduced compute. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7633–7648). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.602

Leitgeb, H. (2017). *The stability of belief. How rational belief coheres with probability*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198732631.001.0001

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1–5. http://jmlr.org/papers/v18/16-365.html

Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The Winograd schema challenge. In S. A. M. Gerhard Brewka, Thomas Eiter (Ed.), *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning* (pp. 552–561). AAAI Press. https://dl.acm.org/doi/10.5555/3031843.3031909

Lewis, D. D. (1997). *Reuters-21578 (Distribution 1.0)*. [Data set]. http://www.daviddlewis.com/resources/testcollections/reuters21578/

Lewis, D. D. & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In B. W. Croft & C. J. van Rijsbergen (Eds.), *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)* (pp. 3–12). Springer. https://dl.acm.org/doi/10.5555/188490.188495

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V.,

& Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7871–7880). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Li, F.-F., Krishna, R., & Xu, D. (2020a). *CS231n: Convolutional neural networks for visual recognition. Neural networks I* [Lecture Notes]. Stanford University. https://cs231n.github.io/neural-networks-1/

Li, F.-F., Krishna, R., & Xu, D. (2020b). *CS231n: Convolutional neural networks for visual recognition. Optimization I* [Lecture Notes]. Stanford University. https://cs231n.github.io/optimization-1/

Li, F.-F., Krishna, R., & Xu, D. (2020c). *CS231n: Convolutional neural networks for visual recognition. Optimization II* [Lecture Notes]. Stanford University. https://cs231n.github.io/optimization-2/

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., & Gonzalez, J. (2020d). Train big, then compress: Rethinking model size for efficient training and inference of Transformers. *Proceedings of Machine Learning Research*, 119, 5958–5968. http://proceedings.mlr.press/v119/li20m.html

Linder, F. (2017). Reducing bias in online text datasets: Query expansion and active learning for better data from keyword searches. SSRN. http://dx.doi.org/10.2139/ssrn.3026393

Little, R. J. A. & Rubin, D. B. (2002). *Statistical analysis with missing data (2nd ed.)*. Wiley. https://doi.org/10.1002/9781119013563

Liu, A. T., wen Yang, S., Chi, P.-H., chun Hsu, P., & yi Lee, H. (2020). Mockingjay: Unsupervised speech representation learning with deep bidirectional Transformer encoders. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*. IEEE. https://doi.org/10.1109/icassp40776.2020.9054458

Liu, B. (2015). *Sentiment analysis*. Cambridge University Press. https://doi.org/10.1017/CBO9781139084789

Liu, H., Simonyan, K., & Yang, Y. (2019a). DARTS: Differentiable architecture search. In T. Sainath (Ed.), *7th International Conference on Learning Representations, ICLR 2019*. https://openreview.net/forum?id=S1eYHoC5FX

Liu, J., Chen, J., & Ye, J. (2009). Large-scale sparse logistic regression. In J. Elder & F. S. Fogelman (Eds.), *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 547–556). Association for Computing Machinery. https://doi.org/10.1145/1557019.1557082

Liu, X. & Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 186–193). Association for Computing Machinery. https://doi.org/10.1145/1008992.1009026

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019b). RoBERTa: A robustly optimized BERT pretraining approach. arXiv. https://arxiv.org/abs/1907.11692

Lo, J., Proksch, S.-O., & Slapin, J. B. (2016). Ideological clarity in multiparty competition: A new measure and test using election manifestos. *British Journal of Political Science*, *46*(3), 591–610. https://doi.org/10.1017/S0007123414000192

Loshchilov, I. & Hutter, F. (2019). Decoupled weight decay regularization. In T. Sainath (Ed.), *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net. https://openreview.net/forum?id=Bkg6RiCqY7

Louis, A. (2020). A brief history of natural language processing – Part 1. *Medium.com*. https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbcb937ebce

Lowe, W. & Benoit, K. (2013). Validating estimates of latent traits from textual data using human judgment as a benchmark. *Political Analysis*, *21*(3), 298–313. https://doi.org/10.1093/pan/mpt002

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

Lucas, C., Nielsen, R. A., Roberts, M. E., Stewart, B. M., Storer, A., & Tingley, D. (2015). Computer-assisted text analysis for comparative politics. *Political Analysis*, *23*(2), 254–277. https://doi.org/10.1093/pan/mpu019

Luo, H., Ji, L., Shi, B., Huang, H., Duan, N., Li, T., Li, J., Bharti, T., & Zhou, M. (2020). UniVL: A unified video and language pre-training model for multimodal understanding and generation. arXiv. https://arxiv.org/abs/2002.06353

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1412–1421). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1166

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In D. Lin, Y. Matsumoto, & R. Mihalcea (Eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:*

Human Language Technologies - Volume 1 (pp. 142–150). Association for Computational Linguistics. https://www.aclweb.org/anthology/P11-1015

MacCartney, B. (2014). *Understanding natural language understanding* [Presentation]. ACM SIGAI Bay Area Chapter Inaugural Meeting. https://nlp.stanford.edu/~wcmac/papers/20140716-UNLU.pdf

Magyar, Z. B. (2022). What makes party systems different? A principal component analysis of 17 advanced democracies 1970–2013. *Political Analysis*, *30*(2), 250–268. https://doi.org/10.1017/pan.2021.21

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., Pfetsch, B., Heyer, G., Reber, U., Häussler, T., Schmid-Petri, H., & Adam, S. (2018). Applying LDA topic modeling in communication research: Toward a valid and reliable methodology. *Communication Methods and Measures*, *12*(2-3), 93–118. https://doi.org/10.1080/19312458.2018.1430754

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press. https://nlp.stanford.edu/fsnlp/

Martin, G. J. & McCrain, J. (2019). Local news and national politics. *American Political Science Review*, *113*(2), 372–384. https://doi.org/10.1017/S0003055418000965

Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É., Seddah, D., & Sagot, B. (2020). CamemBERT: a tasty French language model. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7203–7219). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.645

Masters, D. & Luschi, C. (2018). Revisiting small batch training for deep neural networks. arXiv. https://arxiv.org/abs/1804.07612

McCallum, A. & Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (pp. 188–191). https://aclanthology.org/W03-0430

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2018). Learned in translation: Contextualized word vectors. arXiv. https://arxiv.org/abs/1708.00107

McCormick, C. & Ryan, N. (2019). BERT fine-tuning tutorial with PyTorch. *Chris McCormick*. https://mccormickml.com/2019/07/22/BERT-fine-tuning/

McKinney, W. (2010). Data structures for statistical computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference (SciPy 2010)* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Meidinger, M. & Aßenmacher, M. (2021). A new benchmark for NLP in social sciences: Evaluating the usefulness of pre-trained language models for classifying open-ended survey responses. In A. P. Rocha, L. Steels, & H. J. van den Herik (Eds.), *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021* (pp. 866–873). SciTePress. https://doi.org/10.5220/0010255108660873

Miao, Y., Yu, L., & Blunsom, P. (2016). Neural variational inference for text processing. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on International Conference on Machine Learning* (pp. 1727–1736). Journal of Machine Learning Research. https://dl.acm.org/doi/10.5555/3045390.3045573

Michael Waskom and Team (2020). *Seaborn*. [Python package]. Zenodo. https://zenodo.org/record/4379347

Mikhaylov, S., Laver, M., & Benoit, K. R. (2012). Coder reliability and misclassification in the human coding of party manifestos. *Political Analysis*, *20*(1), 78–91. https://doi.org/10.1093/pan/mpr047

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv. https://arxiv.org/abs/1301.3781

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In T. Kobayashi, K. Hirose, & S. Nakamura (Eds.), *11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (pp. 1045–1048). International Speech Communication Association. https://doi.org/10.21437/Interspeech.2010-343

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3111–3119). Curran Associates Inc. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Mikolov, T., Yih, W.-t., & Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In L. Vanderwende, H. Daumé III, & K. Kirchhoff (Eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 746–751). Association for Computational Linguistics. https://www.aclweb.org/anthology/N13-1090

Miller, B., Linder, F., & Mebane, W. R. (2020). Active learning approaches for labeling text: Review and assessment of the performance of active learning approaches. *Political Analysis*, *28*(4), 532–551. https://doi.org/10.1017/pan.2020.4

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39–41. https://doi.org/10.1145/219717.219748

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 1–38. https://doi.org/10.1016/j.artint.2018.07.007

Mitchell, M., Aguilar, J., Wilson, T., & Van Durme, B. (2013). Open domain targeted sentiment. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1643–1654). Association for Computational Linguistics. https://aclanthology.org/D13-1171

Mitts, T. (2019). From isolation to radicalization: Anti-Muslim hostility and support for ISIS in the West. *American Political Science Review*, *113*(1), 173–194. https://doi.org/10.1017/S0003055418000618

Moens, M.-F., Boiy, E., Palau, R. M., & Reed, C. (2007). Automatic detection of arguments in legal texts. In A. Gardner & R. Winkels (Eds.), *Proceedings of the 11th International Conference on Artificial Intelligence and Law* (pp. 225–230). Association for Computing Machinery. https://doi.org/10.1145/1276318.1276362

Mohammad, S. M., Bravo-Marquez, F., Salameh, M., & Kiritchenko, S. (2018). SemEval-2018 task 1: Affect in tweets. In M. Apidianaki, S. M. Mohammad, J. May, E. Shutova, S. Bethard, & M. Carpuat (Eds.), *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*. Association for Computational Linguistics. https://doi.org/10.18653/v1/S18-1001

Mohammad, S. M., Sobhani, P., & Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Transactions on Internet Technology*, *17*(3), 1–22. https://doi.org/10.1145/3003433

Moilanen, K. & Pulman, S. (2007). Sentiment composition. In G. Angelova & R. Mitkov (Eds.), *Proceedings of the Recent Advances in Natural Language Processing International Conference (RANLP-2007)* (pp. 378–382).

Molnar, C. (2014). *Statistical modeling: The two cultures* [Presentation]. Ludwig-Maximilians-Universität München. https://github.com/christophM/presentation-two-cultures/blob/master/presentation.pdf

Molnar, C. (2022). *Interpretable machine learning*. https://christophm.github.io/interpretable-ml-book/

Montgomery, D. C. (2012). *Design and analysis of experiments (8th ed.)*. Wiley.

Moore, W. H. & Siegel, D. A. (2013). *A mathematics course for political and social research*. Princeton University Press.

Moosbrugger, H. (2012). Item-Response-Theorie (IRT). In H. Moosbrugger & A. Kelava (Eds.), *Testtheorie und Fragebogenkonstruktion (2nd ed.)*. Springer.

Mosbach, M., Andriushchenko, M., & Klakow, D. (2021). On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In S. Mo-

hamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*. https://openreview.net/forum?id=nzpLWnVAyah

Mozer, R., Miratrix, L., Kaufman, A. R., & Anastasopoulos, J. L. (2020). Matching with text data: An experimental evaluation of methods for matching documents and of measuring match quality. *Political Analysis*, *28*(4), 445–468. https://doi.org/10.1017/pan.2020.1

Muchlinski, D., Yang, X., Birch, S., Macdonald, C., & Ounis, I. (2021). We need to go deeper: Measuring electoral violence using convolutional neural networks and social media. *Political Science Research and Methods*, *9*(1), 122–139. https://doi.org/10.1017/psrm.2020.32

Münchener Digitalisierungszentrum der Bayerischen Staatsbibliothek (dbmdz) (2021). Model card for bert-base-german-uncased from dbmdz. *huggingface.co*. https://huggingface.co/dbmdz/bert-base-german-uncased

