

Matthias Aßenmacher

Comparability, Evaluation and Benchmarking of large pre-trained language models

Dissertation an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

Eingereicht am 16.07.2021

Matthias Aßenmacher

Comparability, Evaluation and Benchmarking of large pre-trained language models

Dissertation an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

Eingereicht am 16.07.2021

Erster Berichterstatter: Prof. Dr. Christian Heumann
Zweiter Berichterstatter: Prof. Dr. Benjamin Roth
Dritter Berichterstatter: Prof. Dr. Hinrich Schütze

Tag der Disputation: 13.10.2021

Acknowledgments

During these almost four and a half years of writing this thesis, I received help and guidance from some incredible people. Without their support and advice, I would have never been able to finish this piece of hard work.

In particular, I would like to express my sincere gratitude to ...

- ... my supervisor Prof. Dr. Christian Heumann for his outstanding support, all the advice and encouragement. Not only have you shaped my way of how to approach research and teaching, but you were also a great teacher regarding leadership and general attitude.*
- ... Prof. Dr. Benjamin Roth and Prof. Dr. Hinrich Schütze for their willingness to act as the second and third reviewer for my Ph.D. thesis.*
- ... PD Dr. Fabian Scheipl and Prof. Dr. Helmut Küchenhoff for their availability to be part of the examination panel at my Ph.D. defense.*
- ... Prof. Dr. Helmut Küchenhoff for awakening my interest in research during my work at the Statistical Consulting Unit, my student consulting project and my master thesis.*
- ... Georg Hansbauer and Markus Steinhauser (Testbirds GmbH) whose collaboration within the SALUTT project initially got me started on NLP back in 2017.*
- ... all my coauthors (in temporal order: Christian, Max, Maike, Vanessa, Elisabeth, Naiwen, Korbinian, Alessandra, Patrick) for the fruitful collaborations and our collective fights against submission deadlines.*
- ... Prof. Dr. Christian Heumann, Prof. Dr. Hinrich Schütze, Daniel Schalk, Nina Poerner and Leonie Weißweiler for our collective effort of organizing great NLP-related lectures and seminars.*
- ... my good colleague Ben Sischka and all my former student assistants (in temporal order: Katrin, Sandra, Sevag, Julia, Alex, Patricia, Christoph, Max, Charlotte, Ann-Kathrin, Federico, Simon, Lisa, Katrin, Lukas, Nora) who did an excellent job in teaching and correcting exams during our collective struggle of convincing the economics students of the importance of statistics.*
- ... Prof. Dr. Hinrich Schütze and Prof. Dr. Alex Fraser for welcoming me to their Ph.D. seminar and the reading group at the Center for Information and Language Processing.*
- ... all current and former office colleagues: Thank you Clara and Christopher for the very nice working atmosphere, I really enjoyed it.*
- ... and last but not least my parents Christine and Lothar, my sisters Katharina and Jule and my significant other, Susanne, whose support and understanding during the time of my Ph.D. studies meant the world to me. Thank you for tolerating all the weekend work and enduring my constant jabbering about academia, my research and NLP related topics.*

Summary

The concept of transfer learning aims at transferring knowledge learned during solving a specific task in a specific domain to other tasks or domains, respectively. While this paradigm was already employed in the field of Computer Vision in the early 2010s, it revolutionized the field of Natural Language Processing about half a decade later. This doctoral thesis deals with three crucial aspects which have to be considered and payed attention to when applying and researching about these kinds of model architectures.

The first part of this work addresses critical aspects when it comes to the definition of fair comparisons for pre-trained language models. Contrary to classical machine learning it is not straightforward to define what a model essentially is, since a model is not just the mere architecture but it also comprises the complete pre-training procedure (pre-training text corpus and amount of computational power). Besides this, also the model size plays a crucial role as it sometimes may be prohibitively large for some practitioners or devices which is why it should also be considered when comparing state-of-the-art (SOTA) models. The first contributing article raises awareness for the above-mentioned issues and proposes potential circumventions for them when performing or evaluating model comparisons.

In the second part, the usefulness of several state-of-the-art architectures on a set of complex tasks is evaluated. As for the second contributing article, the model performance is evaluated on the task of automatically classifying answers to open-ended questions into a predefined set of categories. This showcases an (extreme) multi-label classification task which social scientists are commonly facing. Alongside with this, a fully reproducible preparation of the data from the *American National Election Studies* (ANES 2008) for machine learning purposes is provided. In the third contribution pre-trained models are applied to the task of *Fake News Detection*, with a special focus on the sensitivity towards hyperparameters during model fine-tuning. Experiments and grid search results for different freezing techniques, batch sizes and sequence lengths as well as learning rate schedules are shown. The fourth and fifth contributing articles showcase industrial use cases: The former is about trying to incorporate domain-specific knowledge from an external corpus via *Continual Pre-training* of the language model with the aim of enabling the language model to act as a sort of knowledge base for specific domains. Evaluation at fixed intervals during the training procedure already show partly promising results. The latter project aimed at building a pipeline, heavily relying on pre-trained (German) language models, to measure the concept of *Customer Centricity*. Unstructured customer feedback about car insurances is classified with respect to the addressed aspects and the respective tonality and subsequently (visually) summarized in a radar chart. With the sixth contribution, the attempt is made to contribute to closing a large research gap: Language-specific evaluation of pre-trained models. In this work, currently existing German and multilingual pre-trained architectures are evaluated on the task of (*Aspect-based*) *Sentiment Analysis*, resulting in a substantial increase of the state-of-the-art results.

The third part rounds off the scope of this thesis by showing experimental results from a benchmark study. In the seventh and final contributing article, down-scaled versions of language models were benchmarked on a set of tasks constraining external factors like the budget of computational power and size of the pre-training text corpus.

Zusammenfassung

Das Ziel des Ansatzes des *Transfer Learnings* ist es, das beim Training bezüglich eines spezifischen Tasks erlernte Wissen auf andere (ähnliche) Tasks bzw. Domänen zu übertragen. Während dieses Vorgehen bereits seit Anfang der 2010er Jahre im Bereich der Computer Vision üblich ist, fand es erst etwa ein halbes Jahrzehnt später im Bereich *Natural Language Processing* breite Anwendung. In dieser Dissertation werden drei verschiedene Aspekte beleuchtet, welche bei der Anwendung von und Forschung über diese Art von Modellarchitekturen berücksichtigt werden sollten.

Im ersten thematischen Teilkomplex dieser Arbeit wird der Fokus auf die aktuelle Praxis zum Vergleich von vortrainierten Sprachmodellen gelegt. Hierbei werden insbesondere kritische Aspekte herausgearbeitet, welche sich im Vergleich zum "klassischen" maschinellen Lernen ergeben. Dies rührt daher, dass die Modellarchitektur in diesem Fall nicht einfach nur aus dem Algorithmus selbst, sondern auch dem gesamten Verfahren des Pre-Trainings (Pre-Training-Textkorpus und verwendete Rechenleistung) besteht. Darüber hinaus spielt auch die Zahl der Modellparameter eine nicht zu vernachlässigende Rolle, weshalb auch sie beim Vergleich von state-of-the-art (SOTA) Modellen berücksichtigt werden sollte. Der erste wissenschaftliche Beitrag versucht das Bewusstsein hierfür zu schärfen und gleichzeitig mögliche Ansätze zur Verbesserung vorzuschlagen.

Der zweite Teilkomplex beinhaltet fünf verschiedene Anwendungsfälle anhand derer die Performance mehrerer state-of-the-art Architekturen verglichen wird. Der zweite Beitrag beschäftigt sich mit der Anwendung der Modelle zum Zwecke der automatischen Klassifizierung von Antworten auf offene Fragen in vordefinierte Kategorien. Hierbei handelt es sich um ein multi-label Klassifikationsproblem, ein Task mit hohem Komplexitätsgrad, mit dem Sozialwissenschaftler oft konfrontiert sind. Im Rahmen der Analyse war eine vollständig reproduzierbare Aufbereitung der Daten der *American National Election Studies* (ANES 2008) notwendig, welche ebenfalls einen wichtigen Teil dieser Publikation darstellt. Im dritten Beitrag wird die Empfindlichkeit vortrainierter Modelle gegenüber Hyperparametern beim Fine-Tuning am Anwendungsfall der *Fake News Detection* untersucht. Es werden Experimente mittels Grid Search hinsichtlich verschiedener Freezing Techniques, Batch Sizes und Sequence Lengths, sowie für verschiedene Learning Rate Schedules durchgeführt. Der vierte und fünfte Beitrag zeigen Anwendungsfälle im industriellen Kontext: Bei ersterem wird versucht domänenspezifisches Wissen aus einem externen Textkorpus durch fortgesetztes Pre-Training in das Sprachmodell einzubringen, mit dem Ziel, es anschließend als *Knowledge base* für die Automobil-Domäne verwenden zu können. Die Evaluierung des Modells in fixen Abständen zeigt hierbei bereits teilweise vielversprechende Ergebnisse. Das zweite der beiden Projekte zielt darauf ab, eine Pipeline zur Messung des Konzepts der *Customer Centricity* aufzubauen. Hierbei wird unstrukturiertes Kundenfeedback zu Kfz-Versicherungen mit Hilfe von (deutschen) vortrainierten Sprachmodellen hinsichtlich angesprochener Aspekte und deren Tonalität klassifiziert und anschließend in einem Radar-Chart visualisiert. Im sechsten Beitrag liegt der Fokus auf der sprachenspezifischen Evaluation von vortrainierten Modellen. Hier werden deutsche und multilinguale vortrainierte Sprachmodelle auf dem Task der (*aspektbasierten*) *Sentimentanalyse* evaluiert, wobei eine Verbesserung der state-of-the-art Ergebnisse erreicht wird.

Der dritte thematische Abschnitt rundet den Anwendungsbereich dieser Arbeit durch experimentelle Ergebnisse einer Benchmark-Studie ab. Im siebten Beitrag wurden hierbei herunterskalierte Versionen populärer Sprachmodelle, unter Einschränkung externer Faktoren wie Rechenleistung und Größe des Pre-Trainings-Textkorpus, hinsichtlich ihrer Performance auf einer Reihe verschiedener Tasks verglichen.

Contents

List of Figures	xi
List of Tables	xiii
Notation	xv
I. Introduction and Background	1
1. Introduction	3
1.1. Outline	3
1.2. Motivation and Scope	3
2. Methodological and General Background	5
2.1. Benchmarks Tasks	5
2.2. Performance Measures	7
2.3. Machine Learning Methods for Text Data	12
2.4. Neural network architectures	13
2.5. Shallow Networks for Natural Language Processing	15
2.5.1. Neural probabilistic language model	15
2.5.2. Representation learning	15
2.6. Deep Learning for Natural Language Processing	19
2.6.1. Recurrent architectures	19
2.6.2. The Transformer	22
3. (Sequential) Transfer Learning	25
3.1. Self-Supervised Pre-training	26
3.2. Feature-based Transfer Learning	28
3.3. Fine-tuning approach	29
II. Comparability	35
4. On the comparability of pre-trained language models	37
III. Task-specific evaluation	49
5. Evaluating pre-trained language models on applications in Social Sciences	51
6. Detecting stances of Fake News using pre-trained language models	61
7. Pre-trained language models as knowledge bases for automated complaint analysis	73
8. Applying pre-trained language models for measuring Customer Centricity	79
9. Re-Evaluating GermEval17 using German pre-trained language models	89

IV. Benchmark Studies	103
10. Benchmarking down-scaled transformer-based architectures	105
V. Conclusion	121
11. Future Directions and Concluding Remarks	123
11.1. Few-Shot Learning	123
11.2. Final Thoughts	124
Contributing Publications	125
Further References	127

List of Figures

2.1. The two word2vec model architectures proposed by Mikolov et al. (2013a)	16
2.2. The two doc2vec model architectures proposed by Le and Mikolov (2014)	18
2.3. The Transformer encoder (Vaswani et al., 2017)	23
2.4. The complete Transformer architecture (Vaswani et al., 2017)	24
3.1. Transfer Learning Timeline Part I	29
3.2. Transfer Learning Timeline Part II	31
3.3. Transfer Learning Timeline Part III	32

List of Tables

2.1. Different Variants of classification tasks	6
2.2. Exemplary confusion matrix \mathbf{C} for a binary classification task.	9
2.3. Exemplary confusion matrix \mathbf{C} for a multi-class classification task.	10
2.4. Resources for pre-trained word embeddings	19

Notation

Throughout this thesis, the following conventions will be used:

\mathcal{C}	A corpus of M documents
\mathbf{C}	The confusion matrix of a classification problem
C	Size of the context window
E	Embedding size // Size of the vector representation
h	Number of Attention heads in a Transformer-based architecture
H	Dimension of the hidden layer of a neural network
i	Running index for observations in a data set, words/tokens in a sequence or documents in a corpus
j	Running index for covariates or levels of a categorical target variable
k	The number of classes of a categorical target variable
\mathcal{L}	A data generating process a.k.a. language
\mathcal{M}	A (language) model
m	The number of words/tokens in a sequence s
M	The number of documents in a corpus \mathcal{C}
n	The number of observations in a data set
p	The number of features/covariates in a tabular data set
s	An ordered sequence of m words
θ	The parameters of a model
$V = \{w_1, \dots, w_N\}$	The vocabulary of a corpus \mathcal{C} or a language \mathcal{L}
$ V $ or N	The size (number of distinct words) of a vocabulary V
w_t	The word/token at the t -th position in a sequence
\mathbf{W}	The weight matrix of a neural network
x	Influential variable(s)
y	Target variable(s)

Scalars are denoted by lower case letters (x), while bold lower case letters (\mathbf{x}) denote (column) vectors and bold upper case letters (\mathbf{X}) matrices. Subscripts are used to refer to (scalar) elements of vectors (x_i) or matrices (X_{ij}), to entire rows/columns of matrices (\mathbf{x}_i) or to denote different elements (\mathbf{X}_1 vs. \mathbf{X}_2).

Part I.

Introduction and Background

1. Introduction

1.1. Outline

The time frame during which this thesis was written most definitely ranks among the most exciting times regarding the developments in the field of Natural Language Processing (NLP). This thesis will first describe the development of the most important methods and model architectures along a time axis starting from the early 2000s. The contributions will then mainly focus on the comparability of these architectures by evaluating them on a diverse set of real-world tasks and performing benchmark experiments focused on rather technical parameters like run time and performance under resource constraints.

Part I will lay the foundation for a thorough understanding of this thesis by introducing general concepts and methods one has to deal with when it comes to language modeling. While Ch. 2 presents a set of (benchmark) tasks (Sec. 2.1), as well as accompanying performance measures (Sec. 2.2) NLP researchers are generally interested in, the remaining part of this chapter (Sec. 2.3 – Sec. 2.6) puts a focus on methodological developments. Starting with rather simple, count-based machine learning algorithms the focus is then shifted towards neural network based algorithms and finally reaches cutting-edge deep learning methods for NLP. Why and how these methods fit in the paradigm of (sequential) transfer learning will subsequently be explained in Ch. 3.

The remainder of this work is divided into three main parts (i.e., Part II – IV), in which all contributions to this thesis are embedded as chapters. Part II deals with the issue of deficient model comparability and thus provides the general scope and motivation for this thesis. The subsequent Parts III and IV focus on comparing different models with respect to different criteria. Alongside with the original publication, full reference as well as copyright information and a description of the author’s contributions are provided for each chapter. Supplementary materials like data sets, accompanying software or other are linked if applicable.

The concluding Part V describes the most recent research insights from the field of NLP and puts the contributions in relation to them. Furthermore, open research questions and possible future directions are emphasized.

1.2. Motivation and Scope

Written sequences of human language, i.e. text data, probably represent the most ubiquitous type of data. While for a long time the focus of statistical modeling techniques and machine learning algorithms has been on tabular data sets, it has somewhat shifted in the past two decades. These tabular data sets (partly) have to be created manually, generating a lot of extra effort and potentially also costs. Text data on the other hand is omnipresent in nearly every situation of everyday life, be it in daily communication via email or text messages, on the internet or even

in good old-fashioned books. The vast amount of data as well as the overwhelming complexity of human language still present a major challenge when it comes to its automated processing by machines. Nevertheless it has become, alongside with visual data (images, videos), the most important resource for algorithms which are nowadays most often referred to as "*Artificial Intelligence*". While this might sound quite impressive at first, it is (at least in my humble opinion) still a little bit of a very far reach since most of the architectures are basically highly specialized algorithms performing extraordinary well on some specific (set of) tasks. So the scope of this thesis is about evaluating and understanding the performance of these recently developed algorithms, more specifically a set of pre-trained language models all centered around one focal point: *BERT* (Devlin et al., 2019).

Several major breakthroughs concerning the methodology of how to process written forms of natural language have shaped the past ten years, with two of them really standing out: Mikolov et al. (2013c) revolutionized the way researchers approach the problem of representing words by introducing a framework which is able to efficiently transform these discrete units (words) into semantically and syntactically meaningful continuous, real-valued representations. Since then more (and even better) algorithms have been developed with an overall goal of *Representation Learning* and it has become the standard procedure to build deep learning architectures on top of such representations. All these efforts to enable models to learn ever better representations ultimately culminated in the development of an algorithm framed "*Bidirectional Encoder Representations from Transformers*", short BERT (Devlin et al., 2019). Since its introduction (10/2018 on ArXiv, 06/2019 at NAACL) it has become the point of focus for the whole field of NLP, triggering an enormous amount of research aiming at (i) improving upon its performance, (ii) explaining its inner workings, (iii) improving its computational efficiency, (iv) adapting it to various tasks and more. This streamline of research is by now often referred to as an own research direction named BERTology (Rogers et al., 2020).

The main motivation of all the research articles that ultimately ended up in constituting this thesis, was to also contribute to this body of research. To enable a better understanding of the architectures and to urge researchers to think a little bit more about somewhat more subtle aspects when it comes to comparability and benchmarking of BERT & Co. have since the beginning been the driving forces of this research.

More recently a new algorithm (GPT-3; Brown et al., 2020) was proposed, which might turn out to be the next major breakthrough since the introduction of BERT. The architecture sets new standards with respect to the amount of computational power as well as to the amount of data that was used for training it. Furthermore, it employs a completely different paradigm compared to BERT & Co., when it comes to adapting to new data and shows impressive results in doing so. Nevertheless, there are also critical remarks about whether the approach of (colloquially spoken) "higher, faster, further" will ultimately "solve"¹ the problem of teaching language to machine learning models or if smaller and smarter alternatives (e.g. Schick and Schütze, 2020a) are the solution. Section 11 provides some more details on recent and potential further developments.

¹Of course the word "solve" is only meant metaphorically in this context. This is, on a side note, a pretty nice example of *word sense disambiguation* that (most probably) presents no problem for you, a human reader, but most definitely would prove to be quite hard to understand for a machine.

2. Methodological and General Background

2.1. Benchmarks Tasks

Natural Language Processing has nowadays become an umbrella term for a variety of different tasks and algorithms. Because of this, it is important to obtain a thorough understanding of the nature of different possible tasks and for how to evaluate performance. In the context of this thesis, the term *task* will be almost exclusively used as a proxy for *Supervised Classification Task* if not stated otherwise. While unsupervised¹ machine learning methods like e.g. LSI (Deerwester et al., 1990), pLSI (Hofmann, 1999), LDA (Blei et al., 2003) or the STM (Roberts et al., 2016) are clearly very relevant topics and are still subject to active research, they are outside the scope of this work. (Neural) Machine Translation and Sequence-to-sequence modeling (Sutskever et al., 2014) in general will also, for the sake of completeness, be touched on very briefly during the course of this chapter.