Munzert, S., Rubba, C., Meißner, P., & Nyhuis, D. (2015). *Automated data collection with R*. John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118834732

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve Restricted Boltzmann machines. In J. Fürnkranz & T. Joachims (Eds.), *Proceedings of the 27th International Conference on International Conference on Machine Learning* (pp. 807–814). Omnipress. https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

Nakagawa, T., Inui, K., & Kurohashi, S. (2010). Dependency tree-based sentiment classification using CRFs with hidden variables. In R. Kaplan, J. Burstein, M. Harper, & G. Penn (Eds.), *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 786–794). Association for Computational Linguistics. https://aclanthology.org/N10-1120

Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., & Stoyanov, V. (2016). SemEval-2016 task 4: Sentiment analysis in Twitter. In S. Manandhar & D. Yuret (Eds.), *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 1–18). Association for Computational Linguistics. https://doi.org/10.18653/v1/S16-1001

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., & Wilson, T. (2013). SemEval-2013 task 2: Sentiment analysis in Twitter. In S. Manandhar & D. Yuret (Eds.), *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (pp. 312–320). Association for Computational Linguistics. https://aclanthology.org/S13-2052

Nallapati, R., Zhou, B., dos Santos, C., Gülçehre, Ç., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In S. Riezler & Y. Goldberg (Eds.), *Proceedings of The 20th SIGNLL Conference on Computational*

Natural Language Learning (pp. 280–290). Association for Computational Linguistics. https://aclanthology.org/K16-1028

Nanni, F., Zirn, C., Glavaš, G., Eichorst, J., & Ponzetto, S. P. (2016). TopFish: Topic-based analysis of political position in US electoral campaigns. In D. Širinić, J. Šnajder, Z. Fazekas, & S. Bevan (Eds.), *PolText 2016: The International Conference on the Advances in Computational Analysis of Political Text* (pp. 61–67). University of Zagreb. https://madoc.bib.uni-mannheim.de/41550/

National Institute on Deathness and Other Communication Disorders (NIDCD) (2021). *American Sign Language*. NIDCD. https://www.nidcd.nih.gov/health/american-sign-language

Neelakantan, A., Shankar, J., Passos, A., & McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1059–1069). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1113

Nelson, L. K. (2020). Computational grounded theory: A methodological framework. *Sociological Methods & Research*, *49*(1), 3–42. https://doi.org/10.1177/0049124117729703

Nelson, L. K., Burk, D., Knudsen, M., & McCall, L. (2021). The future of coding: A comparison of hand-coding and three types of computer-assisted text analysis methods. *Sociological Methods & Research*, *50*(1), 202–237. https://doi.org/10.1177/0049124118769114

Nguyen, D. Q. & Tuan Nguyen, A. (2020). PhoBERT: Pre-trained language models for Vietnamese. In T. Cohn, Y. He, & Y. Liu (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 1037–1042). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.92

Nozza, D., Bianchi, F., & Hovy, D. (2020). What the [MASK]? Making sense of language-specific BERT models. arXiv. https://arxiv.org/abs/2003.02912

Oliphant, T. E. (2006). *A Guide to NumPy*. Trelgol Publishing.

Oller Moreno, S. (2021). *facetscales: facet grid with different scales per facet.* (Version 0.1.0.9000) [R Package]. github.com/zeehio. https://github.com/zeehio/facetscales

Olsson, F., Sahlgren, M., Ben Abdesslem, F., Ekgren, A., & Eck, K. (2020). Text categorization for conflict event annotation. In A. Hürriyetoğlu, E. Yörük, V. Zavarella, & H. Tanev (Eds.), *Proceedings of the Workshop on Automated Extraction of Socio-Political Events from News 2020* (pp. 19–25). European Language Resources Association (ELRA). https://aclanthology.org/2020.aespen-1.5

Osnabrügge, M., Ash, E., & Morelli, M. (2021). Cross-domain topic classification for political texts. *Political Analysis*, (pp. 1–22). https://doi.org/10.1017/pan.2021.37

Paltoglou, G., Theunis, M., Kappas, A., & Thelwall, M. (2013). Predicting emotional responses to long informal text. *IEEE Transactions on Affective Computing*, *4*(1), 106–115. https://doi.org/10.1109/T-AFFC.2012.26

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pang, B. & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In K. Knight (Ed.), *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics* (pp. 115–124). Association for Computational Linguistics. https://doi.org/10.3115/1219840.1219855

Pang, B. & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2), 1–135. https://doi.org/10.1561/1500000011

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 79–86). Association for Computational Linguistics. https://doi.org/10.3115/1118693.1118704

Park, B., Greene, K., & Colaresi, M. (2020). Human rights are (increasingly) plural: Learning the changing taxonomy of human rights from large-scale text reveals information effects. *American Political Science Review*, *114*(3), 888–910. https://doi.org/10.1017/S0003055420000258

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). Towards the unification and robustness of perturbation and gradient based explanations. *Proceedings of Machine Learning Research*, *28*(3), 1310–1318. https://proceedings.mlr.press/v28/pascanu13.pdf

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Patwa, P., Aguilar, G., Kar, S., Pandey, S., PYKL, S., Gambäck, B., Chakraborty, T., Solorio, T., & Das, A. (2020). SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, & E. Shutova (Eds.), *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 774–790). International Committee for Computational Linguistics. https://doi.org/10.18653/v1/2020.semeval-1.100

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Perry, P. O. & Benoit, K. (2017). Scaling text with the Class Affinity Model. arXiv. https://arxiv.org/abs/1710.08963

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018a). Deep contextualized word representations. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 2227–2237). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1202

Peters, M., Neumann, M., Zettlemoyer, L., & Yih, W.-t. (2018b). Dissecting contextual word embeddings: Architecture and representation. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1499–1509). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1179

Peters, M. E., Ruder, S., & Smith, N. A. (2019). To tune or not to tune? Adapting pretrained representations to diverse tasks. In I. Augenstein, S. Gella, S. Ruder, K. Kann, B. Can, J. Welbl, A. Conneau, X. Ren, & M. Rei (Eds.), *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)* (pp. 7–14). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4302

Phang, J., Févry, T., & Bowman, S. R. (2019). Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. arXiv. https://arxiv.org/abs/1811.01088v2

Pilehvar, M. T. & Camacho-Collados, J. (2020). *Embeddings in natural language processing: Theory and advances in vector representations of meaning.* Morgan & Claypool Publishers. https://doi.org/10.2200/S01057ED1V01Y202009HLT047

Pilny, A., McAninch, K., Slone, A., & Moore, K. (2019). Using supervised machine learning in automated content analysis: An example using relational uncertainty. *Communication Methods and Measures*, *13*(4), 287–304. https://doi.org/10.1080/19312458.2019.1650166

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M.,

Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., & Eryiğit, G. (2016). SemEval-2016 task 5: Aspect based sentiment analysis. In S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, & T. Zesch (Eds.), *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)* (pp. 19–30). Association for Computational Linguistics. https://doi.org/10.18653/v1/S16-1002

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015). SemEval-2015 task 12: Aspect based sentiment analysis. In P. Nakov, T. Zesch, D. Cer, & D. Jurgens (Eds.), *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 486–495). Association for Computational Linguistics. https://doi.org/10.18653/v1/S15-2082

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 task 4: Aspect based sentiment analysis. In P. Nakov & T. Zesch (Eds.), *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (pp. 27–35). Association for Computational Linguistics. https://doi.org/10.3115/v1/S14-2004

Poole, K. T. & Rosenthal, H. (1985). A spatial model for legislative roll call analysis. *American Journal of Political Science*, *29*(2), 357–384. https://doi.org/10.2307/2111172

Poole, K. T. & Rosenthal, H. (1991). Patterns of congressional voting. *American Journal of Political Science*, *35*(1), 228–278. https://doi.org/10.2307/2111445

Porter, E. & Velez, Y. R. (2021). Placebo selection in survey experiments: An agnostic approach. *Political Analysis*, (pp. 1–14). https://doi.org/10.1017/pan.2021.16

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., & Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In S. Pradhan, A. Moschitti, & N. Xue (Eds.), *Joint Conference on EMNLP and CoNLL - Shared Task* (pp. 1–40). Association for Computational Linguistics. https://aclanthology.org/W12-4501

Princeton University (2010). *About WordNet*. https://wordnet.princeton.edu/

Proksch, S.-O. & Slapin, J. B. (2009). How to avoid pitfalls in statistical analysis of political texts: The case of Germany. *German Politics*, *18*(3), 323–344. https://doi.org/10.1080/09644000903055799

Proksch, S.-O. & Slapin, J. B. (2010). Position taking in European Parliament speeches. *British Journal of Political Science*, *40*(3), 587–611. https://doi.org/10.1017/S0007123409990299

Proksch, S.-O. & Slapin, J. B. (2012). Institutional foundations of legislative speech. *American Journal of Political Science*, *56*(3), 520–537. https://doi.org/10.1111/j.1540-5907.2011.00565.x

Puglisi, R. & Snyder, J. M. (2011). Newspaper coverage of political scandals. *The Journal of Politics*, *73*(3), 931–950. https://doi.org/10.1017/s0022381611000569

Quinn, K. M., Monroe, B. L., Colaresi, M., Crespin, M. H., & Radev, D. R. (2010). How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science*, *54*(1), 209–228. https://doi.org/10.1111/j.1540-5907.2009.00427.x

R Core Team (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 8748–8763). PMLR. https://proceedings.mlr.press/v139/radford21a.html

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. Manuscript. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. Manuscript. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. http://jmlr.org/papers/v21/20-074.html

Raganato, A., Camacho-Collados, J., & Navigli, R. (2017). Word sense disambiguation: A unified evaluation framework and empirical comparison. In M. Lapata, P. Blunsom, & A. Koller (Eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers* (pp. 99–110). Association for Computational Linguistics. https://aclanthology.org/E17-1010

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. arXiv. https://arxiv.org/abs/1806.03822

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. arXiv. https://arxiv.org/abs/1606.05250

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-shot text-to-image generation. *Proceedings of Machine Learning Research*, 139, 8821–8831. https://proceedings.mlr.press/v139/ramesh21a.html

Ramey, A. J., Klingler, J. D., & Hollibaugh, G. E. (2019). Measuring elite per-

sonality using speech. *Political Science Research and Methods*, *7*(1), 163–184. https://doi.org/10.1017/psrm.2016.12

Raschka, S. (2020). *watermark*. [Python package]. github.com/rasbt. https://github.com/rasbt/watermark

Rauh, C., Bes, B. J., & Schoonvelde, M. (2020). Undermining, defusing or defending European integration? Assessing public communication of European executives in times of EU politicisation. *European Journal of Political Research*, 59, 397–423. https://doi.org/10.1111/1475-6765.12350

Reddy, S., Chen, D., & Manning, C. D. (2019). CoQA: A conversational question answering challenge. arXiv. https://arxiv.org/abs/1808.07042

Rehbein, I., Lapesa, G., Glavaš, G., & Ponzetto, S. (Eds.) (2021a). *Proceedings of the 1st Workshop on Computational Linguistics for Political Text Analysis (CPSS-2021)*. German Society for Computational Linguistics & Language Technology. https://gscl.org/media/pages/arbeitskreise/cpss/cpss-2021/workshop-proceedings/352683648-1631172151/cpss2021-proceedings.pdf

Rehbein, I., Ponzetto, S. P., Adendorf, A., Bahnsen, O., Stoetzer, L., & Stuckenschmidt, H. (2021b). Come hither or go away? Recognising pre-electoral coalition signals in the news. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7798–7810). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.615

Rehbein, I., Ruppenhofer, J., & Bernauer, J. (2021c). Who is we? Disambiguating the referents of first person plural pronouns in parliamentary debates. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 147–158). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.13

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., & Kim, B. (2019). Visualizing and measuring the geometry of BERT. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf

Reimers, N. & Gurevych, I. (2018). Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. arXiv. http://arxiv.org/abs/1803.09578

Reimers, N. & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp.