Supervised classification tasks can, in general, be subdivided into three different variants as depicted in Tab. 2.1. Since language modeling deals with a set of discrete units (e.g. words) it easily fits in the framework of multi-class classification, where a "class" is simply one word from the language's vocabulary. The classic language modeling task would thus be a multi-class classification task with $|V|$ classes, where V is the vocabulary. Besides this, binary classification is probably the most common variant and includes tasks like e.g. Sentiment Analysis (Socher et al., 2013), with the labels *positive* and *negative*, Duplicate/Spam Detection (Shankar et al., 2017) or Recognizing Textual Entailment (Wang et al., 2018). A multi-class classification problem typically occurs (besides for the language modeling task), when more fine-grained labels are present. This pertains e.g. to tasks like Sentiment Analysis, when the label *neutral* is present additionally to *positive* and *negative*, or Stance Detection (cf. Sec. 6). Multi-label classification differs from the other two variants with respect to the number of labels which can be associated with one observation, which is potentially larger than one for this case. Illustrative examples for this task are e.g. (supervised) topic modeling, aspect-based sentiment analysis (Pontiki et al., 2014, 2015, 2016; Wojatzki et al., 2017) or coding of open-ended questions in the social sciences (cf. Sec. 5).

So in order to better understand the issues when it comes to comparing and evaluating (pre-trained) language models, this section will introduce the general concept of benchmarking as well as the most frequently used tasks and practices in NLP. Furthermore, the measures for evaluation will be explained, subdivided in task-specific and task-agnostic measures.

Benchmarking In the context of machine learning, the term *Benchmarking* refers to the comparison of different algorithms on one (or multiple) tasks with respect to selected performance measures. Additionally, to ensure a maximum degree of comparability, identical train and test sets

¹While supervised learning methods require external labels to be present, unsupervised methods can be applied to unannotated data sets.

Task Variant	Output space	Targets
Binary Classification	$L = \{0; 1\}$	$y_i = \{0; 1\}^1$
Multi-Class Classification	$L = \{0; \dots; k\}$	$y_i = \{0; \dots; k\}^1$
Multi-Label Classification	$L = \{0; \dots; k\}$	$y_i = \{0; \dots; k\}^k$

Table 2.1.: The different variants of supervised classification. In binary classification an observation can only belong to one of two possible classes, while in multi-class classification it belongs to one of k classes. Multi-label classification extends the multi-class case in a way, that multiple labels can be associated to one observation.

as well as the same resampling strategy are used. When transferring this definition to the context of deep learning and transfer learning several issues arise, which are addressed in the *Discussion* section of the first contribution (cf. Sec. 4). Transfer learning models are not solely defined by the specification of an algorithm, but also inherit a lot of their power from the pre-training procedure (cf. Sec. 3.1). In the NLP context this specifically means the text corpus, which the model is pre-trained on, as well as its size and the computational resources spent on pre-training. Thus, these parameters should be taken into account when it comes to defining the term *Benchmarking* in the context of transfer learning for NLP, but often they are not. This is oftentimes not the fault of other researchers, but rather a result from constraints concerning time and computational resources.

Benchmark Tasks in NLP Comparisons of different NLP (transfer learning) models are commonly conducted on benchmark data sets (e.g. Rajpurkar et al., 2016, 2018) or collections of multiple data sets at once, with each data set providing a different task referring to (multiple) different facets of e.g. Natural Language Understanding (NLU) (Wang et al., 2018, 2019). The values of selected performance measures are subsequently aggregated and commonly displayed on public leaderboards (cf. <https://gluebenchmark.com/leaderboard>). Of course numerous other benchmark data sets exist, which is why it would be nearly impossible (and not in the scope of this thesis) to provide an exhaustive list. Instead, this paragraph will provide an overview on different possible *types* of tasks used for benchmarking in order to provide the basis for a better understanding of the subsequent sections:

- **Single sentence; binary:** This one is probably the simplest of the listed tasks. Examples are (as mentioned above) Sentiment Analysis (SST-2, Socher et al. 2013) or Linguistic acceptability judgements (CoLa, Warstadt et al. 2019).
- **Single sentence; multi-class:** Besides being slightly more challenging than binary classification, this one is probably also the more frequently occurring task since many problems naturally do not just have two possible solutions. Sentiment analysis with three categories (*negative*, *neutral*, *positive*) as well as many types of review data with "star-ratings", e.g. Amazon reviews (Ni et al., 2019), belong to this type of task.
- **Single sentence; multi-label:** This type of task is the rarest one, since it is included in none of the most popular benchmark data sets with natural language input. Nevertheless there are text-based data sets in the MULAN (<http://mulan.sourceforge.net/datasets-mlc.html>) repository, namely the "bibtex" data set (Katakis et al., 2008) and the "EUR-Lex" data set (Mencia and Fürnkranz, 2008) and there are multi-label problems used

2.2 Performance Measures

for Kaggle (<https://www.kaggle.com/>) competitions, e.g. the *Toxic comment classification challenge* (Jigsaw and ConversationAI, 2018).

- **Two sentences; binary:** If the input consists of more than one sentence, the task often refers to determining the relation between the two input sequences. Typically these are similarity (QQP, Shankar et al. 2017) and paraphrase tasks (MRPC, Dolan and Brockett 2005) as well as inference tasks (QNLI, RTE, WNLI, Wang et al. 2018).
- **Two sentences; multi-class:** The more complex type of task with a multiple sentence input encompasses similarity (STS-B, Cer et al. 2017) and inference (MNLI, Williams et al. 2017) tasks, which vary in complexity due to the size of the label set.
- **Two sentences; multi-label:** At the moment of writing this thesis, I am not aware of any existing data sets (commonly used for evaluation/benchmarking) taking two sentences as input and having a multi-label objective.

Section 2.2 will provide a more detailed overview on task-agnostic measures for *intrinsic evaluation* and task-specific measures for *extrinsic evaluation* as well as their advantages and drawbacks with respect to the tasks presented above.

2.2. Performance Measures

The language modeling task allows measuring model performance of many NLP (transfer learning) models in a way that is not related to any specific (classification) task. The only prerequisite for these measures to be applicable, is that the model needs to be able to perform language modeling. One might argue that this then does not meet the definition of "task-agnostic", but since language modeling is the most basic task that can be performed by all of the considered transfer learning architectures it is deemed to be sufficiently task-agnostic for this purpose.

Task-agnostic measures From a statistical point of view, a *language* can easiest be thought of as a data generating process \mathcal{L} with a discrete probability distribution which assigns a probability to every word/token from its vocabulary V :

$$P(W_i = w_i) = f(w_i) = p_i \quad \forall i = 1, 2, \dots, |V| \quad (2.1)$$

$$\text{with } f(w_i) \geq 0, \quad \sum_{i=1}^{|V|} f(w_i) = 1$$

But since this refers to the context of language and text, one is rather interested in describing and modeling the probabilities of *sequences* of words instead of single words themselves. Using the conditional probability of a word w_t at the t -th position of a sequence

$$P(w_t) = P(w_t | w_1, w_2, \dots, w_{t-1}) \quad (2.2)$$

one can write the probability of a whole sequence $s = (w_1, w_2, \dots, w_m)$ as follows:

$$P(s) = P(w_1, w_2, \dots, w_m) = \prod_{i=1}^m P(w_i | w_0, \dots, w_{i-1}), \quad (2.3)$$

where the special token w_0 denotes the beginning of the sequence, i.e. $P(w_1 | w_0) = P(w_1)$.

The language modeling task can be formulated as predicting these conditional probabilities of the upcoming word/token given the sequence of all previous words/tokens. Assume a language model \mathcal{M} was trained on a specific training corpus \mathcal{C} generated by a language \mathcal{L} . A held-out set, generated by the same language, can be used to calculate the *Perplexity*, measuring the degree of surprise of the model when confronted with a previously unseen sequence. Formally, the perplexity is defined as the inverse of the geometric mean of the conditional probabilities of all words in a sequence (given all previous words):

$$\begin{aligned} \text{Perplexity}(\mathcal{M}, s) &= \hat{P}(s)^{-\frac{1}{m}} = \sqrt[m]{\prod_{i=1}^m \frac{1}{\hat{P}(w_i|w_0, \dots, w_{i-1})}} \\ &= \sqrt[m]{\frac{1}{\hat{P}(w_1)} \cdot \frac{1}{\hat{P}(w_2|w_1)} \cdots \frac{1}{\hat{P}(w_m|w_1, \dots, w_{m-1})}} \end{aligned} \quad (2.4)$$

where $\hat{P}(\cdot)$ denotes the probability predicted by the Model \mathcal{M} (as opposed to the true, underlying probability P from the data generating process \mathcal{L}).

High predicted (conditional) probabilities for each word/token, which would signify a lower degree of surprise by the model, lead to a lower perplexity. Vice versa, if the model does not anticipate the occurrence of the words/tokens very well, i.e. low predicted probabilities, the perplexity will take a rather high value. The theoretical lower bound of the perplexity is 1, which is obviously an unrealistic scenario as the model would, in this case, be able to predict every token correctly with a certainty of 100%. This implies, that there would be only one possible continuation per starting word of a sequence, making the language (colloquially speaking) quite "boring". On the other end of the spectrum, the worst possible value for the perplexity would be the size of the vocabulary $|V|$. This would imply a random guess of the model with probability $\frac{1}{|V|}$ for each position in the sequence, meaning the preceding words do not convey any information at all (to the model). Since both of these extreme values are rather unrealistic, the following enumeration provides an overview on state-of-the-art perplexities for several data sets:

- For the *1B Word Benchmark* (Chelba et al., 2013), a corpus with a vocabulary size of $\sim 800k$, the state-of-the-art perplexity is **21.8** (Dai et al., 2019).
- For *WikiText-103* (Merity et al., 2016a,b), a corpus of high-quality articles from Wikipedia comprising a vocabulary of $\sim 270k$ words, state-of-the-art perplexity is **10.8** (Shoeybi et al., 2019). For the subset *WikiText-2* (Merity et al., 2016c) it is **18.3** (Radford et al., 2019).
- For the *Penn Treebank* (Marcus et al., 1994), a rather small corpus with $|V| \approx 10k$, state-of-the-art perplexity is **20.5** (Brown et al., 2020).

Entropy (Shannon, 1948) is a concept from information theory that is also applicable for measuring the quality of language models. Generally, the Entropy of a single random variable X is defined as²:

$$H(X) = - \sum_{x \in \mathcal{X}} P(x) \cdot \log_2(P(x)). \quad (2.5)$$

²The choice of the log's base is basically arbitrary, but using base 2 yields the entropy measured in *bits*.

2.2 Performance Measures

Since in the case of language modeling we are confronted with sequential data, this requires the extension of $H(X)$ to the *Entropy rate* of a sequence $s = (w_1, w_2, \dots, w_m)$ defined as

$$\frac{1}{m}H(s) = -\frac{1}{m} \sum_{s \in \mathcal{L}} P(s) \cdot \log_2(P(s)), \quad (2.6)$$

and finally considering sequences of theoretically infinite length in order to generalize to \mathcal{L} :

$$H(\mathcal{L}) = -\lim_{m \rightarrow \infty} \frac{1}{m} \sum_{s \in \mathcal{L}} P(s) \cdot \log_2(P(s)). \quad (2.7)$$

The concept of *Cross-Entropy* is defined using the actual probabilities $P(\cdot)$ from \mathcal{L} as well as the predicted ones $\hat{P}(\cdot)$ from a model \mathcal{M}

$$H(\mathcal{L}, \mathcal{M}) = -\lim_{m \rightarrow \infty} \frac{1}{m} \sum_{s \in \mathcal{L}} P(s) \cdot \log_2(\hat{P}(s)) = -\lim_{m \rightarrow \infty} \frac{1}{m} \log_2(\hat{P}(s)), \quad (2.8)$$

with the equality holding under the assumptions³ of the *Shannon-McMillan-Breiman Theorem* (Algoet and Cover, 1988; Cover and Thomas, 1991). Cross-Entropy itself is defined in the limit, but can be approximated using a sufficiently long (fixed length) sequence s , as in the latter part of Eq. (2.8). When being used to measure the quality of a character-level language model it is referred to as *Bits-per-Character (BPC)*, if the model is on a word-/token-level it is called *Bits-per-word/-token*.

Task-specific measures (binary) Naturally, there is a larger set of applicable measures which is partly determined by the type of task the model is applied to. But besides this, one always has to carefully consider further aspects, like e.g. the class distribution (balanced vs. imbalanced), when choosing a suitable measure for a combination of task and data set. This passage presents suitable measures for a range of different (classification) tasks. It is not an exhaustive list, but rather covers the measures which are necessary to understand the contributing publications and contains pointers to further measures for the interested reader.

		true condition	
		positive	negative
predicted	positive	TP	FP
	negative	FN	TN

Table 2.2.: Exemplary confusion matrix \mathbf{C} for a binary classification task.

The most straightforward performance measure for binary tasks is the plain accuracy, which measures the fraction of correctly classified instances. Following Tab. 2.2 it can be written as

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}, \quad \text{with } 0 \leq Accuracy \leq 1. \quad (2.9)$$

Since all correctly classified examples are given equal weight, using this measure in presence of a skewed distribution of the classes can lead to bias towards predicting the majority class⁴. A

³The central assumptions are stationarity and ergodicity of a stochastic process, here: the language \mathcal{L} .

⁴Consider a situation where 80% of the observations belong to one class. Even a completely useless classifier, predicting the majority class all of the time, would reach an accuracy of 0.8.

measure that alleviates this shortcoming of plain accuracy is the F_1 -Score which is defined as the harmonic mean of precision and recall

$$F_1 = 2 \cdot \frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}}, \quad \text{with } 0 \leq F_1 \leq 1 \quad (2.10)$$

and $\textit{precision} = \frac{TP}{TP + FP}$ and $\textit{recall} = \frac{TP}{TP + FN}$ according to Tab. 2.2.

Despite balancing better between different classification goals (i.e. precision and recall), there are still two drawbacks which should be considered when choosing a suitable measure. First, the F_1 -score takes into account both precision and recall, but it does so by giving equal weight to them. This neglects the (potentially) different importance/cost of different types of misclassification and can thus be problematic if there are differences. Second, it does not account for the number of TN making it still susceptible to problems in the presence of class imbalance. Mainly the latter aspect is addressed by the Mathew’s Correlation Coefficient (MCC; Matthews, 1975) since it is based on all four cells of the confusion matrix \mathbf{C} . It is calculated via the following formula

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \quad \text{with } -1 \leq MCC \leq 1. \quad (2.11)$$

The MCC is deemed the most suitable measure for (imbalanced) binary classifications tasks and is e.g. used as performance measure for many tasks from the GLUE benchmark collection (Wang et al., 2018).

Task-specific measures (multi-class) If the target variable possesses multiple classes, out of which only one can be the *true condition*, the confusion matrix \mathbf{C} looks as follows:

		true condition			
		class 1	class 2	...	class k
predicted	class 1	TP_1			
	class 2		TP_2		
	...			\ddots	
	class k				TP_k

Table 2.3.: Exemplary confusion matrix \mathbf{C} for a multi-class classification task.

It holds the correctly predicted cases for each class on the diagonal while all false predictions are located in the upper and lower triangular. *Accuracy* has to be calculated by summing up the values on the diagonal and dividing this sum by the number of observations n :

$$Accuracy_{multi} = \frac{1}{n} \sum_{j=1}^k TP_j \quad (2.12)$$

But similar to the binary case this measure struggles in the presence of class-imbalance. Since every observation is given equal weight in this calculation, a classifier which does well in predicting the predominant class(es) will achieve a high accuracy. Performance on classes with very few numbers of training examples (which are typically harder to predict) will conversely suffer.

2.2 Performance Measures

For the F_1 -Score there are two distinct ways to adapt this measure to the presence of multiple classes. One can either calculate TP , FP , FN for each class, use them to calculate separate F_1 -Scores and average them across all classes⁵ (called "macro-averaging") or one can aggregate TP , FP , FN over all classes first (called "micro-averaging"). For the former approach the F_1 is defined as

$$F_1^{macro} = \frac{1}{n} \sum_{j=1}^k F_1^{class\ j} = \frac{1}{n} \sum_{j=1}^k 2 \cdot \frac{precision_j \cdot recall_j}{precision_j + recall_j}, \quad \text{with } 0 \leq F_1^{macro} \leq 1, \quad (2.13)$$

$$precision_j = \frac{TP_j}{TP_j + FP_j} \quad \text{and} \quad recall_j = \frac{TP_j}{TP_j + FN_j},$$

whereas for the latter one it is calculated as follows:

$$F_1^{micro} = 2 \cdot \frac{precision_{micro} \cdot recall_{micro}}{precision_{micro} + recall_{micro}}, \quad \text{with } 0 \leq F_1^{micro} \leq 1, \quad (2.14)$$

$$precision_{micro} = \frac{\sum_{j=1}^k TP_j}{\sum_{j=1}^k TP_j + FP_j} \quad \text{and} \quad recall_{micro} = \frac{\sum_{j=1}^k TP_j}{\sum_{j=1}^k TP_j + FN_j}.$$

The macro-averaged version gives equal weight to each class and is thus well suited for imbalanced data sets. Contrary, the micro-averaged version gives equal weight to each observation which is why it is not well suited when labels are imbalanced (same reason as for the accuracy).

Gorodkin (2004) generalized correlation to the multivariate scenario, by defining R_K as a correlation coefficient applicable to K -dimensional data points. Further it was shown that by discretizing R_K , one obtains a multi-class equivalent for the MCC:

$$R_K = MCC_{multi-class} \frac{n \cdot tr(\mathbf{C}) - \sum_{k,l} \mathbf{C}_k \mathbf{C}_l}{\sqrt{n^2 \cdot \sum_{k,l} \mathbf{C}_k (\mathbf{C}^\top)_l} \sqrt{n^2 \cdot \sum_{k,l} (\mathbf{C}^\top)_k \mathbf{C}_l}} \quad (2.15)$$

where \mathbf{C}_k denotes the k th row, \mathbf{C}_l the l th column and \mathbf{C}^\top the transpose of \mathbf{C} .

Task-specific measures (multi-label) Besides that both (micro- and macro-averaged) versions of the F_1 -score can also be calculated for the multi-label case, there are other measures more specifically tailored to this situation. The *Subset Accuracy* is a measure which is calculated on the observation-level and is defined as the ratio of correctly classified observations (i.e. predictions for **all** classes are correct) and the total number of observations

$$Subset\ Accuracy = \frac{1}{n} \sum_{i=1}^n \mathbb{1}(P_i = Y_i), \quad (2.16)$$

where Y_i are the true and P_i the predicted labels. As one can imagine this is a rather harsh metric, as each observation for which not *every* class is predicted correctly is regarded as misclassified. Potential partial correctness of prediction is thus neglected by this measure, a problem which

⁵There exist two versions of the macro-averaged F_1 -Score, which are both commonly used. Here, the definition from above (arithmetic mean of class-wise F_1 -scores) will be used. According to the other definition one calculates the macro-averaged F_1 by first calculating arithmetic means of class-wise *precision* and *recall* and uses these values to calculate F_1^{macro} . Opitz and Burst (2019) evaluated the properties of these different existing versions.

could be alleviated by using one the different versions of the F_1 -Scores. For multi-label tasks, the F_1 -Score can furthermore be calculated one a per-sample basis

$$F_1^{sample} = \frac{1}{n} \sum_{i=1}^n \frac{2|Y_i \cap P_i|}{|Y_i| + |P_i|}. \quad (2.17)$$

Since this thesis is essentially about *Natural Language Processing* and this section just provides some background knowledge, further details are considered out of scope. The interested reader is (for the multi-label case) referred to to the excellent overview paper of [Gibaja and Ventura \(2014\)](#) and their tutorial on multi-label learning ([Gibaja and Ventura, 2015](#)).

2.3. Machine Learning Methods for Text Data

Representing text data While text is potentially more rich in information content than any other data source, traditional⁶ machine learning methods severely struggle when it comes to the analyses of this kind of data. This type of models typically expects the input \mathbf{X} to be of the format $n \times p$, where n is the number of observations and p is the number of covariates/features. Text, however, mostly occurs in sequences of variable length which is why it is subject to further pre-processing before machine learning methods can be applied to it. The easiest way to transform a corpus \mathcal{C} of M documents into a tabular data set is representing every document d as a "bag of its words". To obtain these representations, the following steps have to be taken:

1. The number of features is defined by the size of the vocabulary, i.e. $p = |V|$
2. Each of the p columns in the tabular data set belongs to one word from the vocabulary, oftentimes ordered alphabetically.
3. The number of observations n is equal to the number of documents, i.e. $n = M$
4. Every cell $x_{i,j}$ holds the word count of the j -th word in the i -th document.

The representation of a corpus of training examples through such a *Document-Term-Matrix* implicitly inherits the "**Bag-of-Words**" (BoW) assumption. This a very strong assumption, since its crucial point is that the word order of a sequence is completely disregarded and words are treated as independent from their context. The two exemplary sequences $s_1 = ("not", "bad", ", ", "actually", "quite", "nice")$ and $s_2 = ("not", "nice", ", ", "actually", "quite", "bad")$ show the drawback of this assumption in quite a vivid way. Despite conveying completely opposing meanings, both sequences would be represented by the exact same BoW-representation. One approach to mitigate this problem is the use of *n-grams*, which are sub-sequences of n successive words. In this context, the standard BOW-representations can be coined as a "Bag-of-Unigrams" (1-grams), while the use of "**Bigrams**" (2-grams) would mean to additionally consider all occurring sequences of two successive words. Further, also trigrams (3-grams), four-grams, etc. can also be used.

Nevertheless, this ability to capture context comes at a cost. Since many machine learning models tend to perform worse in the presence of an unfavorable relation between the number of observations n and the number of features p ($p \gg n$), one has to be careful at this point. The use

⁶"Traditional" machine learning methods are strictly separated from neural networks (cf. Sec. 2.4) in this thesis.

2.4 Neural network architectures

of n -grams leads to an enlargement of V and consequently also of p , while the number of documents (and thus n) remains constant. This leads to a trade-off between the ability of capturing local contexts (using n -grams) and the risk of having an unfavorable ratio of n and p . Methods to reduce the number of features in a reasonable way include the removal of stop-words (*words that convey little or no meaning*), stemming (*reducing a word to its "stem" by cutting of pre- and suffixes*) or lemmatization (*similar to stemming, but the "lemma" has to be an existing word*). For further implementation details the interested reader is referred to Bird et al. (2009, Ch. 3).

A further shortcoming of the naive BoW is the inherent assumption that the raw counts of a word reflect its importance in a corpus. Actually, the information content of a word (generally) decreases with increasing occurrence. This problem can be addressed by the "*term frequency – inverse document frequency*" (tf-idf) re-weighting scheme, which scales the raw counts of words (or n -grams) with their (logarithmic) inverse document frequency⁷

$$idf(w_i) = \log \left(\frac{n}{\sum_{d|w_i \in d} 1} \right). \quad (2.18)$$

At the time of writing this thesis, there exist more advanced and more efficient methods for representing words and sequences (cf. Sec. 2.5 and 2.6.2). This part is nevertheless included to provide context and to be referred to when stressing the advantages of the advanced methods.

Task-specific models for text data After pre-processing the data as described above, most of the standard as well as advanced machine learning models can consume BoW-based representations of dimension $n \times p$ as input and perform the task at hand. These models include (but are not limited to) methods like (regularized) regression models, tree-based models (e.g. CART, Random Forest), gradient-based methods (e.g. Boosting) or support vector machines. Depending on (i) the type of the target variable, (ii) the amount of data, (iii) the exact characteristics of \mathbf{X} and (iv) the chosen hyperparameters basically any of these architectures could be well suited for a given task and data situation.

2.4. Neural network architectures

Before dealing with NLP-related architectures, the basic⁸ concepts of artificial neural networks will be introduced. First, the functionality of a single neuron will be described, before it is embedded in the larger context of a (fully-connected) feed-forward neural network. Extensions towards deeper networks with multiple layers and the training procedure of neural networks will be provided in order to lay the foundation for more complex and deeper architectures in Ch. 2.5 and 2.6.

Single neurons A neuron can be thought of as the most basic unit of a network, performing exactly one single part of the calculation in a two-step procedure. Assume a neuron receives the vector $\mathbf{x} = (x_1, x_2, \dots, x_p)$ as an input. In a first step, this input vector is multiplied by a

⁷There exist different variants of *td-idf*, like sublinear scaling of the term frequency or smoothing of the inverse document frequency. See e.g. the implementation in Python's scikit-learn library (Pedregosa et al., 2011).

⁸For deeper insights, the respective chapters from Goodfellow et al. (2016) will be referenced where suitable.

(trainable) weight vector $\mathbf{w} \in \mathbb{R}^p$, the single elements are summed up and finally a (trainable) so-called bias term b is added:

$$f_{in} = \mathbf{w}^\top \mathbf{x} + b \quad (2.19)$$

In a second step, f_{in} is transformed by a non-linear *activation function* τ and subsequently emitted by the neuron:

$$f_{out} = \tau(f_{in}) = \tau(\mathbf{w}^\top \mathbf{x} + b) \quad (2.20)$$

Note, that while the activation function is a given function⁹ (e.g. sigmoid, softmax, tanh or ReLu), \mathbf{w} and b are model parameters that will be updated during the learning procedure.

Fully-connected feed-forward neural networks A neural network consists of multiple neurons (per layer). If each of the neurons in one layer is connected to each of the neurons in the previous layer, it is straightforward to extend the formulation from Eq. (2.19) and (2.20) to

$$\begin{aligned} \mathbf{f}_{in} &= \mathbf{W}\mathbf{x} + \mathbf{b} \\ \mathbf{f}_{out} &= \tau(\mathbf{f}_{in}) = \tau(\mathbf{W}\mathbf{x} + \mathbf{b}) \end{aligned} \quad (2.21)$$

for a whole layer¹⁰ instead of a single neuron, where in the weight matrix \mathbf{W} each row $\mathbf{W}_i \in \mathbb{R}^p$. Typically the first (*input*) layer holds the covariates, while the second (*hidden*) layer does the computations described above. The output layer is always adapted to the type of task which is to be solved by the network. This requires two crucial design choices for the final layer. First, the number of output neurons has to be suitable for the task, e.g. only one neuron for binary classification, since the probability for one class has to be predicted. Second, the activation function also has to fit the task, e.g. the sigmoid function for binary classification, since it squashes any real number to an interval between zero and one. This is why the activation function(s) of the hidden layer(s) do(es) not necessarily have to be the same as for the output layer. While the former can be considered a (tunable) hyperparameter, the latter is predetermined by the task. If there are multiple hidden layers stacked between the input and the output layer one speaks of *deep* neural networks. These models contain more trainable weights and have thus (generally) a higher learning capacity, but are obviously more expensive to train.

Training (deep) neural networks¹¹ Before the start of the training, the trainable weights are (potentially using some smart strategy) initialized with (small) random values. The training itself consists of two separate phases: The forward pass and the backward pass. During the forward pass, the input data flows through the computational graph defined by the network at the end of which a predicted output $f(\mathbf{x}, \theta) = \hat{y}$ is calculated. This prediction is, jointly with the true, known value y of the target variable, used to calculate the error using a chosen loss function $J(y, f(\mathbf{x}, \theta))$.¹² In order to minimize the loss, some version of (stochastic) gradient descent (SGD) is performed via Backpropagation (Rumelhart et al., 1986). After the training procedure is finished, the neural network can be used in order to make predictions for unseen inputs.

⁹For more details about activation functions, see Goodfellow et al. (2016, Ch. 6.3)

¹⁰Note, that the activation function τ is now applied element-wise.

¹¹This is a *very* high-level explanation, for detailed explanations see Goodfellow et al. (2016, Ch. 6 and Ch. 8)

¹²Common loss functions for classification tasks are binary or categorical cross-entropy.

2.5. Shallow Networks for Natural Language Processing

2.5.1. Neural probabilistic language model

Bengio et al. (2003) proposed a neural network based model for the language modeling task. In their architecture they did not rely on features like (tf-idf weighted) word or n-gram counts in the input layer, but utilized binary indicators for the presence/absence of words which are mapped to (trainable) continuous representations in the first hidden layer. This mapping is performed by matrix multiplication of a one-hot encoded vector \mathbf{x} of dimension $1 \times |V|$ with a look-up table matrix of dimension $|V| \times E$.¹³ Thus, the i -th row of this matrix can be regarded as the feature vector associated with the i -th word in the vocabulary. Note that the look-up table matrix in the first hidden layer is shared for every occurrence of a word, which means that a word will be represented by the same feature vector irrespective of (i) the position and (ii) the context it appears in. Since the model is trained on the language modeling task, it consumes a chosen number C of words and uses them to predict some word w_t in the sequence. The dimensionality E of the look-up table matrix as well as the context size C are hyperparameters of the model.

In order to predict the word at position t , the words w_{t-C}, \dots, w_{t-1} are first fed to the input layer of the network and are transformed to their vector representations $\mathbf{w}_{t-C}, \dots, \mathbf{w}_{t-1}$ from the look-up table matrix in the first hidden layer. These vector representations are subsequently concatenated into a context vector $\mathbf{x} = \text{concat}(\mathbf{w}_{t-C}, \dots, \mathbf{w}_{t-1})$ which is transformed by a *tanh* non-linearity in the second hidden layer before it is fed into the output layer with a softmax activation function producing a $|V|$ -dimensional vector of probabilities for the the word w_t :

$$P(w_t = w_i | w_{t-1}, \dots, w_{t-C}) = \frac{\exp(\boldsymbol{\eta}_{w_i})}{\sum_{w_j \in V} \boldsymbol{\eta}_{w_j}} \quad \text{with } \boldsymbol{\eta} = \mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{W}_3 \mathbf{x} + \mathbf{b}_2, \quad (2.22)$$

where $\boldsymbol{\eta}_{w_i}$ is the w_i -th element of $\boldsymbol{\eta}$ corresponding to the i -th word in the vocabulary.

The model is trained via stochastic gradient ascent maximizing the training corpus' penalized log-likelihood, which is equivalent to minimizing the negative penalized log-likelihood via SGD.

The authors showed the superiority of their model over different models based on n-grams by measuring the perplexity (cf. Ch. 2.2) on a held out test set on several different corpora. With respect to the rest of this thesis, the most important fraction of this model is the mapping of discrete units (*words* or *tokens*) to continuous representations (*embeddings*) which happens in the first hidden layer. Most of current research fundamentally builds on this idea, starting with the extremely influential work of Mikolov et al. (2013a,b,c,d) in the early 2010s.

2.5.2. Representation learning

Word2Vec Mikolov et al. (2013a) picked up on the idea of representing words as continuous vectors, but dropped the simultaneous objective of training a functioning language model.¹⁴ The authors solely focus on generating high-quality word embeddings while simultaneously employing computational tricks in order to avoid the high computational cost which is implied by the softmax

¹³The notation in this thesis differs (for the sake of consistency) from the one used by Bengio et al. (2003).

¹⁴Interestingly this behaviour can be observed at a later point in time (cf. Sec. 3.3). Devlin et al. (2019) also use (parts of) an existing type of architecture (Transformer, Vaswani et al. 2017) and "abuse" it in order to learn high-quality word representations, while thereby sacrificing the language modeling objective.

function in the output layer of the Neural probabilistic language model (NPLM). Mikolov et al. (2013a) specify the computational complexity of one single training example in the NPLM as

$$n \times E + n \times E \times H + H \times |V|, \quad (2.23)$$

where H is the dimensionality of the hidden layer. While n is typically around 5 – 20 words and E and H are typically three-digit or low four-digit numbers, $|V|$ is for most of the languages between 10.000 and 100.000 words. The most complex part ($H \times |V|$) could at this time already be efficiently reduced to $H \times \log_2(|V|)$ by using hierarchical softmax (Morin and Bengio, 2005), such that $n \times E \times H$ remained the most costly part of this model. Mikolov et al. (2013a) simply drop this expensive part and thus reduce the complexity of the model by a large margin. Furthermore, in a follow-up paper, Mikolov et al. (2013c) add the idea of noise-contrastive estimation (NCE; Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012) to this architecture, enabling it to learn high-quality word embeddings in a very efficient manner.

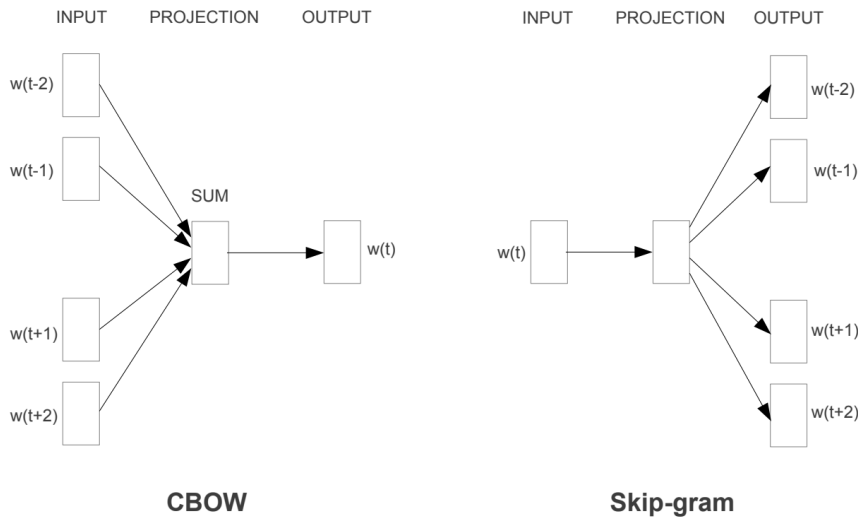


Figure 2.1.: The two word2vec model architectures proposed by Mikolov et al. (2013a), figures adopted from the original paper.

The two log-linear models (which are colloquially summarized under the umbrella term *word2vec*) are both simple feed-forward architectures which can be distinguished by their objective and hence also by how training examples are constructed from the training corpus. In the *Continuous Bag-of-Words (CBOW)* architecture, the objective is to maximize the average log probability of a center word given the surrounding in a defined context window of size $2 \cdot C$,

$$\frac{1}{m} \sum_{t=1}^m \log p(w_t | w_{t-C}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+C}) \quad (2.24)$$

$$\text{with } p(w_t | w_{t-C}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+C}) = \frac{\exp(\mathbf{w}_t^{\text{out}} \tau \mathbf{w}_C^{\text{in}})}{\sum_{i=1}^{|V|} \exp(\mathbf{w}_i^{\text{out}} \tau \mathbf{w}_C^{\text{in}})},$$

where the probabilities are calculated via the softmax function and \mathbf{w}_C^{in} is the summation of the word embeddings of all context words w_{t-C}, \dots, w_{t+C} (cf. left part of Fig. 2.1). It is important to distinguish between \mathbf{w}_t^{in} , which is the word embedding of w_t (i.e. the weights of the input/projection layer), and $\mathbf{w}_t^{\text{out}}$, which depicts the weights of the output layer for w_t .

2.5 Shallow Networks for Natural Language Processing

Employing the *Skip-Gram (SG)* architecture basically reverses the idea of *CBOW*, since one single center word is now used to predict the surrounding words in a context window of size $2 \cdot C$. Hence the objective (graphically depicted in the right part of Fig. 2.1) can be formulated as follows:

$$\frac{1}{m} \sum_{t=1}^m \sum_{-C \leq j \leq C, j \neq 0} \log p(w_{t+j}|w_t) \quad (2.25)$$

with $p(w_{t+j}|w_t) = \frac{\exp(\mathbf{w}_{t+j}^{\text{out}} \top \mathbf{w}_t^{\text{in}})}{\sum_{i=1}^{|V|} \exp(\mathbf{w}_i^{\text{out}} \top \mathbf{w}_t^{\text{in}})}$

Note, that the formulation of "maximizing the average log probability" is equivalent to minimizing the negative log-likelihood for both architectures. Introducing *Negative Sampling* (Mikolov et al., 2013c), based on the idea of NCE as mentioned above, further decreases the computational cost of both architectures. Negative Sampling trains the model to distinguish the true target word from a chosen number K of "negative examples" randomly drawn from a noise distribution $P_n(w)$. This replaces $p(w_{t+j}|w_t)$ in the *SG* objective by

$$\log(\sigma(\mathbf{w}_{t+j}^{\text{out}} \top \mathbf{w}_t^{\text{in}})) + \sum_{i=1}^K \log(\sigma(-\mathbf{w}_i^{\text{out}} \top \mathbf{w}_t^{\text{in}})) \quad (2.26)$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid activation function. As noise distribution $P_n(w)$, Mikolov et al. (2013c) (empirically) chose the unigram distribution of the training corpus raised to the power of $3/4$.

GloVe Another paradigm for learning vector representations was presented by Pennington et al. (2014), who rely on a matrix factorization algorithm. In a first step a global co-occurrence matrix \mathbf{X} with entries X_{ij} is constructed by counting how often a word w_j occurs in the context of a word w_i . The definition of context, i.e. the window size C , can be considered one of the most important hyperparameters of this model architecture. The vector representations are learned by minimizing the following objective:

$$\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} f(X_{ij}) \cdot \left(\mathbf{w}_i^{\text{in}} \top \mathbf{w}_j^{\text{out}} + b_i^{\text{in}} + b_j^{\text{out}} - \log(X_{ij}) \right)^2, \quad (2.27)$$

with $f(x) = \begin{cases} (x/100)^\alpha & \text{if } x < 100 \\ 1 & \text{otherwise} \end{cases}$,

where b_i^{in} and b_j^{out} are the bias terms of target and context word and $f(\cdot)$ is a weighting function. The resulting representations are denoted as *Global Vectors (GloVe)*, since this method relies on global instead of local (as in *CBOW* and *SG*) co-occurrences.

Following these important milestone publications, two important extensions were proposed based on the *word2vec* framework. First, Le and Mikolov (2014) extended the architecture with respect to being able to learn representations for whole paragraphs or documents instead of just for words. Second, Bojanowski et al. (2017) addressed to problem of out-of-vocabulary words by using subword embeddings. The remainder of this section will be dedicated to briefly introducing these two concepts, starting with the *doc2vec* framework (Le and Mikolov, 2014).