3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Rheault, L., Beelen, K., Cochrane, C., & Hirst, G. (2016). Measuring emotion in parliamentary debates with automated textual analysis. *PLoS ONE*, *11*(12), e0168843. https://doi.org/10.1371/journal.pone.0168843

Rheault, L. & Cochrane, C. (2020). Word embeddings for the analysis of ideological placement in parliamentary corpora. *Political Analysis*, *28*(1), 112–133. https://doi.org/10.1017/pan.2019.26

Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4902–4912). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.442

Richardson, L. (2020). *Beautiful Soup 4*. [Python library]. Crummy. https://www.crummy.com/software/BeautifulSoup/

Riedl, M. (2020). AI democratization in the era of GPT-3. *The Gradient*. https://thegradient.pub/ai-democratization-in-the-era-of-gpt-3/

Riezler, S. & Maxwell, J. T. (2005). On some pitfalls in automatic evaluation and significance testing for MT. In J. Goldstein, A. Lavie, C.-Y. Lin, & C. Voss (Eds.), *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (pp. 57–64). Association for Computational Linguistics. https://aclanthology.org/W05-0908

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., & Fergus, R. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, *118*(15), e2016239118. https://doi.org/10.1073/pnas.2016239118

Roberts, M. E., Stewart, B. M., & Airoldi, E. M. (2016a). A model of text for experimentation in the social sciences. *Journal of the American Statistical Association*, *111*(515), 988–1003. https://doi.org/10.1080/01621459.2016.1141684

Roberts, M. E., Stewart, B. M., & Tingley, D. (2016b). Navigating the local modes of big data: The case of topic models. In R. M. Alvarez (Ed.), *Computational Social Science: Discovery and Prediction*, (pp. 51–97). Cambridge University Press, New York.

Roberts, M. E., Stewart, B. M., & Tingley, D. (2019). stm: An R package for Structural Topic Models. *Journal of Statistical Software*, *91*(2), 1–40. https://doi.org/10.18637/jss.v091.i02

Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Luis, J. L., Gadarian, S. K., Albertson, B., & Rand, D. G. (2014). Structural Topic Models for open-

ended survey responses. *American Journal of Political Science*, *58*(4), 1064–1082. https://doi.org/10.1111/ajps.12103

Rodman, E. (2020). A timely intervention: Tracking the changing meanings of political concepts with word vectors. *Political Analysis*, *28*(1), 87–111. https://doi.org/10.1017/pan.2019.23

Rodriguez, P. L. & Spirling, A. (2022). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. *The Journal of Politics*, *84*(1), 101–115. https://doi.org/10.1086/715162

Rona-Tas, A., Cornuéjols, A., Blanchemanche, S., Duroy, A., & Martin, C. (2019). Enlisting supervised machine learning in mapping scientific uncertainty expressed in food risk analysis. *Sociological Methods & Research*, *48*(3), 608–641. https://doi.org/10.1177/0049124117729701

Rothe, S. & Schütze, H. (2015). AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1793–1803). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-1173

Ruder, S. (2018). NLP's ImageNet moment has arrived. *Sebastian Ruder*. https://ruder.io/nlp-imagenet/

Ruder, S. (2019a). *Neural transfer learning for natural language processing*. PhD thesis, National University of Ireland, Galway. https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf

Ruder, S. (2019b). Unsupervised cross-lingual representation learning. *Sebastian Ruder*. https://ruder.io/unsupervised-cross-lingual-learning/index.html

Ruder, S. (2020a). *NLP-Progress*. https://nlpprogress.com/ (accessed August 04, 2020)

Ruder, S. (2020b). NLP-Progress. *nlpprogress.com*. https://nlpprogress.com/english/text_classification.html

Ruder, S. (2020c). Why you should do NLP beyond English. *Sebastian Ruder*. https://ruder.io/nlp-beyond-english/

Ruder, S. (2021). Recent advances in language model fine-tuning. *Sebastian Ruder*. https://ruder.io/recent-advances-lm-fine-tuning/

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, S., & Sedlmair, M. (2018). More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures*, *12*(2–3), 140–157. https://doi.org/10.1080/19312458.2018.1455817

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 379–389). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1044

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., & Gurevych, I. (2021). How good is your tokenizer? on the monolingual performance of multilingual language models. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 3118–3135). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.243

Saeidi, M., Bartolo, M., Lewis, P., Singh, S., Rocktäschel, T., Sheldon, M., Bouchard, G., & Riedel, S. (2018). Interpretation of natural language rules in conversational machine reading. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2087–2097). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1233

Salton, G. (1971). *The SMART retrieval system — Experiments in automatic document processing*. Prentice-Hall, Inc.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Sanders, J., Lisi, G., & Schonhardt-Bailey, C. (2018). Themes and topics in parliamentary oversight hearings: A new direction in textual data analysis. *Statistics, Politics and Policy*, *8*(2), 153–194. https://doi.org/10.1515/spp-2017-0012

Sap, M., Gabriel, S., Qin, L., Jurafsky, D., Smith, N. A., & Choi, Y. (2020). Social bias frames: Reasoning about social and power implications of language. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 5477–5490). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.486

Sarawagi, S. (2007). Information extraction. *Foundations and Trends in Databases*, *1*(3), 261–377.