Doc2Vec The *doc2vec* framework presents a simple, but yet very powerful, extension to the word2vec framework. The basic idea is to enlarge the vocabulary V by a unique identifier for every document in the training corpus, such that the size of the new "vocabulary" equals $|V| + M$. The *Distributed Bag of Words version of Paragraph Vector (PV-DBOW)* architecture resembles the *SG word2vec* architecture, since a single document identifier is used as "center word" to predict the words in the context window (i.e. the whole document, cf. left part of Fig. 2.2). Vice versa, the *Distributed Memory Model of Paragraph Vectors (PV-DM)* architecture aims at predicting a center word given the surrounding context word plus the document identifier. It is thus exactly alike to the *CBOW* version of word2vec, with the document identifier as only difference. According to Le and Mikolov (2014), the identifier "acts as a memory that remembers what is missing from the current context", hence the name "distributed memory".

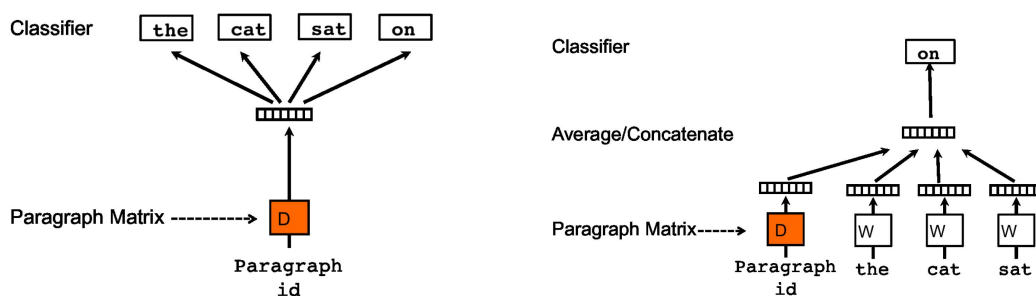


Figure 2.2.: The two doc2vec model architectures (PV-DBOW on the left, PV-DM on the right) proposed by Le and Mikolov (2014), figures adopted from the original paper.

FastText This algorithm was introduced by Bojanowski et al. (2017), who modified the Skip-Gram architecture by adding character n-grams ("subwords") to the vocabulary. By doing so, not only the center word itself, but also its character n-grams¹⁵ are used as model input. Generating character n-grams ($n = 3$) for e.g. the word "hello" would lead to the following decomposition:

$$\text{hello} \Rightarrow \langle \text{he}, \text{hel}, \text{ell}, \text{llo}, \text{lo} \rangle, \langle \text{hello} \rangle,$$

where the special tokens "<" and ">" denote word boundaries and the word itself is also kept. This leads to a replacement of the dot product between the vector representations of w_t and w_{t+j} in Eq. (2.25) and (2.26) by the sum of the dot products between the vector representations of the character n-grams of w_t and w_{t+j} . Let \mathcal{G}_{w_t} denote the set of character n-grams appearing in w_t (including w_t itself, see above), such that the scoring function looks as follows:

$$\sum_{g \in \mathcal{G}_{w_t}} \mathbf{w}_{t+j}^{\text{out}} \top \mathbf{w}_g^{\text{in}}. \quad (2.28)$$

By incorporating n-grams, the model now also learns embeddings for these n-grams and is thus highly likely able to represent previously unseen words by combining different n-gram embeddings. This property of being able to represent a basically open vocabulary by a fixed-size vocabulary of subword tokens is a very central concept for (i) neural machine translation architectures (which are out of the scope of this thesis) and (ii) for transfer learning architectures (cf. Ch. 3). While simply adding a set of character n-grams to the vocabulary is a rather heuristic way of achieving this,

¹⁵ n is a hyperparameter determining how fine-grained a word will be disassembled. In practice, a *range* is defined for n by specifying the upper and lower boundary, meaning n_1 - to n_2 -grams (e.g. 3- to 5-grams) are used.

2.6 Deep Learning for Natural Language Processing

more elaborated data-driven procedures for efficiently constructing vocabularies will be introduced in Sec. 2.6.2.

Algorithm	Authors	Pre-training Corpus	Languages
word2vec	Mikolov et al. (2013a)	Google News data set	English
GloVe	Pennington et al. (2014)	Wikipedia (amongst others)	English
FastText	Bojanowski et al. (2017)	Wikipedia (amongst others)	157+

Table 2.4.: An overview on the resources for the pre-trained word embeddings generated using the different algorithm presented in the section.

Regarding the computational complexity of the presented algorithms for representation learning, the following can be stated: In general it is feasible to train these architectures on standard consumer machines in a reasonable amount of time. There exist stable implementations for multiple programming languages, including the `gensim` module (Řehůřek and Sojka, 2010) for Python or the `text2vec` package (Selivanov et al., 2020) for R. Nevertheless pre-trained versions were also made available by the authors, which makes the respective vector representations usable as out-of-the-box tool for a variety of different tasks. Tab. 2.4 holds the information on where to obtain the pre-trained embeddings.

2.6. Deep Learning for Natural Language Processing

Neural network architectures in general have been introduced in Sec. 2.4 and shallow neural networks have been employed in the models presented in Sec. 2.5. As already briefly mentioned, neural networks can consist of *multiple* hidden layers of arbitrarily complex form. These architectures are commonly referred to as *Deep Learning Models*. Since neural networks are inherently modular architectures, it is straightforward to extract the key idea of the embedding models (i.e. the projection layer, holding the embeddings) and to combine it with deeper, more complex architectures. The additional parts can include tailored (combinations of) activation functions or task-specific output layer activations in order to make the models end-to-end trainable for downstream tasks. This section will present two types of neural architectures which have shaped the way deep learning models were applied to NLP tasks in the past decade. Starting with recurrent neural networks (RNN) in general, two wildly popular variations, namely Long-Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU; Cho et al., 2014), will be introduced briefly. Subsequently the Attention mechanism (Bahdanau et al., 2014), a major innovation for improving RNNs, as well as the *Transformer* architecture (Vaswani et al., 2017), the backbone of all transfer learning architectures in Ch. 3, is introduced and will be explained in thorough detail.

2.6.1. Recurrent architectures

While conventional fully-connected feed-forward neural networks (FFNNs) probably represent the most basic type of a deep network architecture, the above mentioned recurrent architectures are specifically tailored for processing sequential data, such as language. A FFNN typically consumes and processes all features of a given example at once, which may not necessarily be desirable when

it comes to modeling sequences. Instead, RNNs only consume one "feature" of the input sequence (x_1, \dots, x_p) at a time, with the "features" being ordered words, i.e. (w_1, \dots, w_m) .

The network itself can be defined using *hidden states*

$$h_t = \tau(h_{t-1}; w_t; \theta_{hidden}), \quad (2.29)$$

which are continuously re-used for computing the subsequent hidden state of the model.¹⁶ Designing the network this way, permits it access to all previous words w_1, \dots, w_{t-1} in the sequence when computing the hidden state for the word at position t . The output can (but does not necessarily have to) be computed at each time step:

$$o_t = \tau(h_t; \theta_{out}) \quad (2.30)$$

Computation of an output is executed at each time step when performing token-level tasks (e.g. Named Entity Recognition), whereas for sequence-level tasks (e.g. Sentiment Classification) it is usually only issued at the last time step. Also note, that the parameters θ_{hidden} are (i) shared across all time steps and (ii) distinct from the parameters θ_{out} of the output layer.

The parameter sharing across time steps is both a benefit, when considering the parsimony of deep learning architectures, as well as a threat, when it comes to the stability during the process of the optimizing procedure via (stochastic) gradient descent. Since the same mathematical operation (cf. Eq. 2.29) is performed using the same parameters at each time step of the sequence, gradients will be identical and thus be raised to the power of the length of the sequence. For gradients which are somewhat further away from an absolute value of one, this will cause them either to *vanish* (if the absolute value is smaller than one) or to *explode* (if the absolute value is larger than one) further back through the network. The vanishing gradients problem makes learning difficult in general, since the overall direction of suitable parameter updates becomes unclear. The exploding gradients problem on the other hand, leads to an unstable learning procedure and can be dealt with using methods like gradient clipping (Pascanu et al., 2013).

The LSTM (Hochreiter and Schmidhuber, 1997) was initially proposed to deal with the vanishing gradients problem and is thus able to model dependencies over a much longer range compared to vanilla RNNs. This architecture stands out by the introduction of (gated) self-loops defined as follows:

$$\begin{aligned} f_t &= \sigma(h_{t-1}; x_t; \theta_{forget}) \\ i_t &= \sigma(h_{t-1}; x_t; \theta_{input}) \\ h'_t &= \tanh(h_{t-1}; x_t; \theta_{candidate}) \\ o_t &= \sigma(h_{t-1}; x_t; \theta_{output}) \end{aligned} \quad (2.31)$$

The forget gate f_t determines the information kept in the long-term memory c_t (cf. Eq. 2.32), while the input gate i_t , together with the candidate state h'_t , determines what is added. The output gate is used to compute the next hidden state using also the long-term memory c_t (cf. Eq. 2.32). The newly introduced long-term memory c_t allows the gradient to "travel" further back in time, while h_t can be thought of as a short-term memory:

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot h'_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad (2.32)$$

¹⁶The initial hidden state h_0 is initialized together with the other parameters θ of the network.

2.6 Deep Learning for Natural Language Processing

The GRU (Cho et al., 2014) is a slightly less complex alternative to the LSTM, achieving an oftentimes competitive performance at a somewhat lower computational expense.

All of the presented architectures are able to take into account the sequential nature of text by processing it in a left-to-right fashion and thus taking into account the left-side context w_1, \dots, w_{t-1} when processing w_t . This makes sense when considering the "causal" process of reading or writing, but for some tasks (e.g. machine translation) internal representations of the network depending on context from both sides might make sense. Bidirectional RNNs (biRNNs, and biLSTMs/biGRUs respectively) solve this issue by simultaneously running a RNN (LSTM/GRU) on the sequence in reversed order, thus conditioning on the right-side context only. Concatenating the two internal representations from the forward and the backward RNN hence results in representations for each token that are conditioned on both sides of the context.

When performing sequence-to-sequence modeling on text data (e.g. Machine Translation or Question Answering), the model needs to be allowed to produce an output sequence of length m_{out} dissimilar to the length m_{in} of the input sequence. This requirement can be fulfilled by using *encoder-decoder architectures*¹⁷ (Sutskever et al., 2014), where the encoder network maps the input sequence to a fixed-size internal representation. The decoder network is subsequently conditioned on this internal representation to generate the output sequence. Since the internal representation is assumed to contain the information about the "context" (i.e. the complete input sequence) this can lead to an information bottleneck in case of an unfavorable ratio of input sequence length to dimension of the internal representation.

Bahdanau et al. (2014) picked up on this limitation in the context of machine translation and alleviated it by proposing the *Attention* mechanism, which allows the decoder network to access all hidden layer representations ($h_1, \dots, h_{m_{in}}$) of the encoder network instead of just one single fixed size representation. The basic idea of this mechanism is to use a weighted combination of the hidden layer representations as context representation for computing the decoder hidden state s_i . The weights are determined by the similarity between the previous decoder hidden state s_{i-1} and the elements in $(h_1, \dots, h_{m_{in}})$. This approach can be formalized as follows:

1. Calculate the alignment: $e_{ij} = a(s_{i-1}, h_j)$ with $a(\cdot)$ as the *alignment model*.
2. Softmax-Normalization of the weights: $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{m_{in}} \exp(e_{ik})}$.
3. Calculate the context representation: $c_i = \sum_{j=1}^{m_{in}} \alpha_{ij} \cdot h_j$.

The alignment model mentioned in the first step can be thought of as a hyperparameter of the model, just like any other activation function inside a neural network. Luong et al. (2015), besides extending and generalizing the Attention mechanism to Global and Local Attention, provided an overview on different options for the alignment model:

$$a(s_{i-1}, h_j) = \begin{cases} s_{i-1}^T h_j & \text{dot} \\ s_{i-1}^T W_a h_j & \text{general} \\ v_a^T \tanh(W_a [s_{i-1}^T h_j]) & \text{concat} \end{cases} \quad (2.33)$$

While Bahdanau et al. (2014) employed the variant which is referred to as *concat* in Eq. (2.33), Luong et al. (2015) show the superiority of the other, much simpler variants.

¹⁷Note, that this type of architecture is not limited to its use in conjunction with RNNs, but can also be used with other types of neural networks as to be shown in Section 2.6.2.

2.6.2. The Transformer

(Multi-Head) Self-Attention The Attention mechanism presented in Sec. 2.6.1 can be viewed in a more general way by introducing the notion of *keys*, *queries* and *values*. Attention then simply means to compare a *query* to a set of *keys* in order to determine weights which can be used to compute a weighted sum of some *values*. Translating these concepts to the Attention mechanisms proposed by Bahdanau et al. (2014) and Luong et al. (2015) makes the previous decoder hidden state the *query* and all the encoder hidden layer representations simultaneously the *keys* and the *values*. Generalizing from this has essentially two implications:

- *Keys* and *values* do not necessarily have to be the same, any (meaningful) vector representations are possible.
- *Query* and *keys* can refer to the same underlying sequence, this will be introduced as Self-Attention in this section.

Vaswani et al. (2017), with the goal in mind to reduce the number of sequential computations in a network, proposed the Transformer architecture. This model framework still has an encoder-decoder structure as described in Sec. 2.6.1, but instead of employing RNNs as encoder and decoder networks, the authors solely rely on the so-called "Self-Attention" mechanism. Their *Scaled dot-product Attention* mechanism can be formalized as follows:

1. Mapping of the input sequence¹⁸ $(w_1, \dots, w_{m_{in}})$ to E -dimensional input embeddings $(x_1, \dots, x_{m_{in}})$.
2. Linear projection of each input embedding to a query-vector q_i , a key-vector k_i and a value-vector v_i of dimensions d_q, d_k, d_v (where $d_q = d_k$).
3. This results in three matrices: $\mathbf{Q} \in \mathbb{R}^{m_{in} \times d_k}$; $\mathbf{K} \in \mathbb{R}^{m_{in} \times d_k}$; $\mathbf{V} \in \mathbb{R}^{m_{in} \times d_v}$
4. Calculate the alignment: $a(\mathbf{Q}, \mathbf{K}) = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}$
5. Softmax-Normalization and Multiplication with \mathbf{V} : $\text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$

The output of the Self-Attention mechanism can be described as a new, contextualized d_v -dimensional embedding for every token w_i from the input sequence. In the Transformer, these computations are executed h times in parallel by so-called *Attention Heads*. This is implemented by letting the model learn h different projection matrices (as described in the second step) which project the input embeddings into different sub-spaces of the original embedding space. The dimensions d_k and d_v are thereby chosen such that $d_k = d_v = E/h$, which again ensures an embedding size of E after performing Multi-Head Self-Attention and concatenation the different d_v -dimensional embeddings resulting from the h different heads.

Since this mechanism does not process the input in a sequential fashion (as opposed to RNNs, cf. Sec. 2.6.1) all information about the word order would be lost. In order to prevent this from happening, Vaswani et al. (2017) add E -dimensional *positional encodings* to the input embeddings before linearly projecting them to \mathbf{Q} , \mathbf{K} and \mathbf{V} . After experimenting with learned and fixed positional encodings, they decided to use fixed one composed of sine and cosine functions.

¹⁸Before feeding the input sequence to the model, Vaswani et al. (2017) apply a specific tokenization algorithm (BPE; Gage, 1994) producing subword units. Further details on tokenization will be provided in Sec. 3.1.

Encoder-Decoder Architecture The encoder network of the Transformer is constructed by stacking six identical layers, where each of these layers is composed of a Multi-Head Self-Attention module and an FFNN (cf. Fig. 2.3). Both of these components are supplemented by (i) residual connections (He et al., 2016) and (ii) subsequent layer normalization (Ba et al., 2016).

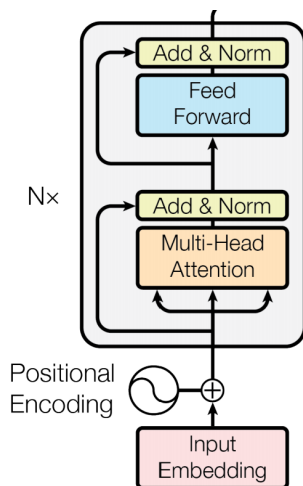


Figure 2.3.: The encoder network of the Transformer architecture proposed by Vaswani et al. (2017), figure adopted from the original paper.

Unconstrained employment of Self-Attention only makes sense in the encoder network (similar to biRNNs), since it would destroy the auto-regressive property of the decoder required for the machine translation task it was proposed for. Therefore, in the decoder Vaswani et al. (2017) employ a constrained version of the Self-Attention mechanism, namely *Masked Self-Attention* (sometimes also referred to as "*causal*" Attention). This modification prevents information flow from right to left by "masking" all tokens on the right-hand side of the current position by setting the scaled-dot product attention weights to $-\infty$ (resulting in attention weights of zero after softmax normalization). Additionally, the first layer of the decoder is an embedding layer mapping the tokens of the sequence to their output embeddings (which can be distinct from the encoder's input embeddings) and offsetting them by one position to the right. This offset, together with the Masked Self-Attention, ensures that only tokens which have a strictly lower position index can be used for prediction. Similar to the encoder, the decoder adds positional encodings and employs (Masked) Multi-Head Self-Attention as well as an FFNN (again each of them supplemented by residual connections and layer normalization). Between these two sub-modules another Multi-Head Self-Attention module is added, operating on the final *keys* and *values* from the encoder and the *queries* from the decoder. This part of the model grants the decoder full access to the contextualized input sequence. Vaswani et al. (2017) stack six identical layers for constructing the decoder network, which makes the complete architecture look as depicted in Fig. 2.4.

This architecture was initially proposed to solve a machine translation task, which is why the output layer of the network is a softmax layer generating a probability distribution over all tokens from the vocabulary. In subsequent research nevertheless, hardly ever the complete Transformer architecture was used again, but rather either the encoder or the decoder. For many models that were built on the basis of the Transformer encoder, the authors came up with a meaningful task which enabled training the encoder-only model in order to obtain high-quality, contextualized representations.

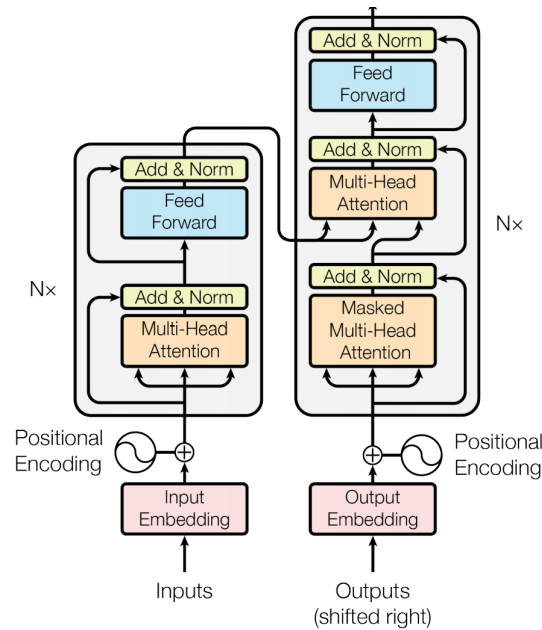


Figure 2.4.: The encoder and the decoder network of the Transformer architecture proposed by Vaswani et al. (2017), figure adopted from the original paper.

Computational implications One of the key contributions of this architecture is the complete abandonment of recurrence in the network. Substituting the recurrence mechanism by Self-Attention reduces the number of sequential operations necessary to model a sequence of length m from $O(m)$ to $O(1)$. The first implication of this drastic reduction is the reduced maximum path length between two arbitrary positions in the input and the output. It can be seen as an indicator for the ease with which a model is able to learn long-range dependencies and is also reduced from $O(m)$ to $O(1)$.

The second implication is the change in the degree to which a model is parallelizable. The $O(m)$ required sequential operations in a recurrent architecture clearly hinder parallelization, whereas in architectures based on the Self-Attentions mechanism the whole sequence can be processed in parallel due to $O(1)$ required sequential operations.

This property to be highly parallelizable helps in turn (partly) to overcome the drawback of architectures based on the Self-Attention mechanism. Due to the necessity to calculate the attention scores for a token with *all* the other tokens in the sequence, the computational complexity $O(m^2 \cdot E)$ of the Self-Attention mechanism scales quadratically with the sequence length m . Compared to the complexity $O(m \cdot E^2)$ of recurrent architectures this depicts a drawback as soon as the sequence length exceeds the dimensionality of the embedding layer. This quadratic scaling is still a problem nowadays and has led to a variety of different approaches proposing different amendments to the vanilla Self-Attention mechanisms. These proposed changes include (but are not limited to) recurrence mechanisms (Dai et al., 2019; Rae et al., 2019), low-rank approximation or kernel methods (Choromanski et al., 2020; Wang et al., 2020) or the use of specific (sparse) Attention patterns (Zaheer et al., 2020; Kitaev et al., 2020; Beltagy et al., 2020). This categorization is adopted from Tay et al. (2020), who provide a comprehensive overview on the methodologies of the aforementioned architectures and further approaches.

3. (Sequential) Transfer Learning

The paradigm of Transfer Learning is especially appealing for NLP applications since all of them are based on a natural language input. This common input across tasks makes it similar to the field of Computer Vision, where the inputs are mostly images (i.e. tensors of pixels) or videos (i.e. time series of images). In Computer Vision, Transfer Learning has become the de-facto standard with the introduction of ImageNet (Deng et al., 2009), a large-scale data set of 3.2 million images belonging to over 5.000 categories. Interestingly, up until the time of writing this thesis, there has been no comparable data set for NLP which is as influential as ImageNet is for Computer Vision.¹

The remainder of this chapter will be organized as follows: First, the general taxonomy of Transfer Learning will be established so that all subsequently presented concepts and architectures can be adequately categorized. Sec. 3.1 lays out the concept of Self-Supervised Pre-training, which depicts the foundation for all sequential transfer learning models. "Self-Supervised" in this case refers to a lack of necessity of externally labeled data, since the labels can be generated from the data itself. Two other important aspects of pre-training, corpora and computational resources, will be addressed as well. Hereafter different architectures will be introduced in the temporal order of their proposal, differentiating between feature-based transfer learning approaches (Sec. 3.2) and approaches relying on fine-tuning of complete pre-trained architectures (Sec. 3.3).

In order to establish a general taxonomy, the work of Ruder (2019), who adapted the general notation of Pan and Yang (2009) to NLP, will serve as the foundation. Ruder (2019) differentiates between the concepts of *Transductive Transfer Learning* and *Inductive Transfer Learning*, both of which will be briefly explained in order to allow for a better placement of the contributions of this thesis in the research context.

Transductive Transfer Learning With this concept, Ruder (2019) refers to transfer learning in situations where labeled data are solely present in the so-called *source domain* and the task is the same one across domains. Again, two areas can be subsumed under this umbrella term:

- If *domain* refers to an actual thematic concept, like for example different product categories when talking about review data, the learning problem can be referred to as *Domain Adaptation*. In this case, the model is pre-trained on a (labeled) training corpus from the source domain and is subsequently used for inference on data from the target domain.
- *Cross-lingual Learning* refers to the situation where *domain* actually describes a language. It aims at transferring knowledge gained from pre-training on (labeled) data from a source language to performing the same task on data from the target language.

¹My personal intuition why this might be the case, is that an image is a much more universal type of data than written text. There are probably much less diverse examples needed in order to visually represent concepts, like e.g. a *dog*, than there are when representing this concept with written text (considering different languages, paraphrases and writing styles). This makes it harder to create one universal pre-training resource for text.

Inductive Transfer Learning When the knowledge transfer happens across tasks, Ruder (2019) frames this as Inductive Transfer Learning. Here, labeled data is only present in the *target domain*, which is why it is mostly combined with *general domain* pre-training on a self-supervised task.

- *Multi-task Learning* denotes the approach to train a model with the ability to learn multiple tasks simultaneously, i.e. there are multiple target tasks for which labeled data are present.
- If tasks are not learned simultaneously, but in a sequential fashion, one speaks of *Sequential Transfer Learning*. This often also implies the presence of multiple models, which are fine-tuned separately starting from identical copies of a pre-trained architecture.

3.1. Self-Supervised Pre-training

In general, learning algorithms are broadly classified into two² distinct categories: Unsupervised and Supervised Learning. While the first one mostly refers to clustering or pattern detection algorithms, which do not require external labels attached to the training samples, the latter one, subsuming various regression or classification algorithms, explicitly requires these labels. Regarding the pre-training procedure, which large language models are subject to, neither of these two categories are perfectly adequate. On the one hand side, plain unannotated text corpora without any external labels can be seamlessly used for pre-training, which makes the learning problem look unsupervised. On the other hand, smart definitions of tasks in conjunction with data augmentation techniques allow generating labeled training examples from the plain corpus itself. The actual training is subsequently carried out in a supervised fashion, most of the time as a classification task. Overall, these two properties required coming up with a distinct name for this learning problem, which is why it is mostly referred to as *Self-Supervised Learning*.

Objectives Designing a suitable pre-training objective is crucial for a successful pre-training step, since this predominantly influences the ability to effectively transfer knowledge into the model's weights. During the course of this thesis, already a couple of objectives were presented without explicitly naming them as self-supervised pre-training objectives: The *language modeling task* (cf. Sec. 2.2 and 2.5), the *SG* and *CBOW* objective (cf. Sec. 2.5) as well as the *PV-DM* and the *PV-DBOW* objective (cf. Sec. 2.5) can be classified as such. Further self-supervised objectives will be introduced during the course of this chapter together with the architectures using them for pre-training.

Tokenization An important pre-step, before raw text data can be entered into a model, is tokenization. While machine learning models trained for single tasks (cf. Sec. 2.3) often employ custom tokenization schemes, e.g. simple whitespace tokenization, a more subtle approach has to be taken when it comes to transfer learning. The two main requirements of a tokenization scheme in order to be suitable for transfer learning are

- (a) that it should be able to cover the whole vocabulary of the language(s)/domain(s) it is intended to be used for, not just the vocabulary seen during training, and

²Note, that the field of Reinforcement Learning is intentionally omitted here, since it does not (yet) play a major role in NLP and is not relevant for the subsequent explanations. But one can surely argue that a classification of all learning algorithms into these three categories is *in general* more adequate.

3.1 Self-Supervised Pre-training

- (b) it should be perfectly reproducible, since the subsequent use of the pre-trained model crucially depends on the employment of the exact same tokenization.

Overall, these two points also make the tokenization scheme an important design choice (including its hyperparameters) of an architecture. A slightly more elaborated, but still kind of heuristic, tokenization scheme is the one used by FastText (cf. Sec. 2.5) where a subword tokenization based on character n-grams is employed.

BytePair Encoding (BPE; Gage, 1994) is an algorithm originally proposed for data compression and was adapted to be used for neural machine translation by Sennrich et al. (2016). This conceptually appealingly simple algorithm needs to be initialized with a starting token vocabulary, which is in most cases just the set of all characters plus an end-of-word token in order to control word boundaries. Subsequently, co-occurrences of tokens are counted (without crossing word boundaries) and the most frequently co-occurring token pair is merged to a new token, which is then added to the vocabulary (e.g. "A" + "B" → "AB"). This process is repeated until a pre-defined vocabulary size is reached. The WordPiece algorithm (Schuster and Nakajima, 2012; Wu et al., 2016), which is heavily used in current SOTA architectures, works in a related way, with the major difference that a language model is trained on the initial vocabulary and the increase of the likelihood in the training data is used as a criterion for merging two tokens. Additionally to the vocabulary size, the incremental increase of the likelihood can also serve as stopping criterion for this algorithm. SentencePiece (Kudo and Richardson, 2018) represents a pipeline-style alternative which does not require any language-specific pre-processing prior to tokenization, but consumes complete sequences. Under the hood, either BPE or a unigram language model are applied for tokenization. Just recently, Clark et al. (2021) proposed an architecture which directly consumes character sequences and thus poses an interesting alternative to the data-driven tokenization algorithms. It will be described in more detail in Sec. 11.

Text corpora Since there are no requirements to pre-training corpora regarding external labels, the most important criteria are size, diversity and quality.³ As already mentioned, there is no such standard pre-training data set as ImageNet is for Computer Vision, but still there are some corpora which are used more often than others. The English Wikipedia and Wikitext-103 (Merity et al., 2016a,b) represent examples for high quality data, while different subsets of CommonCrawl, WebText (Radford et al., 2019) or OpenWebText (Gokaslan and Cohen, 2019) are representatives for rather large corpora. Linguistic diversity is introduced by using corpora like e.g. the BooksCorpus (Zhu et al., 2015), the 1B Word Benchmark (Chelba et al., 2013) or the Stories corpus (Trinh and Le, 2018). A more in-depth discussion on the use of pre-training corpora can be found in the first contribution to this thesis (cf. Ch. 4).

Computational resources As mentioned in Sec. 2.6, current SOTA deep learning architectures require a rather large amount of computational power, especially when pre-trained on sometimes > 100 GB of plain text. Large-scale pre-training requires the use of (at least multiple) GPUs, but even better TPUs⁴, whereas the task-specific fine-tuning step is relatively cheap (at least if compared to the pre-training). The use of computational resources for SOTA architectures as well as the resulting implications are discussed more in depth in the contribution in Ch. 4.

³Quality here refers to measurable quantities like e.g. grammatical correctness or percentage of spelling mistakes (cf. Kiefer (2019)) since other criteria like writing style or stylistic beauty are subjective and hardly measurable.

⁴More information on Tensor Processing Units: <https://cloud.google.com/tpu/>

3.2. Feature-based Transfer Learning

After the language model has been pre-trained, there are two very distinct proceedings which will be described in further detail in this chapter. When the pre-trained architecture is utilized *as is*, i.e. the model weights are not further altered, but used as features in a second architecture building thereupon, one speaks of *feature-based transfer learning*. On the other hand, when the pre-trained weights are further adapted, this is called the *fine-tuning approach*. A third direction, which is recently emerging, can be categorized as *low-shot learning* and will be touched on briefly in Sec. 11.

Without explicitly naming it, some representatives of the class of feature-based transfer learning approaches have already been introduced in Sec. 2.5. All types of (sub)word or document embedding algorithms are used in such a fashion, since the (static) representations are extracted from the pre-trained model and subsequently used for some specific task at hand. A severe shortcoming of all these algorithms is the missing contextuality in the representations they produce. All of these architectures are only able to learn *one single* embedding per word/token which is independent of the context it appears in.

A more recent algorithm for feature-based transfer learning is able to overcome this shortcoming, by employing a recurrent architecture for contextualizing the embeddings: **Embeddings from Language Models** (ELMo; Peters et al., 2018) is an architecture consisting of two biLSTM layers preceded by a (character-based) token embedding layer which is pre-trained on a forward and a backward language modeling task. Denoting the context-independent representation from the embedding layer as x_k^{LM} and the context-dependent internal representations as $\vec{\mathbf{h}}_{k,j}^{LM}$ and $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ for the k -th token in the j -th layer, the complete set of representations can be written as

$$\begin{aligned} R_k &= \left\{ \mathbf{x}_k^{LM}, \vec{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, 2 \right\} \\ &= \left\{ \mathbf{h}_{k,j}^{LM} \mid j = 0, 1, 2 \right\}, \text{ where } \mathbf{x}_k^{LM} = \mathbf{h}_{k,0}^{LM} \text{ and } \mathbf{h}_{k,j}^{LM} = [\vec{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM}]. \end{aligned} \quad (3.1)$$

In order to train a downstream model including the pre-trained ELMo embeddings R_k as features, task-specific parameters $\Theta^{task} = (\gamma^{task}, s_j^{task})$ are included and subsequently learned, resulting in the following notation for a task-specific ELMo model:

$$\text{ELMo}_k^{task} = E \left(R_k; \Theta^{task} \right) = \gamma^{task} \sum_{j=0}^L s_j^{task} \mathbf{h}_{k,j}^{LM}, \quad (3.2)$$

where $E(\cdot)$ denotes the task-specific ELMo embedding. The parameters are (softmax-normalized) weights s_j^{task} for weighting the ELMo embeddings and a task-specific scaling parameter γ^{task} .

In order to prove the effectiveness of the pre-trained ELMo embeddings, Peters et al. (2018) selected six benchmark tasks which they, in a first step, trained a (task-specific) baseline model on. In a second step they enhanced this baseline by adding ELMo to the architecture, resulting in a notable increase of the baseline performance as well as in an increase of the SOTA results on these tasks at this point in time (cf. Tab. 1 in Peters et al., 2018)

Nevertheless, this approach shows two major disadvantages: First, ELMo representations can not be further adapted to specific tasks or domains since just the task-specific weighting- and scaling parameters are trainable. Second, having separate forward and backward representation may hinder deep contextualization. Both of these shortcomings will be tackled by the architectures introduced in the upcoming Sec. 3.3.

3.3. Fine-tuning approach

Regarding the *fine-tuning approach* one has to take into consideration one particular model that has changed much of the NLP landscape. **Bidirectional Encoder Representations from Transformers** (BERT; Devlin et al., 2019) is the first architecture simultaneously capturing bidirectional context and relying on the Transformer architecture. Architectures introduced before BERT are either *feature-based* bidirectional contextual models (ELMo; Peters et al., 2018) or unidirectional contextual LSTM- (ULMFiT; Howard and Ruder, 2018) or Transformer-based (GPT; Radford, 2018) models relying on the *fine-tuning paradigm*. The algorithms designed in the time period after BERT either tried to alter and improve the BERT architecture itself or were in some way fundamentally different while still heavily inheriting from BERT. During the remainder of this chapter, these models are introduced in a timely ordered fashion (cf. Fig. 3.1, 3.2 and 3.3).

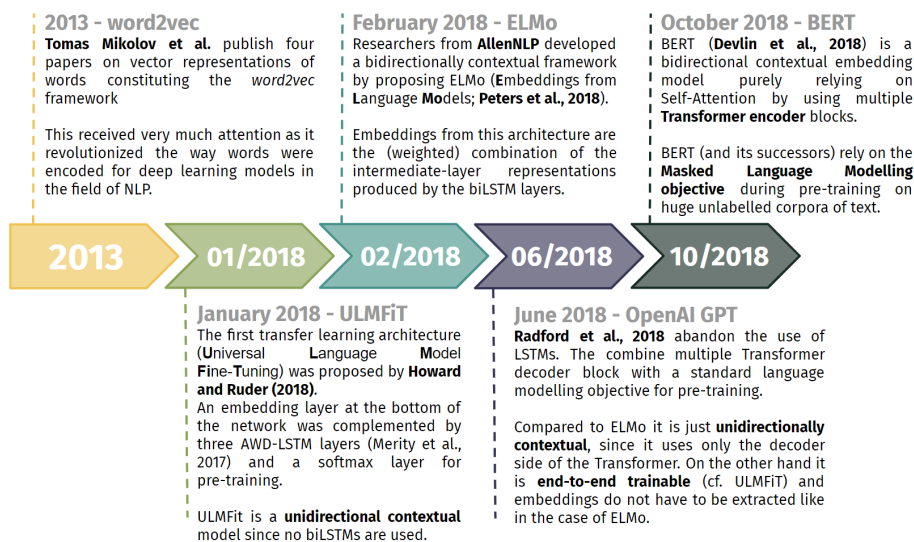


Figure 3.1.: A sketch of the developments in Transfer Learning in NLP upon the introduction of BERT.

Pre-BERT architectures At around the same point in time when ELMo entered the picture, two concurring algorithms were proposed. They were able to alleviate some of ELMo’s shortcomings, but on the other hand also showed some of their own. **Universal Language Model Fine-Tuning** (ULMFiT; 33M parameters; Howard and Ruder, 2018) relies on AWD-LSTMs (Merity et al., 2017), which were the state-of-the-art LSTM architecture at that point in time, and was pre-trained on the language modeling task using Wikitext-103 (Merity et al., 2016b) as pre-training corpus. The model consists of an embedding layer, using complete words as tokens and $|V| = 30k$, and three subsequent LSTM layers complemented by a softmax layer for pre-training. Adaption to the target task is carried out in two different steps. First, the model is further trained⁵ on the language modeling task, but now using data from the target task domain. Finally, a target task specific classifier is added on top of the pre-trained architecture. This classifier’s weights are learned (from scratch) by performing supervised learning on the target task data.

⁵Subtleties of the training procedure are not described in further detail, but it is highly recommendable to have a look at the variety of nifty techniques applied by Howard and Ruder (2018).

The OpenAI GPT (117M parameters; Radford et al., 2018) refers to a pre-trained model consisting of a stack of twelve Transformer decoder blocks which was pre-trained on the BooksCorpus (Zhu et al., 2015) using a standard language modeling objective. Radford et al. (2018) stick to the BPE tokenization algorithm also employed in the original implementation of the Transformer by Vaswani et al. (2017). The rationale for using only the decoder part instead of the whole Transformer architecture is the causality preserving property of the Masked Self-Attention employed in the decoder. This property allows the authors to achieve (unidirectional) contextuality using Self-Attention without compromising the language modeling objective. For fine-tuning, a softmax classification layer is added on top of the pre-trained model and it is trained to jointly minimize the cross-entropy loss of the specific task at hand alongside with the (auxiliary) language modeling loss. A timeline of these architectures (alongside with word2vec, ELMo and BERT) is depicted in Fig. 3.1.

BERT Using the encoder part of the Transformer for learning bidirectionally contextual representations, however, remained an open problem, despite the advances made by Howard and Ruder (2018) and Radford et al. (2018). Unconstrained Self-Attention, inducing bidirectional contextuality, would eventually cause each word in a sequence to have access to its own representations from previous layers. Hence, using a stack of Transformer encoder blocks for pre-training via language modeling – similar to what Radford et al. (2018) do with decoder blocks – is not feasible as it would allow the model to "cheat". Devlin et al. (2019) proposed the pre-training task of *Masked Language Modeling* (MLM), closely related to the cloze task (Taylor, 1953), where a random portion of the input tokens is masked and has to be predicted by the model. Additionally, the *Next Sentence Prediction* (NSP) task is utilized for pre-training, i.e. each input sequence consists of two sentences where the model needs to predict whether the second sentence actually follows the first one or whether it is a random sentence from the training corpus.