Sarkar, D. (2016). *Text analytics with Python*. Apress. https://doi.org/10.1007/978-1-4842-2388-8

Scheipl, F., Greven, S., & Kuechenhoff, H. (2008). Size and power of tests for a zero random effect variance or polynomial regression in additive and linear mixed models.

Computational Statistics & Data Analysis, *52*(7), 3283–3299. https://doi.org/10.1016/j.csda.2007.10.022

Schick, T. & Schütze, H. (2021). Exploiting cloze-questions for few-shot text classification and natural language inference. In P. Merlo, J. Tiedemann, & R. Tsarfaty (Eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (pp. 255–269). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.20

Schnell, R., Hill, P. B., & Esser, E. (2018). *Methoden der empirischen Sozialforschung (11th ed.)*. De Gruyter Oldenbourg.

Schulze, P. & Wiegrebe, S. (2020). Twitter in the Parliament — A text-based analysis of German political entities. Manuscript.

Schulze, P., Wiegrebe, S., Thurner, P. W., Heumann, C., Aßenmacher, M., & Wankmüller, S. (2021). Exploring topic-metadata relationships with the STM: A Bayesian approach. arXiv. https://arxiv.org/abs/2104.02496

Schuster, M. & Nakajima, K. (2012). Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5149–5152). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ICASSP.2012.6289079

Schuster, M. & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123. https://aclanthology.org/J98-1004

Schwarz, D., Traber, D., & Benoit, K. (2017). Estimating intra-party preferences: Comparing speeches to votes. *Political Science Research and Methods*, *5*(2), 379–396. https://doi.org/10.1017/psrm.2015.77

scikit-learn Developers (2020a). 1.4. Support Vector Machines. *scikit-learn*. https://scikit-learn.org/stable/modules/svm.html

scikit-learn Developers (2020b). Metrics and scoring: quantifying the quality of predictions. *scikit-learn*. https://scikit-learn.org/stable/modules/model_evaluation.html

scikit-learn Developers (2020c). RBF SVM Parameters. *scikit-learn*. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

scikit-learn Developers (2021). Metrics and scoring: quantifying the quality of predictions. *scikit-learn*. https://scikit-learn.org/stable/modules/model_evaluation.html

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Sebők, M. & Kacsuk, Z. (2021). The multiclass classification of newspaper articles with machine learning: The hybrid binary snowball approach. *Political Analysis*, *29*(2), 236–249. https://doi.org/10.1017/pan.2020.27

Selivanov, D., Bickel, M., & Wang, Q. (2020). *text2vec: Modern text mining framework for R.* [R package]. CRAN. https://CRAN.R-project.org/package=text2vec

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In K. Erk & N. A. Smith (Eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1715–1725). Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1162

Settles, B. (2010). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. http://burrsettles.com/pub/settles.activelearning.pdf

Shepsle, K. A. & Bonchek, M. S. (2010). *Analyzing politics.* W. W. Norton.

Shmueli, B. (2019). Multi-class metrics made simple, part ii: The F1-score. *Towards Data Science.* https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

Silva, A. & Mendoza, M. (2020). Improving query expansion strategies with word embeddings. In *Proceedings of the ACM Symposium on Document Engineering 2020 (DocEng '20)* (pp. 1–4). Association for Computing Machinery. https://doi.org/10.1145/3395027.3419601

Slapin, J. B. & Kirkland, J. H. (2020). The sound of rebellion: Voting dissent and legislative speech in the UK House of Commons. *Legislative Studies Quarterly*, *45*(2), 153–176. https://doi.org/10.1111/lsq.12251

Slapin, J. B. & Proksch, S.-O. (2008). A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, *52*(3), 705–722. https://doi.org/10.1111/j.1540-5907.2008.00338.x

Smith, N. A. (2011). *Linguistic structure prediction.* Morgan & Claypool Publishers.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In R. Barzilay & M. Johnson (Eds.), *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 151–161). Association for Computational Linguistics. https://aclanthology.org/D11-1014

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*

(pp. 1631–1642). Association for Computational Linguistics. https://aclanthology.org/D13-1170

Soetaert, K. (2019). *plot3D: Plotting multi-dimensional data*. (Version 1.3) [R package]. CRAN. https://CRAN.R-project.org/package=plot3D

Song, H., Tolochko, P., Eberl, J.-M., Eisele, O., Greussing, E., Heidenreich, T., Lind, F., Galyga, S., & Boomgaarden, H. G. (2020). In validations we trust? The impact of imperfect human annotations as a gold standard on the quality of validation of automated content analysis. *Political Communication*, *37*(4), 550–572. https://doi.org/10.1080/10584609.2020.1723752

Spirling, A. & Rodriguez, P. L. (2020). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. Manuscript. http://arthurspirling.org/documents/embed.pdf

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Stein, B. & Wachsmuth, H. (Eds.) (2019). *Proceedings of the 6th Workshop on Argument Mining*. Association for Computational Linguistics. https://aclanthology.org/W19-4500

Stier, S., Bleier, A., Bonart, M., Mörsheim, F., Bohlouli, M., Nizhegorodov, M., Posch, L., Maier, J., Rothmund, T., & Staab, S. (2018). *Systematically monitoring social media: the case of the German federal election 2017*. GESIS - Leibniz-Institut für Sozialwissenschaften, Köln. https://doi.org/10.21241/ssoar.56149

Stimson, J. A., Mackuen, M. B., & Erikson, R. S. (1995). Dynamic representation. *American Political Science Review*, *89*(03), 543–565. https://doi.org/10.2307/2082973

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019a). VideoBERT: A joint model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/ICCV.2019.00756

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019b). How to fine-tune BERT for text classification? arXiv. https://arxiv.org/abs/1905.05583v3