BERT is built on the WordPiece tokenization algorithm (Wu et al., 2016) with $|V| = 30k$ and adds three special tokens to the vocabulary: The [CLS]-token precedes every sequence and is used for classification tasks (i.e. NSP during pre-training), while the [SEP]-token is injected between the two input sequences. Solely during pre-training a special [MASK]-token is added, since it is required for MLM. Summing up the token embeddings with (learned) positional embeddings and special (learned) segment embeddings yields the complete input representations. The actual model thereafter consists of 12 Transformer encoder blocks with a hidden dimension⁶ of $H = E = 768$ and $h = 12$ Attention heads for the **BERT_{BASE}** variant (110M parameters) and 24 encoder blocks ($H = E = 1024$, $h = 16$) for the **BERT_{LARGE}** variant (340M parameters).

In order to prepare the training corpus, consisting of the BooksCorpus (Zhu et al., 2015) and the English Wikipedia, the following steps were taken: For MLM, a portion of 15% of the tokens was randomly selected for prediction. Thereof 80% were replaced by [MASK], 10% were replaced by a random token and another 10% were left unchanged⁷. For NSP, sentences were paired such that in 50% of the cases the second sentence was actually the successor of the first one whereas in the other 50% of the cases it was not. The second requirement was for the paired sentences not to exceed a combined sequence length of 512 tokens, which is the maximum number of tokens BERT is able to process.

⁶The hidden layer dimension is chosen to be equal to the dimension of the embedding layer.

⁷The rationale behind this procedure was to mitigate the discrepancy introduced between the pre-training and the fine-tuning phase through the artificial [MASK]-token which only appears during pre-training.

3.3 Fine-tuning approach

Fine-tuning of the architecture is performed by either using the [CLS] for sequence classification tasks or by using each of the token representations from the final encoder block for token-level tasks. Upon its introduction, BERT set new SOTA results on nearly all of the relevant leaderboards (Wang et al., 2018; Rajpurkar et al., 2016, 2018), partly exceeding the performance of the previous leader by a large margin and thus becoming the new "one to be beaten". Regarding the successors of BERT, the remainder of this chapter will differentiate between models directly altering BERT (cf. Fig. 3.2) and models proposing alternate architectures (nevertheless also built upon the Transformer, cf. Fig. 3.3).

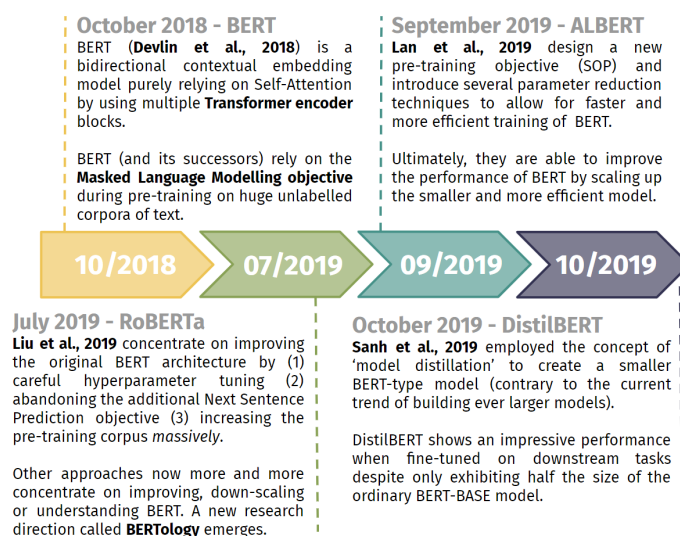


Figure 3.2.: A sketch of the developments in Transfer Learning in NLP based on the architecture of BERT.

BERT-based architectures Much of the following research was about detecting flaws and weaknesses within BERT, which subsequently led to the introduction of BERT-based architectures alleviating these shortcomings. The whole field of interpreting/explaining and improving BERT is often referred to as *BERTology* (Rogers et al., 2020) and is unfortunately by far too large⁸ to be covered more in depth in this thesis. Hence, the remainder of this paragraph introduces three specific architectures which are considered to have had a large impact. Liu et al. (2019) proposed a **Robustly optimized BERT** pre-training approach, short RoBERTa (360M parameters), focussing on turning the adjusting screws on BERT. While architecturally the only change is a larger embedding layer, originating from the use of a 50k BPE vocabulary (instead of 30k Word-Piece in BERT), the pre-training regime is revised substantially. First, the pre-training corpus is replicated ten times and each replicate is (randomly) masked differently.⁹ Second, pre-training is performed (i) on a (more than ten times) larger corpus, (ii) using a notably larger batch size (8k vs. 256) and (iii) waiving the NSP objective. This led to a substantial increase of the SOTA results, supporting the claim of Liu et al. (2019) that "BERT was significantly undertrained".

A Lite BERT (ALBERT; 233M parameters Lan et al., 2019) can be seen as a representative of the model class of (attempted) more efficient BERT-based architectures. The authors substituted

⁸Having a look at <https://github.com/tomohideshibata/BERT-related-papers> might give a good impression of the sheer amount of BERT-related models and literature.

⁹This process is called *Dynamic Masking* and is deemed superior to BERT's static masking once before pre-training, since the model is confronted with the same texts masked differently and is thus able to extract more knowledge.

the NSP objective by their *Sentence Order Prediction* (SOP) objective, where the task is to predict whether the two sentences are fed to the model in the right order or not. More important architectural changes include (i) the disentanglement¹⁰ of H and E and (ii) sharing parameters across layers, increasing parameter efficiency. These changes resulted in an (initially) smaller model, which Lan et al. (2019) scaled up to a size comparable to BERT, achieving superior results on GLUE (Wang et al., 2018), SQuAD (Rajpurkar et al., 2016, 2018) and RACE (Lai et al., 2017). Regarding the claimed higher efficiency it is important to note, that while the memory footprint is reduced by the parameter sharing, the inference speed is not significantly reduced, since the architecture is scaled up again.

Sanh et al. (2019) employ a model distillation technique (Buciluă et al., 2006; Hinton et al., 2015) to compress the pre-trained **BERT_{BASE}** architecture into their DistilBERT model (66M parameters) exhibiting half the size. For pre-training, the same corpus as for BERT but the improvements of RoBERTa (no NSP, dynamic masking, larger batches) are used, resulting in an architecture which can be fine-tuned to achieve nearly 97% of the performance of **BERT_{BASE}**.

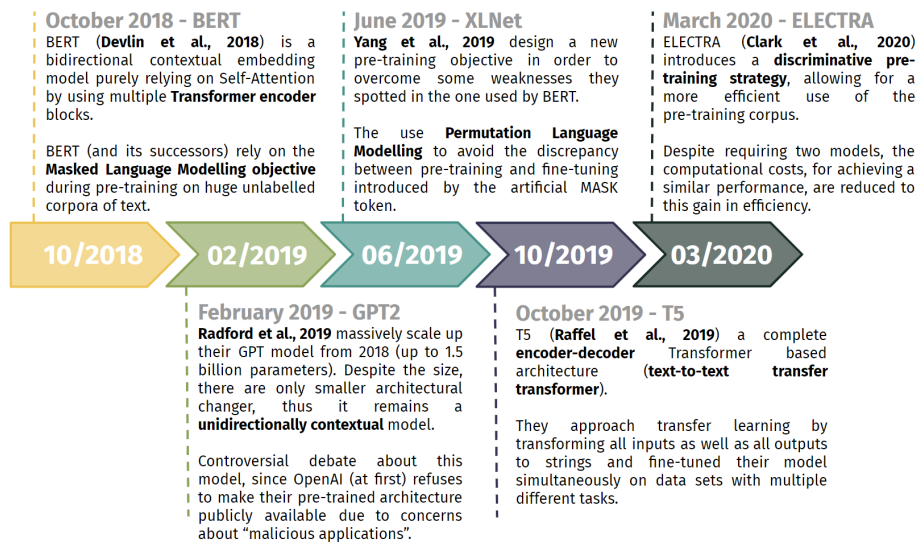


Figure 3.3.: A sketch of further developments in Transfer Learning in NLP since the introduction of BERT.

Post-BERT architectures Like in the previous paragraph, only four of the most notable subsequent alternatives are presented. The next milestone model which was proposed 4 months after BERT is called GPT-2 (Radford et al., 2019), the direct successor of OpenAI GPT. Its most notable features were its sheer size (1.5B parameters, which was the largest parameter count at that point in time) as well as its outstanding ability in text generation. It will be further discussed in Sec. 11.

Another four months later (cf. Fig. 3.3), Yang et al. (2019) proposed the XLNet (340M parameters) architecture, also based on the Self-Attention mechanism. Their new *Permutation Language Modeling* (PLM) objective allowed combining the auto-regressive formulation of a standard left-to-right language model with the deep bidirectionality of BERT’s MLM objective. PLM describes the objective of maximizing the expected log likelihood for all possible permutations of

¹⁰In the Transformer, BERT and RoBERTa those dimensions were tied, i.e. $H = E$, limiting flexibility.

3.3 Fine-tuning approach

the factorization order of a sequence. By including all possible permutations, the model assures that each token has access to tokens from both sides of the context. Despite being computationally more expensive than MLM, PLM addresses two crucial shortcomings of MLM: First, by predicting all of the masked tokens separately, MLM implicitly contains the assumption that the predicted tokens are independent, while PLM does not need this assumption due to its auto-regressive formulation. Second, the discrepancy between pre-training and fine-tuning in BERT, introduced by the artificial [MASK]-token during pre-training, ceases to exist, since XLNet’s auto-regressive structure does not require this kind of input corruption. XLNet exhibits a superior performance compared to BERT on the pertinent benchmark data sets (GLUE, SQuAD, RACE) but also proves to be more computationally demanding.

Instead of just using the Transformer encoder (like BERT & Co.) or the decoder (GPT models), the **Text-to-Text-Transfer-Transformer** T5 (11B parameters; Raffel et al., 2019) is a complete Transformer and thus a sequence-to-sequence model. The underlying idea for this approach is to re-formulate every possible NLP task to a ”text-to-text” problem, i.e. both input and output are sequences of tokens. This allows for *Multi-task learning* (cf. beginning of Chap. 3), since the algorithm is enabled to be trained to solve multiple different tasks simultaneously. Instead, all of the previously introduced models had to be fine-tuned to each of the different tasks separately. Alongside with this novelty, Raffel et al. (2019) introduced the **Colossal Clean Crawled Corpus** (C4, 750GB of plain text) with the intention for (subsets of) it to be used as a standardized pre-training resource. They also performed exhaustive experiments with respect to the architecture, model size, pre-training objective as well as fine-tuning methods and multi-task learning strategies. Again, their final T5 model conquered the top of the most commonly used leaderboards.

The last model presented in this section is ELECTRA (340M parameters; Clark et al., 2020), an architecture relying on a pre-training objective of discriminative (rather than generative¹¹) nature. This algorithm requires two language models for pre-training: A smaller helper model (comparable to BERT) performing masked language modeling on corrupted sequences and the actual ELECTRA model, which is trained on discriminating between ”original” tokens and tokens that were generated by the helper model. During pre-training, the objective is for the complete architecture to minimize a combined loss consisting of the MLM loss from the helper model and the discriminator’s loss. After the pre-training is finished, the helper model is discarded and the discriminator model can be used for fine-tuning. Clark et al. (2020) dubbed this objective *Replaced Token Detection* and claim that it makes more efficient use of the pre-training corpus since it enables the model to learn from *all* tokens in the sequence instead of just from e.g. 15% that were masked (cf. BERT) before pre-training. The final ELECTRA model of a size similar to **BERT_{LARGE}** was pre-trained on the same corpus as XLNet using approximately the same amount of compute as RoBERTa. Performance-wise it is able to outperform (or to reach at least similar performance compared to) all of the presented models, except for T5, on the GLUE and SQuAD benchmarks.

¹¹Nearly all of the pre-training objective described before (Language modeling, Skip-gram, MLM, PLM) can be described as generative, since the model is trained to *generate* plausible substitutions for the next or the masked token.

Part II.

Comparability

4. On the comparability of pre-trained language models

Chapter 4 describes several state-of-the-art transfer learning architectures (at the time of writing this article). It provides a comprehensive overview on the differences between the models with regard to their architectural details as well as the usage of computational resources, size and quality of the corpora used for pre-training.

Furthermore, a comparison of the performance values for the evaluated architectures on benchmarks sets them into context by relating performance to (i) model size, (ii) utilized amount of computational power and (iii) size and accessibility of the pre-training corpora.

Contributing article:

Aßenmacher, M. and Heumann, C. (2020). On the comparability of pre-trained language models. *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS), Zurich, Switzerland (Online), June 23-25, 2020*. <http://ceur-ws.org/Vol-2624/paper2.pdf>.

Copyright information:

This article is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>).

Author contributions:

Matthias Aßenmacher brought up the idea of addressing this issue, at first drafted and finally wrote this paper. Christian Heumann substantially contributed by continuously revising the manuscript and adding ideas.

Supplementary material available at:

- Video of the conference talk: <https://www.youtube.com/watch?v=5GK34g9vyqY>

On the comparability of pre-trained language models

Matthias Aßenmacher
Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
{matthias, chris}@stat.uni-muenchen.de

Christian Heumann
Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany

Abstract

Recent developments in unsupervised representation learning have successfully established the concept of transfer learning in NLP. Instead of simply plugging in static pre-trained representations, end-to-end trainable model architectures are making better use of contextual information through more intelligently designed language modelling objectives. Along with this, larger corpora are used for self-supervised pre-training of models which are afterwards fine-tuned on supervised tasks. Advances in parallel computing made it possible to train these models with growing capacities in the same or even in shorter time than previously established models. These developments agglomerate in new state-of-the-art results being revealed in an increasing frequency. Nevertheless, we show that it is not possible to completely disentangle the contributions of the three driving forces to these improvements.

We provide a concise overview on several large pre-trained language models, which achieved state-of-the-art results on different leaderboards in the last two years, and compare them with respect to their use of new architectures and resources. We clarify where the differences between the models are and attempt to gain some insight into the single contributions of lexical and computational improvements as well as those of architectural changes. We do not intend to quantify these contributions,

but rather see our work as an overview in order to identify potential starting points for benchmark comparisons.

1 Introduction

For solving NLP tasks, most researchers turn to using pre-trained word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017) as a key component of their models. These representations map each word of a sequence to a real valued vector of fixed dimension. Drawbacks of these kinds of externally learned features are that they are (i) fixed, i.e. can not be adapted to a specific domain they are used in, and (ii) context independent, i.e. there's only one embedding for a word by which it is represented in any context.

More recently, transfer learning approaches, as for example convolutional neural networks (CNNs) pre-trained on ImageNet (Krizhevsky et al., 2012) in computer vision, have entered the discussion. Transfer learning in the NLP context means pre-training a network with a self-supervised objective on large amounts of plain text and fine-tuning its weights afterwards on a task specific, labelled data set. For a comprehensive overview on the current state of transfer learning in NLP, we recommend the excellent tutorial and blog post by Ruder et al. (2019)¹.

With ULMFiT (Universal Language Model Fine Tuning), Howard and Ruder (2018) proposed a LSTM-based (Hochreiter and Schmidhuber, 1997) approach for transfer learning in NLP using AWD-LSTMs (Merity et al., 2017). This model can be characterised as unidirectional contextual, while a bidirectionally contextual LSTM-based model was presented in ELMo (Embeddings from Language Models) by Peters et al. (2018).

The bidirectionality in ELMo is achieved by using

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

¹<https://ruder.io/state-of-transfer-learning-in-nlp/>

biLSTMs instead of AWD-LSTMs. On the other hand, ULMFiT uses a more "pure" transfer learning approach compared to ELMo, as the ELMo-embeddings are extracted from the pre-trained model and are *not* fine-tuned in conjunction with the weights of the task-specific architecture.

The OpenAI GPT (Generative Pre-Training, Radford et al., 2018) is a model which resembles the characteristics of ULMFiT in two crucial points. It is a unidirectional language model and it allows stacking task specific layers on top after pre-training, i.e. it is fully end-to-end trainable. The major difference between them is the internal architecture, where GPT uses a Transformer decoder architecture (Vaswani et al., 2017).

Instead of processing one input token at a time, like recurrent architectures (LSTMs, GRUs) do, Transformers process whole sequences all at once. This is possible because they utilize a variant of the *Attention* mechanism (Bahdanau et al., 2014), which allows modelling dependencies without having to feed the data to the model sequentially. At the same time, GPT can be characterised as unidirectional as it just takes into account the left side of the context. Its successor OpenAI GPT2 (Radford et al., 2019) possesses (despite some smaller architectural changes) the same model architecture and thus can also be termed as unidirectional contextual.

BERT (Bidirectional Encoder Representations from Transformers, Devlin et al., 2019), and consequently the other two BERT-based approaches discussed here (Liu et al., 2019; Lan et al., 2019) as well, differ from the GPT models by the fact that they are bidirectional Transformer encoder models. Devlin et al. (2019) proposed *Masked Language Modelling* (MLM) as a special training objective which allows the use of a bidirectional Transformer encoder without compromising the language modelling objective. XLNet (Yang et al., 2019) on the contrary relies on an objective which the authors call *Permutation Language Modelling* (PLM) and is also able to model a bidirectional context despite being an auto-regressive model.

2 Related work

In their stimulating paper, Raffel et al. (2019) take several steps in a similar direction by trying to ensure comparability among different Transformer-based models. They perform various experiments with respect to the transfer learning ability of a Transformer encoder-decoder architecture by vary-

ing the pre-training objective (different variants of denoising vs. language modelling), the pre-training resources (their newly introduced C4 corpus vs. variants thereof) and the parameter size (from 200M up to 11B). Especially, their idea of introducing a new corpus and creating subsets resembling previously used corpora like RealNews (Zellers et al., 2019) or OpenWebText (Gokaslan and Cohen, 2019) is a promising approach in order to ensure comparability.

However, their experiments do not cover an important point we are trying to address with our work: Focussing on only one specific architecture does not yield an answer to the question which components explain the performance differences between models where the overall architecture differs (e.g. Attention-based vs. LSTM-based). Yang et al. (2019) also address comparability to some extent by performing an ablation study to compare their XLNet explicitly to BERT. They train six different XLNet-based models where they modify different parts of their model in order to quantify how these design choices influence performance. At the same time they restrict themselves to an architecture of the same size as BERT-BASE and use the same amount of lexical resources for pre-training. Liu et al. (2019) vary RoBERTa with respect to model size and amount of pre-training resources in order to perform an ablation study also aiming at comparability to BERT. Lan et al. (2019) go one step further with ALBERT by also comparing their model to BERT with regard to run time as well as width and depth of the model.

Despite all these experiments are highly valuable steps into the direction of better comparability, there are still no clear guidelines on which comparisons to perform in order to ensure a maximum degree of comparability with respect to multiple potentially influential factors at the same time.

3 Materials and Methods

First, we present the different corpora which were utilised for pre-training the models and compare them with respect to their size and their accessibility (cf. Tab. 1). Subsequently, we will briefly introduce benchmark data sets which the models are commonly fine-tuned and evaluated on.

While conceptual differences between the evaluated models have been addressed in the introduction, the models will now be described in more detail. This is driven by the intention to emphasise

differences beyond the obvious, conceptual ones.

3.1 Pre-training corpora

English Wikipedia (Devlin et al., 2019) state that they used data from the English Wikipedia and provide a manual for crawling it, but no actual data set. Their version encompassed around 2.5B words. Wikipedia data sets are available in the Tensorflow `Datasets`-module.

CommonCrawl Among other resources, Yang et al. (2019) used data from CommonCrawl. Besides stating that they filtered out short or low-quality content, no further information is given. Since CommonCrawl is a dynamic database, which is updated on a monthly base (and the extracted amount of data always depends on the user) we can not provide a word count for this source in Tab. 1.

ClueWeb (Callan et al., 2009), **Giga5** (Parker et al., 2011) The information about ClueWeb and Giga5 is similarly sparse as for CommonCrawl. ClueWeb was obtained by crawling ~ 2.8M web pages in 2012, Giga5 was crawled between 01/2009 and 12/2010.

1B Word Benchmark² (Chelba et al., 2013) This corpus, actually introduced as a benchmark data set by Chelba et al. (2013), combines multiple data sets from the EMNLP 2011 workshop on Statistical Machine Translation. The authors normalised and tokenized the corpus and performed further pre-processing steps in dropping duplicate sentences as well as discarding words with a count below three. Additionally, they randomised the ordering of the sentences in the corpus. This constitutes a corpus with a vocabulary of 793.471 words and a total word count of 829.250.940 words.

BooksCorpus³ (Zhu et al., 2015) In 2015, Zhu et al. introduced the BooksCorpus, which is heavily used for pre-training language models (cf. Tab. 1). In their work, they used the BooksCorpus in order to train a model for retrieving sentence similarity. Overall, the corpus comprises 984.846.357 words in 74.004.228 sentences obtained from analysing 11.038 books. They report a vocabulary consisting of 1.316.420 unique words, making the corpus lexically more diverse than the 1B Word Benchmark, as it possesses a by 66% larger vocabulary whereas having a word count which is only 19% higher.

²<https://research.google/pubs/pub41880/>

³<https://yknzhu.wixsite.com/mbweb>

Wikitext-103 (Merity et al., 2016a,b) The authors emphasised the necessity for a new large scale language modelling data set by stressing the shortcomings of other corpora. They highlight the occurrence of complete articles, which allows learning long range dependencies, as one of the main benefits of their corpus. This property is, according to the authors, not given in the 1B Word Benchmark as the sentence ordering is randomised there. With a count of 103.227.021 tokens and a vocabulary size of 267.735, it is about one eighth of the 1B Word Benchmark's size concerning token count and about one third concerning the vocabulary size. Note, that there is also the smaller Wikitext-2 corpus (Merity et al., 2016c) available, which is a subset of about 2% of the size of Wikitext-103.

CC-News (Nagel, 2016) This corpus was presented and used by Liu et al. (2019). They used a web crawler proposed by Hamborg et al. (2017) to extract data from the CommonCrawl News data set (Nagel, 2016) and obtained a data set similar to the RealNews data set (Zellers et al., 2019).

Stories⁴ (Trinh and Le, 2018) The authors built a specific subset of the CommonCrawl data based on questions from common sense reasoning tasks. They extracted nearly 1M documents, most of which are taken from longer, coherent stories.

WebText (Radford et al., 2019) This pre-training corpus, obtained by creating "a new web scrape which emphasised document quality" (Radford et al., 2019), is not publicly available.

OpenWebText (Gokaslan and Cohen, 2019) As a reaction to Radford et al. (2019) *not* releasing their pre-training corpus, Gokaslan and Cohen (2019) started an initiative to emulate an open-source version of the WebText corpus.

It becomes obvious that there is a lot of heterogeneity with respect to the observed combinations of availability, quality and corpus size. Thus, we can state that there is some lack of transparency when it comes to the lexical resources used for pre-training. Especially, the missing standardised availability of the BooksCorpus is problematic as this corpus is heavily used for pre-training.

⁴https://console.cloud.google.com/storage/browser/commonsense-reasoning/reproduce/stories_corpus

Corpora	Word-count [♡]	Accessibility	Used by
English Wikipedia	~ 2.500M	Fully available	BERT; XLNet; RoBERTa; ALBERT
CommonCrawl	Unclear	Fully available	XLNet
ClueWeb 2012-B, Giga5	Unclear	Fully available (\$\$)	XLNet
1B Word Benchmark	~ 830M	Fully available	ELMo
BooksCorpus	~ 985M	Not available	GPT; BERT; XLNet; RoBERTa; ALBERT
Wikitext-103	~ 103M	Fully available	ULMFit
CC-News	Unclear	Crawling Manual	RoBERTa
Stories	~ 7.000M [◇]	Fully available	RoBERTa
WebText	Unclear	Not available	GPT2
OpenWebText	Unclear	Fully available	RoBERTa

Table 1: Pre-training resources (sorted by date). *Crawling Manual* means the authors did not provide data, but at least a manual for crawling it. Dollar signs signify the necessity of a payment in order to get access. RealNews (Zellers et al., 2019) and C4 (Raffel et al., 2019) are not included as they were not used by the evaluated models.

♡ We report the word-count as given in the respective articles proposing the corpora. Note that the number of *tokens* reported in other articles depends on the tokenization scheme used by a specific model.

◇ Stated by one of the authors on twitter: https://twitter.com/thtrieu_/status/1096672446864748545

3.2 Benchmark data sets for fine-tuning

GLUE⁵ (Wang et al., 2018) The *General Language Understanding Evaluation* (GLUE) benchmark is a freely available collection of nine data sets on which models can be evaluated. It provides a fixed train-dev-test split with held out labels for the test set, as well as a leaderboard which displays the top submissions and the current state-of-the-art (SOTA). The relevant metric for the SOTA is an aggregate measure of the nine single task metrics. The benchmark includes two binary classification tasks with single-sentence inputs (CoLa [Warstadt et al., 2018] and SST-2 [Socher et al., 2013]) and five binary classification tasks with inputs that consist of sentence-pairs (MRPC [Dolan and Brockett, 2005], QQP [Shankar et al., 2017], QNLI, RTE and WNLI [all Wang et al., 2018]). The remaining two tasks also take sentence-pairs as input but have a multi-class classification objective with either three (MNLI [Williams et al., 2017]) or five classes (STS-B [Cer et al., 2017]).

SuperGLUE⁶ (Wang et al., 2019) As a reaction to human baselines being surpassed by the top ranked models, Wang et al. (2019) proposed a set of benchmark data sets similar to, but, according to the authors, more difficult than GLUE. It did not make sense to include it as a part of our model comparison, as (at the time of writing) only two of the

discussed models were evaluated on SuperGLUE.

SQuAD⁷ (Rajpurkar et al., 2016, 2018) The *Stanford Question Answering Dataset* (SQuAD) 1.1 consists of 100.000+ questions explicitly designed to be answerable by reading segments of Wikipedia articles. The task is to correctly locate the segment in the text which contains the answer. A shortcoming is the omission of situations where the question is not answerable by reading the provided article. Rajpurkar et al. (2018) address this problem in SQuAD 2.0 by adding 50.000 hand-crafted unanswerable questions to SQuAD 1.1. The authors provide a train and development set as well as an official leaderboard. The test set is completely held out, participants are required to upload their models to CodaLab. The SQuAD 1.1 data is, in an augmented form (QNLI), also part of GLUE.

RACE⁸ (Lai et al., 2017) The *Large-scale Reading Comprehension Dataset From Examinations* (RACE) contains English exam questions for Chinese students (middle/high school). In most of the articles using RACE for evaluation, it is described to be especially challenging due to (i) the length of the passages, (ii) the inclusion of reasoning questions and (iii) the intentionally tricky design of the questions in order to test a human’s ability in reading comprehension. The data set can be subdivided

⁵<https://gluebenchmark.com/>

⁶<https://super.gluebenchmark.com/>

⁷<https://rajpurkar.github.io/SQuAD-explorer/>

⁸http://www.qizhexie.com/data/RACE_leaderboard.html

in RACE-M (middle school examination) and RACE-H (high school examination) and comprises a total of 97.687 questions on 27.933 passages of text.

3.3 Evaluated Models

ULMFiT (Howard and Ruder, 2018) The AWD-LSTMs in this architecture make use of DropConnect (Wan et al., 2013) for better regularisation and apply averaged stochastic gradient descent (ASGD) for optimization (Polyak and Juditsky, 1992). The model consists of an embedding layer followed by three LSTM layers with a softmax classifier on top for pre-training. It is complemented by a task specific final layer during fine-tuning. The vocabulary size is limited to 30k words as in Johnson and Zhang (2017).

ULMFiT was not evaluated on GLUE, but on several other data sets (IMDb [Maas et al., 2011], TREC-6 [Voorhees and Tice, 1999], Yelp-bi, Yelp-full, AG’s news, DBpedia [all Zhang et al., 2015]).

ELMo (Peters et al., 2018) Consisting of multiple BiLSTM layers, one can extract multiple intermediate-layer representations from ELMo. These representations are used for computing a (task-specific) weighted combination, which is concatenated with external, static word embeddings. During the training of the downstream model, ELMo embeddings are not updated, only the weights for combining them are. For the GLUE benchmark there are multiple ELMo-based architectures available on the leaderboard. In Tab. 3, we report the best-performing model, an ELMo-based BiLSTM-model with Attention (Wang et al., 2018).

OpenAI GPT (Radford et al., 2018) The OpenAI GPT is a pure attention-based architecture that does not make use of any recurrent layers. Pre-training is performed by combining Byte-Pair encoded (Sennrich et al., 2015) token embeddings with learned position embeddings, feeding them into a multi-layer transformer decoder architecture with a standard language modelling objective. Fine-tuning was, amongst others, performed on the nine tasks that together form the GLUE benchmark.

BERT (Devlin et al., 2019) BERT can be seen as a reference point for everything that came thereafter. Similar to GPT it uses Byte-Pair Encoding (BPE) with a vocabulary size of 30k. By introducing the MLM objective, the authors were able to combine deep bidirectionality with Self-

Attention for the first time. Additionally, BERT also utilizes the next-sentence prediction (NSP) objective, the usefulness of which has been debated in other research papers (Liu et al., 2019). The BERT-BASE model consists of 12 bidirectional transformer-encoder blocks (24 for BERT-LARGE) with 12 (16 respectively) attention heads per block and an embedding size of 768 (1024 respectively).

OpenAI GPT2 (Radford et al., 2019) Compared to its predecessor GPT, it contains some smaller changes concerning the placement of layer normalisation and residual connections. Overall, there are four different versions of GPT2 with the smallest one being equal to GPT, the medium one being of similar size as BERT-LARGE and the xlarge one being released as the actual GPT2 model with 1.5B parameters.

XLNet (Yang et al., 2019) In order to overcome (what they call) the *pretraining-finetune discrepancy*, which is a consequence of BERT’s MLM objective, and to simultaneously include bidirectional contexts, Yang et al. (2019) propose the PLM objective. They use two-stream self-attention for preserving the position information of the token to be predicted, which would otherwise be lost due to the permutation. While the *content stream attention* resembles the standard Self-Attention in a transformer-decoder, the *query stream attention* doesn’t allow the token to see itself but just the preceding tokens of the permuted sequence.

RoBERTa (Liu et al., 2019) With RoBERTa (Robustly optimized BERT approach), Liu et al. (2019) introduce a replicate of BERT with tuned hyperparameters and a larger corpus used for pre-training. The masking strategy is changed from static (once during pre-processing) to dynamic (every sequence just before feeding it to the model), the additional NSP objective is removed, the BPE vocabulary is increased to 50k and training is performed on larger batches than BERT. These adjustments improve performance of the model and make it competitive to the performance of XLNet.

ALBERT (Lan et al., 2019) By identifying that the increase of the model size is a problem, ALBERT (A Lite BERT) goes into another direction compared to most of post-BERT architectures. Parameter-reduction techniques are applied in order to train a faster model with lower memory demands that, at the same time, yields a comparable

Model	Compute			Resources	
	Hardware	Training time	pfs-days [♡]	#parameters	lexical
ULMFiT	NA	NA	NA	33M	0.18GB
GPT	8 GPUs (P600)	~ 30 days	0.96	117M	< 13GB
BERT-BASE	4 Cloud TPUs	~ 4 days	0.96 [2.24] [◇]	110M	13GB
BERT-LARGE	16 Cloud TPUs	~ 4 days	3.84 [8.96] [◇]	340M	13GB
GPT2-MEDIUM	NA	NA	NA	345M	40GB
GPT2-XLARGE	8 v3 Cloud TPUs	~ 7 days	7.84	1.500M	40GB
XLNet-LARGE	128 v3 Cloud TPUs	~ 2.5 days	44.8	340M	126GB
RoBERTa	DGX-1 GPUs (8xV100) [♣]	NA [♣]	NA	360M	160GB
	1024 32GB V100 GPUs [♣]	~ 1 day [♣]	4.78	360M	16GB
ALBERT	64 – 1024 v3 Cloud TPUs	NA	NA	233M	16GB

Table 2: Usage of compute and pre-training resources alongside with model size for the evaluated model architectures. With *lexical resources* we refer to the size of the pre-training corpus. ELMo not included as it is not end-to-end trainable (Size depends on the used model after obtaining the embeddings). The size of ULMFiT is assumed to be the larger value from Merity et al. (2017), since Howard and Ruder (2018) use AWD-LSTMs with a vocabulary size of 30k tokens (Johnson and Zhang, 2016, 2017). Values for GPT2-XLARGE are taken from Strubell et al. (2019).

[♡] *Petaflop-days*: Estimation according to the formula proposed on <https://openai.com/blog/ai-and-compute/>:

$\text{pfs-days} = \text{number of units} \times \text{PFLOPS/unit} \times \text{days trained} \times \text{utilization}$, with an assumed utilization of $\frac{1}{3}$. PFLOPS/unit for TPUs from <https://cloud.google.com/tpu/>.

[◇] Unclear, whether v2 or v3 TPUs were used. Thus, we provide calculations for both: v2[v3]

[♣] Full RoBERTa model (Liu et al., 2019) [♣] RoBERTa variant utilizing less pre-training resources

performance to SOTA models. We will always refer to the best performing ALBERT-XXLARGE, despite also the smaller ALBERT models yield results comparable to BERT.

4 Model comparison

Tab. 2 gives an overview on the amount of computational power needed to pre-train a given architecture on given pre-training (lexical) resources. In Tab. 3 we will directly try to relate model architecture and size as well as usage of lexical resources to model performance.

One thing we can learn from Tab. 2 is the lack of details when it comes to reporting the computational resources used for pre-training. While Howard and Ruder (2018) do not provide any information on the computational power utilised for pre-training, the other articles report it to different degrees. Unfortunately, there are no clear guidelines on how to appraise this when it comes to evaluating and comparing models. This may be attributed to the rapidly growing availability of hardware, but in our opinion it should nevertheless be accounted for, since it might pose environmental issues (Strubell

et al., 2019) and also limits portability to smaller devices.

Further, it is important to consider the differences displayed in the Tab. 2 and Tab. 3 when comparing the model performances. Considering two models of approximately the same size (BERT-BASE vs. GPT), the superior performance of BERT-BASE seems to originate purely from its more elaborated architecture because of the similar size. But one should also be aware of the larger lexical resources (BERT-BASE uses at least twice as much data for pre-training) and the unknown differences in usage of computational power. We approximated the latter as the *pfs-days* (cf. Tab. 2), resulting in an estimation for BERT-BASE being not less than the one for GPT.

Another aspect which should not be ignored when evaluating performance is ensembling. As can be seen in the first column of Tab. 3, the three model ensembles outperform both of the BERT models by a large margin. Only parts of these differences may be attributed to the model architecture or the hyperparameter settings, as the ensembling as well as the larger pre-training resources might give an

4. On the comparability of pre-trained language models

Model	GLUE		SQuAD		RACE	Resources	
	leaderboard	dev [♡]	v1.1 (dev)	v2.0 (dev)	test	#parameters	lexical
BERT-BASE	78.3	–	88.5	76.3 ♣	65.0 ♣	110M	13GB
ELMo-based	- 8.3	–	- 2.9	–	–	–	–
GPT	- 5.5	–	–	–	- 6.0	1.1x	< 0.5x
BERT-LARGE	+ 2.2	84.05	+ 2.4	+ 5.6	+ 7.0 ♣	3.1x	1.0x
XLNet-BASE	–	–	–	+ 5.03	+ 1.05	~ 1.0x	1.0x
XLNet-LARGE	+ 10.1 ◇	+ 3.39	+ 6.0	+ 12.5	+ 16.75	3.1x	9.7x
RoBERTa	+ 10.2 ◇	+ 5.19	+ 6.1	+ 13.1	+ 18.2	3.3x	12.3x
RoBERTa-BASE	–	+ 2.30	–	–	–	1.0x	12.3x
RoBERTa [‡]	–	+ 3.79	+ 5.1	+ 11.0	–	3.3x	1.2x [†]
ALBERT	+ 11.1 ◇	+ 5.91	+ 5.6	+ 13.9	+ 21.5	2.1x	1.2x [†]

Table 3: Performance values as well as model size and resource usage (Reference in *italics*, highest improvements in **bold**). Performance differences are given in percentage points (%pts), differences in size/resources as factors. ULMFiT and GPT2 are omitted as there are no performance values on these data sets publicly available. No model size for ELMo provided, since the performance values are from different models (cf. Sec. 3.3).

Displayed performance measures are Matthews Correlation (GLUE), F1 score (SQuAD) and Accuracy (RACE).

♡ Own calculations based on Lan et al. (2019), Tab. 13; WNLI is excluded ◇ Ensemble performance

♣ Values taken from Yang et al. (2019), Tab. 6 ♠ Values taken from Zhang et al. (2019), Tab. 2

† Liu et al. (2019) and Lan et al. (2019) specify the BooksCorpus + English Wikipedia as 16GB

‡ This variant of RoBERTa uses only BooksCorpus + English Wikipedia for pre-training

advantage to these models. As there are no performance values of *single* models available for XLNet, RoBERTa and ALBERT on the official GLUE leaderboard, we also compare the single model performances from Lan et al. (2019) obtained on the dev sets. From this comparison we get an impression of how high the contribution of ensembling might be: The difference between BERT-LARGE and the XLNet ensemble in the official score (7.9 %pts) is more than twice as high as the difference in dev score (3.4 %pts).

In order to address the differences in size of the pre-training resources, Yang et al. (2019) make the extremely insightful effort to compare a XLNet-BASE variant to BERT-BASE using the same pre-training resources. While the F1 score on SQuAD v2.0 is still remarkably higher than for BERT-BASE (comparable to BERT-LARGE) it does not show a large improvement on RACE (which might have been expected due to the large improvement of XLNet-LARGE over both BERT models).

The comparability of RoBERTa from the GLUE leaderboard (ensemble + larger pre-training resources) to BERT-LARGE is limited, but the authors perform several experiments in order to show the usefulness of their optimisations. Pre-training

a single model on comparable lexical resources (13GB for BERT vs. 16GB for RoBERTa), the RoBERTa model shows a smaller (compared to the RoBERTa ensemble), but still remarkable, improvement over BERT-LARGE. In another ablation study, Liu et al. (2019) train a RoBERTa-BASE variant on larger pre-training resources. Even though comprising only about one third of the size of BERT-LARGE, the larger pre-training corpus in conjunction with the optimised training leads to a slightly better performance on the GLUE dev set. We are not able to compare RoBERTa-BASE to BERT-BASE, as neither the "official" leaderboard score for RoBERTa-BASE nor the "inofficial" dev set score for BERT-BASE are available.