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (pp. 3104–3112). MIT Press. https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2021). Long Range Arena: A benchmark for efficient Transformers. In S. Mohamed (Ed.), *9th International Conference on Learning Representations (ICLR 2021)*. https://openreview.net/forum?id=qVyeW-grC2k

Tay, Y., Tran, V. Q., Ruder, S., Gupta, J., Chung, H. W., Bahri, D., Qin, Z., Baumgartner, S., Yu, C., & Metzler, D. (2022). Charformer: Fast character Transformers via gradient-based subword tokenization. In K. Hofmann & A. Rush (Eds.), *10th International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=JtBRnrlOEFN

Tenney, I., Das, D., & Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4593–4601). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1452

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., & Pavlick, E. (2019b). What do you learn from context? Probing for sentence structure in contextualized word representations. In T. Sainath (Ed.), *7th International Conference on Learning Representations, ICLR 2019*. https://openreview.net/pdf?id=SJzSgnRcKX

The American National Election Studies (2015). *ANES 2008 Time Series Study*. [Data set]. The American National Election Studies. https://electionstudies.org/data-center/2008-time-series-study/

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, *61*(12), 2544–2558. https://doi.org/10.1002/asi.21416

Theocharis, Y., Barberá, P., Fazekas, Z., Popa, S. A., & Parnet, O. (2016). A bad workman blames his tweets: The consequences of citizens' uncivil Twitter use when interacting with party candidates. *Journal of Communication*, *66*(6), 1007–1031. https://doi.org/10.1111/jcom.12259

Thet, T. T., Na, J.-C., & Khoo, C. S. (2010). Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, *36*(6), 823–848. https://doi.org/10.1177/0165551510388123

Thomas, G. (2018). *Mathematics for machine learning* [Notes]. University of California, Berkeley. https://gwthomas.github.io/docs/math4ml.pdf

Tjong Kim Sang, E. F. & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (pp. 142–147). https://aclanthology.org/W03-0419

Tong, S. & Koller, D. (2001). Support Vector Machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2, 45–66. https://www.jmlr.org/papers/v2/tong01a.html

Torch Contributors (2021). Reproducibility. *pytorch.org*. https://pytorch.org/docs/stable/notes/randomness.html

Torres, M. & Cantu, F. (2022). Learning to see: Convolutional neural networks for the analysis of social science data. *Political Analysis*, *30*(1), 113–131. https://doi.org/10.1017/pan.2021.9

Toutenburg, H., Heumann, C., & Nittner, T. (2004). Statistische Methoden bei unvollständigen Daten. Open Access LMU. https://doi.org/10.5282/ubm/epub.1750

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In P. Isabelle (Ed.), *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417–424). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073153

Turney, P. D. & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37, 141–188. https://doi.org/10.1613/jair.2934

Ulmer, D., Bassignana, E., Müller-Eberstein, M., Varab, D., Zhang, M., Hardmeier, C., & Plank, B. (2022). Experimental standards for deep learning research: A natural language processing perspective. In S. Chan, R. Agarwal, X. Bouthillier, C. Gulcehre, & J. Dodge (Eds.), *ICLR Workshop on ML Evaluation Standards*. https://arxiv.org/abs/2204.06251

Ushey, K., Allaire, J. J., Wickham, H., & Ritchie, G. (2020). *rstudioapi: Safely access the RStudio API*. (Version 0.11) [R package]. CRAN. https://CRAN.R-project.org/package=rstudioapi

Uyheng, J. & Carley, K. M. (2020). Bots and online hate during the COVID-19 pandemic: Case studies in the United States and the Philippines. *Journal of Computational Social Science*, 3, 445–468. https://doi.org/10.1007/s42001-020-00087-4

van Atteveldt, W., Sheafer, T., Shenhav, S. R., & Fogel-Dror, Y. (2017). Clause analysis: Using syntactic information to automatically extract source, subject, and predicate from texts with an application to the 2008–2009 Gaza War. *Political Analysis*, *25*(2), 207–222. https://doi.org/10.1017/pan.2016.12

van Rijsbergen, C. J. (2000). *Information retrieval. Session 1: Introduction to information retrieval* [Lecture Notes]. Universität Duisburg Essen. https://www.is.inf.uni-due.de/courses/dortmund/lectures/ir_ws00-01/folien/keith_intro.ps

van Rossum, G. & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Vapnik, V. (1991). Principles of risk minimization for learning theory. In J. Moody, S. Hanson, & R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems 4* (pp. 831–838). Morgan-Kaufmann. https://proceedings.neurips.cc/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances*

in Neural Information Processing Systems 30, (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of Support Vector Machines. In T. Dean (Ed.), *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '99)* (pp. 55–60).

Volkens, A., Burst, T., Krause, W., Lehmann, P., Matthieß, T., Regel, S., Weßels, B., & Zehnter, L. (2021a). *The Manifesto Data Collection. Manifesto Project (MRG / CMP / MARPOR). Version 2021a.* [Data set]. Wissenschaftszentrum Berlin für Sozialforschung (WZB). https://doi.org/10.25522/manifesto.mpds.2021a

Volkens, A., Burst, T., Krause, W., Lehmann, P., Matthieß, T., Regel, S., Weßels, B., & Zehnter, L. (2021b). *The Manifesto Project Dataset - Codebook. Manifesto Project (MRG / CMP / MARPOR). Version 2021a.* Wissenschaftszentrum Berlin für Sozialforschung (WZB). https://manifesto-project.wzb.eu/down/data/2021a/codebooks/codebook_MPDataset_MPDS2021a.pdf

Walker, C., Strassel, S., Medero, J., & Maeda, K. (2006). *ACE 2005 Multilingual training corpus.* [Data set]. Linguistic Data Consortium. https://doi.org/10.35111/mwxc-vh88

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 3266–3280). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf

Wang, H. (2020). Logistic regression for massive data with rare events. *Proceedings of Machine Learning Research*, 119, 9829–9836. http://proceedings.mlr.press/v119/wang20a.html

Wang, J., Peng, B., & Zhang, X. (2018). Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*, 322, 93–101.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-attention with linear complexity. arXiv. https://arxiv.org/abs/2006.04768

Wang, X., Liu, Y., Sun, C., Wang, B., & Wang, X. (2015). Predicting polarities of tweets by composing word embeddings with long short-term memory. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1343–1353). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-1130

Wankmüller, S. (2021). Neural transfer learning with Transformers for social science text analysis. arXiv. https://arxiv.org/abs/2102.02111

Wankmüller, S. & Heumann, C. (2021). How to estimate continuous sentiments from texts using binary training data. In K. Evang, L. Kallmeyer, R. Osswald, J. Waszczuk, & T. Zesch (Eds.), *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)* (pp. 182–192). KONVENS 2021 Organizers. https://aclanthology.org/2021.konvens-1.16

Watanabe, K. (2021). Latent semantic scaling: A semisupervised text analysis technique for new domains and languages. *Communication Methods and Measures*, *15*(2), 81–102. https://doi.org/10.1080/19312458.2020.1832976

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2022). Finetuned language models are zero-shot learners. In K. Hofmann & A. Rush (Eds.), *International Conference on Learning Representations (ICLR 2022)*. https://openreview.net/forum?id=gEZrGCozdqR

Welbers, K., van Atteveldt, W., & Benoit, K. (2017). Text analysis in R. *Communication Methods and Measures*, *11*(4), 245–265. https://doi.org/10.1080/19312458.2017.1387238

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer. https://ggplot2.tidyverse.org/

Wickham, H. (2019). *stringr: Simple, consistent wrappers for common string operations*. (Version 1.4.0) [R package]. CRAN. https://CRAN.R-project.org/package=stringr

Wickham, H., François, R., Henry, L., & Müller, K. (2021). *dplyr: A grammar of data manipulation*. (Version 1.0.6) [R package]. CRAN. https://CRAN.R-project.org/package=dplyr

Wild, F. (2020). *lsa: Latent Semantic Analysis*. (Version 0.73.2) [R package]. CRAN. https://CRAN.R-project.org/package=lsa

Williams, A., Nangia, N., & Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1112–1122). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1101

Wiseman, S., Shieber, S., & Rush, A. (2017). Challenges in data-to-document generation. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2253–2263). Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1239

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). Hugging Face's Transformers: State-of-the-art natural language processing. arXiv. https://arxiv.org/abs/1910.03771v5

Wood, W. (1982). Retrieval of attitude-relevant information from memory: Effects on susceptibility to persuasion and on intrinsic motivation. *Journal of Personality and Social Psychology*, *42*(5), 798–810.

Wu, P. Y. & Mebane, W. R. (2021). MARMOT: A deep learning framework for constructing multimodal representations for vision-and-language tasks. arXiv. https://arxiv.org/abs/2109.11526v1

Wu, S. & Dredze, M. (2019). Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 833–844). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1077

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv. https://arxiv.org/abs/1609.08144

xgboost Developers (2020). Python API reference. https://xgboost.readthedocs.io/en/latest/python/python_api.html

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10, 291–306. https://doi.org/10.1162/tacl_a_00461

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, & Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 483–498). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.41

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, (pp. 5753–5763). Curran Associates, Inc. http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2020). XLNet: Generalized autoregressive pretraining for language understanding. arXiv. https://arxiv.org/abs/1906.08237

Yin, P., Neubig, G., Yih, W.-t., & Riedel, S. (2020). TaBERT: Pretraining for joint understanding of textual and tabular data. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8413–8426). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.745

Yin, W., Hay, J., & Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3914–3923). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1404

Ying, L., Montgomery, J. M., & Stewart, B. M. (2021). Topics, concepts, and measurement: A crowdsourced procedure for validating topics as measures. *Political Analysis*, (pp. 1–20). https://doi.org/10.1017/pan.2021.33

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27*, (pp. 3320–3328). Curran Associates, Inc. http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big Bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 17283–17297). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf

Zarrella, G. & Marsh, A. (2016). MITRE at SemEval-2016 task 6: Transfer learning for stance detection. In S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, & T. Zesch (Eds.), *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 458–463). Association for Computational Linguistics. https://doi.org/10.18653/v1/S16-1074

Zellers, R., Bisk, Y., Schwartz, R., & Choi, Y. (2018). SWAG: A large-scale adversarial dataset for grounded commonsense inference. arXiv. https://arxiv.org/abs/1808.05326

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4791–4800). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1472

Zhang, H. & Pan, J. (2019). CASM: A deep-learning approach for identifying collective

action events with text and image data from social media. *Sociological Methodology*, *49*(1), 1–57. https://doi.org/10.1177/0081175019860244

Zhang, M., Zhang, Y., & Vo, D.-T. (2015a). Neural networks for open domain targeted sentiment. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 612–621). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1073

Zhang, M.-L. & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

Zhang, X., Zhao, J., & LeCun, Y. (2015b). Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Proceedings of the 28th International Conference on Neural Information Processing Systems* (pp. 649–657). MIT Press. https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

Zhao, H., Phung, D., Huynh, V., Jin, Y., Du, L., & Buntine, W. (2021). Topic modelling meets deep neural networks: A survey. In Z.-H. Zhou (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (pp. 4713–4720). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2021/638

Zheng, Z., Hui, K., He, B., Han, X., Sun, L., & Yates, A. (2020). BERT-QE: Contextualized query expansion for document re-ranking. In T. Cohn, Y. He, & Y. Liu (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 4718–4728). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.424

Zhou, L., Xu, C., & Corso, J. J. (2017). Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press. https://ojs.aaai.org/index.php/AAAI/article/download/12342/12201

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)* (pp. 19–27). IEEE. https://doi.org/10.1109/ICCV.2015.11