In order to set the results of ULMFiT into context, we present the results published by Yang et al. (2019) alongside with information on size and pre-training resources in Tab. 4. Despite being much larger and pre-training on some orders of magnitude larger corpora, BERT-LARGE and XLNet-LARGE do not exhibit that large improvements over the performance of ULMFiT. This might partly originate from the relative simplicity of the tasks, but partly also from the already achieved high performances.

Model	Sentiment			Topic		Resources	
	IMDb	Yelp-bi	Yelp-full	AG’s news	DBpedia	size	lexical
ULMFiT	95.40	97.84	70.02	94.99	99.20	33M	0.18GB
BERT-LARGE	+ 0.09	+ 0.27	+ 0.66	–	+ 0.16	10.3x	72.2x
XLNet-LARGE	+ 0.81	+ 0.61	+ 2.28	+ 0.52	+ 0.18	10.3x	222.2x

Table 4: Performance comparison (+ model size and resource usage) on the benchmark data sets used by Howard and Ruder (2018). Specification of the differences and highlighting as in Tab. 3. We report accuracies, as opposed to Howard and Ruder (2018); Yang et al. (2019), in order to facilitate a similar interpretation compared to Tab. 3.

5 Discussion

This chapter reflects the main takeaways from the above comparisons and raises some issues for research practices. We do not claim to have a solution to these potentially problematic aspects, but rather think that these points are highly debatable.

Why no benchmark corpus for pre-training?

It is good practice to use benchmark data sets for comparing the performance of pre-trained language models on different types of *Natural language understanding* (NLU) tasks. Many recently published articles (Liu et al., 2019; Yang et al., 2019; Lan et al., 2019) perform (partly extensive) ablation studies controlling for pre-training resources in order to make (versions of) their models comparable to BERT, which is really important as it helps to get an intuition for the impact of pre-training resources. Nevertheless, it is unfortunately not perfect due to two critical issues: (i) BERT and all of its successors make use of the BooksCorpus (Zhu et al., 2015) which is not publicly available and (ii) this only leads to model comparisons in a low pre-training resource environment (compared to more recent models) and yields no insight on the behaviour of the reference model (e.g. BERT) in a medium or high resource context. So we view statements of the type *"Model architecture A is superior to model architecture B on performing task X."* somewhat critical and propose to phrase it more like the following statement: *"Model architecture A is superior to model architecture B on performing task X, when pre-trained on a small/medium/large corpus of low/high quality data from domain Y for pre-training time Z."*

Why no standardised description of (computational) resources? When writing this article, it turned out difficult to get one unified measure for

the amount of the computational power used for pre-training. In our opinion, this is not a carelessness of the authors but rather the lack of a clear reporting standard. We found ourselves confronted with the following situations:

- No information at all (Radford et al., 2019)
- Hardware (Liu et al., 2019; Lan et al., 2019)
- Hardware and training time (Devlin et al., 2019; Yang et al., 2019)
- Standardised measure (Radford, 2018)

While *a)* is clearly unsatisfactory and should be avoided, *b)* and *c)* provide most of the necessary information but miss out on going the last final step to *d)*, where the reporting reaches universal comparability across different articles. The measure we computed (cf. Tab. 2) is of course not as exact as a computation based on the counts of operations in a network, but requires no deep insight into the model architecture and is thus applicable to a wide range of architectures without much effort.

Shouldn't performance be evaluated in relation to size and resource usage?

As larger models have a higher capacity for learning representations and using larger pre-training resources should improve their quality, varying these two components simultaneously with the model architecture might lead to interference between the individual effects on model performance. This aspect has a slight overlap with the question raised above, but while the above is more or less about introducing some reference, this is about carefully varying and evaluating the effects of different model parts.

6 Conclusion

As can be seen from the above analysis, there is a lack of a concise guideline for *fair* comparisons of

large pre-trained language models. It is not sufficient to just rank models by their performance on the common benchmark data sets as this does not take into account all the other factors mentioned in this analysis. Further aspects worth reporting are the use of resources (time and compute) spent on model development (including all experimental runs and trials) and hyperparameter tuning during pre-training. In our opinion, this is important with respect to two facets: On the one hand side it is important to take into account environmental considerations when training deep learning models (Strubell et al., 2019), on the other hand side it is also a signal to the reader/user how difficult it is to train (and to fine-tune) the model. This might have implications for the usage of a model as transfer learning model for diverse downstream tasks. Models that have already been tuned to a high degree during pre-training to reach a certain level of performance, may have, in the long run, less potential for further improvements compared to models which do so without much hyperparameter tuning. To conclude, we unfortunately cannot say with determination which one of the influential factors (architecture or amount of pre-training resources) is more important, but we think that a substantial amount of the recent improvements can be attributed to larger pre-training resources. A detailed disentanglement of the influence of the different components stays an open research question which might be answerable by carefully designed benchmark studies.

Acknowledgments

We would like to thank the three anonymous reviewers for their insightful comments and their feedback on our work.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.
- Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.
- Aaron Gokaslan and Vanya Cohen. 2019. [Openwebtext corpus](#).
- Felix Hamborg, Norman Meuschke, Corinna Rittinger, and Bela Gipp. 2017. News-please: a generic news crawler and extractor. In *15th International Symposium of Information Science (ISI 2017)*, pages 218–223.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Rie Johnson and Tong Zhang. 2016. Convolutional neural networks for text categorization: Shallow word-level vs. deep character-level. *arXiv preprint arXiv:1609.00718*.
- Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.

2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- 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:1907.11692*.
- 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. Association for Computational Linguistics.
- Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016a. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016b. [Wikitext-103](#). Accessed: 2020-02-10.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016c. [Wikitext-2](#). Accessed: 2020-02-10.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Sebastian Nagel. 2016. [Cc-news](#).
- Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07*, 12.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.
- Alec Radford. 2018. [Improving language understanding with unsupervised learning](#). Accessed: 2020-02-10.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. [Transfer learning in natural language processing](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota. Association for Computational Linguistics.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.
- Iyer Shankar, Dandekar Nikhil, and Csernai Kornél. 2017. [First quora dataset release: Question pairs](#). Accessed: 2020-02-10.
- 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*, pages 1631–1642.
- Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.
- Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

- Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82. Citeseer.
- Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropout. In *International conference on machine learning*, pages 1058–1066.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Super-glue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *arXiv preprint arXiv:1905.12616*.
- Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dual co-matching network for multi-choice reading comprehension. *arXiv preprint arXiv:1901.09381*.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.
- Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Part III.

Task-specific evaluation

5. Evaluating pre-trained language models on applications in Social Sciences

Chapter 5 deals with a complicated classification task – namely *multi-label classification* – where one observation can potentially be assigned to multiple classes simultaneously. This use case of classifying answers to open-ended questions on electoral participation, voting behavior and public opinion into up to 72 categories while only having less than 10.000 examples proves to be very challenging for the evaluated transfer learning models. A further main outcome of this contribution is a transparent and fully reproducible preparation of the used data sets, which was not available in this format prior to this research project.

Contributing article:

Meidinger, M. and 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. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021), Vienna, Austria (Online), February 4-6, 2021, Vol. 2: 866–873.* <https://doi.org/10.5220/0010255108660873>.

Copyright information:

Science and Technology Publications Lda (SCITEPRESS), 2021.

Author contributions:

Both authors jointly set up the manuscript and selected the models as well as suitable performance measures for the evaluation on this task. While Matthias Aßenmacher mainly contributed with his expertise regarding deep learning and pre-trained language models, Maximilian Meidinger further added the required domain-specific knowledge. Programming was mainly done by Maximilian Meidinger and jointly revised with Matthias Aßenmacher. Both of the authors put in substantial effort to create a transparent and fully reproducible preparation of the data set. Besides the two authors, also Christian Heumann proofread and contributed to revisions of the paper.

Supplementary material available at:

- Code and data set: https://github.com/mxli417/co_benchmark
- Poster: <https://www.misoda.statistik.uni-muenchen.de/forschung/icaart.pdf>

A New Benchmark for NLP in Social Sciences: Evaluating the Usefulness of Pre-trained Language Models for Classifying Open-ended Survey Responses

Maximilian Meidinger and Matthias Aßenmacher^a

Department of Statistics, Ludwig-Maximilians-Universität, Munich, Germany
mx.meidinger@gmail.com, matthias@stat.uni-muenchen.de

Keywords: Benchmark, Multi-label Classification, Open-ended Responses, Transfer Learning, Pre-trained Language Models.


Abstract: In order to evaluate transfer learning models for Natural Language Processing on a common ground, numerous general domain (sets of) benchmark data sets have been established throughout the last couple of years. Primarily, the proposed tasks are classification (binary, multi-class), regression or language generation. However, no benchmark data set for (*extreme*) *multi-label* classification relying on full-text inputs has been proposed in the area of social science survey research to this date. This constitutes an important gap, as a common data set for algorithm development in this field could lead to more reproducible, sustainable research. Thus, we provide a transparent and fully reproducible preparation of the 2008 American National Election Study (ANES) data set, which can be used for benchmark comparisons of different NLP models on the task of multi-label classification. In contrast to other data sets, our data set comprises full-text inputs instead of bag-of-words representations or similar. Furthermore, we provide baseline performances of simple logistic regression models as well as performance values for recently established transfer learning architectures, namely BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019).

1 INTRODUCTION

The quasi-standard method in machine learning to determine the performance of a newly proposed method is to evaluate it on benchmark data sets. The same applies for the evaluation of pre-trained language models frequently utilized for transfer learning in Natural Language Processing (NLP). Collections of benchmark data sets for different natural language understanding (NLU) tasks (Rajpurkar et al., 2016; Lai et al., 2017; Wang et al., 2018) have gained massive popularity among researchers in this field. These benchmark collections stand out mainly due to two aspects: They are extremely well documented with respect to their creation and they are fixed with respect to the train-test split and the applied evaluation metrics. Furthermore they provide public leaderboards¹, where the results of submitted models are displayed in a unified fashion. For the majority of the proposed benchmark data sets the task is either a binary or a multi-class classification task (cf. data sets from

Wang et al. (2018)). In the context of social science survey research, however, to our knowledge no existing (*extreme*) multi-label data sets (Lewis et al., 2004; Mencia and Fürnkranz, 2008) have been used for performance evaluation by any of the current state-of-the-art (SOTA) transfer learning models. These, and other (tabular) multi-label data sets can e.g. be found in repositories like MULAN.

In the social sciences, especially in survey research, definitive standards for raw data formatting of open-ended survey questions have not yet been established to our knowledge. This is not to say that there exist no current standards for handling and organizing survey research data in general (Inter-University Consortium For Political And Social Research (ICSPR), 2012; CESSDA Training Team, 2020) or the metadata describing the primary data (Vardigan et al., 2008; Hoyle et al., 2016). Yet, for open-ended survey questions and their *coding*², these standards have not been well established, apart from descriptions of best practices by some authors (Züll, 2016; Lupia,

^a <https://orcid.org/0000-0003-2154-5774>

¹e.g. <https://gluebenchmark.com/leaderboard>

²The process of manually assigning survey responses to pre-defined sets of labels (*codes*) is known as *coding*.

2018b,a).

Our data set preparation represents a novelty since it combines an interesting use-case (multi-label classification) for NLP models in Social Sciences with a fully reproducible pre-processing resulting in *full-text strings* as inputs. Note that this combination is not yet included in the benchmark collections mentioned above³. Thus, in the spirit of the growing overall need for standardized data sets and for reproducibility, we provide a description (cf. Sec. 2), an overview on previous use of this data set (cf. Sec. 3) and a thoroughly described pre-processing (cf. Sec. 4.1) of the ANES 2008 data, which enables its usage for benchmark comparisons for multi-label classification. Baseline performance values for a simple machine learning model as well as for more recently proposed transfer learning architectures are provided in Sec. 5.

2 THE "AMERICAN NATIONAL ELECTION STUDIES" SURVEY

The American Election Studies (ANES) provide high-quality data for political and social science research by conducting surveys on political participation, public opinion and voting behavior since 1948. To fulfill this commitment, ANES conducts a series of biennial election studies which cover these topics, sometimes extended by surveys on special-interest topics and expanded methodological instrumentation.

The 28th ANES time series study in 2008 (The American National Election Studies, 2015) has been supplemented by a coding project for open-ended responses (Krosnick et al., 2012) to various pre- and post-election questions. The topics ranged from reasons to vote for a presidential candidate, perceived reasons why a candidate won or lost the 2008 election, across the most important problems for the country and the electorate, over to (dis)likes of the competing political figures and parties among the respondents.

Like in all previous ANES studies conducted in years of presidential elections, respondents were interviewed in pre-election interviews and then re-interviewed in the two months following the election (post-election interviews), hence there was a varying number of respondents.

³Despite these benchmark collections do include data sets with text input, all inputs are provided as bag-of-words representations or similar, but **not** as full-text verbatims.

3 RELATED WORK

Card and Smith (2015) already investigated machine learning methods for automated coding of the ANES 2008 data. Namely, they evaluated (regularized) logistic regression models as well as recurrent neural network architectures, including long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). As a result, they find that recurrent neural network based methods are not generally able to outperform the more "traditional" natural language processing methods, like logistic regression models combined with uni-/bigrams or additional features. An interesting conclusion they draw from their analysis is that this might be due to the limited amount of training data available for this multi-label classification task at hand. Since this is a problem statement explicitly addressed by recent transfer learning approaches, we are curious to find out whether pre-trained architectures like BERT & Co. are able to perform better on this task. Roberts et al. (2014) work on the ANES 2008 data by applying a structural topic model as a fully unsupervised approach for automated coding, which is a highly interesting strategy for previously unlabeled data sets. But since our goal is to evaluate the ability of transfer learning models (which rely on labeled data) for multi-label classification, we do not make use of this methodology.

4 MATERIALS AND METHODS

4.1 Preparation of the ANES Data

The data from the *Open Ended Coding Project*⁴ consists of a main file in *.xls - format which combines all verbatims⁵ from the targeted respondents collected on the individual questions in separate spreadsheets. The codes assigned to these verbatims are stored separately in so-called *codes-files*.

Analogously to the work of Card and Smith (2015), we only use the answers to the open-ended questions *unrelated* to occupation/industry of the respondents. The topics of the questions defining the different data sets are displayed in Tab. 1. As some of the questions share the same code sets, they can be grouped into ten individual data sets comprising all of the questions on the topics mentioned in Sec. 2. With this, we follow the data preparation strategy of Card

⁴Publicly available under: ANES time series study and the Open Ended Coding Project

⁵Answers to the open-ended survey questions are referred to as *verbatims*

and Smith (2015), to keep our later results roughly comparable to their model benchmarks. Until now, there seems to be no broadly accepted data format or structure in the social sciences regarding the storage and publication of codes assigned to individual responses to open-ended questions in surveys. Data sets seem to be structured matrix-like ad-hoc to fit an individual survey's needs.

Besides the obvious structural requirements, namely that the codes assigned to each response have to be identifiable using a particular variable (here this is provided via an "ID", alternatively designated as "caseID") and that there is a limited amount of variables which can be used for storing the code values for a single response, the internals of such data sets seem to be highly idiosyncratic. Another aspect which partially varies between different surveys are the codes being used for indicating that a value is missing. This in turn leads to the problem that these data sets as such are hardly usable for standard machine learning purposes without extensive preprocessing which has to reflect the individual survey's logic.

In the particular case of the ANES 2008, one has to turn to the so-called "coding report" accompanying each response-codes data set to identify the columns which contain the codes for a specific question and to understand their meaning. The pre-defined codes for each question have been manually assigned to the individual responses by professional human coders. The coding procedure has been developed after a thorough review of the ANES open-ended coding methods and a subsequent conference in December 2008⁶ which suggested best practices.

As the sets of predefined codes belonging to individual questions cannot be used for machine learning purposes as such, we have to transform them into a useful format. In order to generate usable data sets from the files distributed by the *Open Ended Coding Project*, we exploit the notion of representing the codes, which have been assigned to each textual observation, by a binary vector.

As described previously by various authors (Tsoumakas and Katakis, 2007; Gibaja and Ventura, 2015; Herrera et al., 2016), multi-label problems can be formalized by proposing an output space $L = L_1, L_2, \dots, L_q$ of q labels ($q > 1$), which allows us to describe each observation in the data as (\mathbf{x}, Y) where $\mathbf{x} = (x_1, \dots, x_d) \in X$ is a d -dimensional instance which has a set of labels associated $Y \subseteq L$. In this paper, we understand the codes assigned to each response in the data as the labels encountered in a multi-label learning problem, just as Card and Smith (2015) did pre-

viously. In order to transform the numeric codes assigned to the responses into *multi-hot encodings*, we exploit the cardinality of the code set associated with each question. This helps us to represent the labels associated to each observation by a q -dimensional binary vector $\mathbf{y} = (y_1, \dots, y_q) = \{0, 1\}^q$ where each element is 1 if the respective label was assigned to the response and 0 otherwise.

To map the numeric codes to binary label vector elements one-to-one, we sourced the total size of each code set from the *codes*-documents enclosed with each data set. Using this information, we defined the length of the binary mapping vectors to be identical to the cardinality of the code sets. To generate multi-hot encoded label vectors for each response contained in the data sets, we designed a mapping dictionary for each code set defining which code from the current set belongs to which element in the binary vector generated for a particular response. To finally obtain the binary label vectors from the set of numeric codes associated to each observation, we transformed all data sets using a custom function which can be fed a mapping dictionary and the raw data row-by-row. The function then returns the binary label vectors of length q for each observation, where each vector element is 1 if the code mapped to this element was assigned to the response and 0 otherwise. For the latter application of machine learning methods we split the data into train and test set (90/10) using an iterative stratification method for balancing the label distributions (Sechidis et al., 2011; Szymański and Kajdanowicz, 2017a) implemented in the novel *scikit-multilearn* library for Python (Szymański and Kajdanowicz, 2017b). This represents an innovation, as such stratification has not been previously used by Card and Smith (2015). The resulting data splits are publicly available.⁷

4.2 Model Architectures

Simple Baseline. As a simple baseline we use a logistic regression classifier (without regularization) for *one vs. rest classification* per label and thus obtain a varying number of single models per label set. Verbatim-level averaged *fasttext*-vectors (Bojanowski et al., 2017) are used as input and one-hot vectors per label as targets. We use *nltk* (Bird et al., 2009) for a mild preprocessing of the raw verbatims, dropping punctuation, interviewer annotation and lowercasing. Then, we fit the model using the *scikit-learn* implementation (Pedregosa et al.,

⁶The ANES Conference on Optimal Coding of Open-Ended Survey Data took place in Dec. 2018

⁷Code, data sets and leaderboard available at <https://github.com/mxli417/co.benchmark>.

Table 1: Overview of the prepared data sets of ANES 2008, which our analysis will be based on, and their respective topics. Additional details and descriptive statistics about the data sets can be found in Appendix 7.1.

ID	Topic	Question ID	n	#labels
1	General Election	T5, T6	238	34
2	Primary Election	T2, T3	288	29
3	Party (Dis-)Likes	C1b, C1d, C2b, C2d	4393	33
4	Person (Dis-)Likes	A8b, A8d, A9b, A9d	4672	34
5	Terrorists	S1	2100	26
6	Important Issues	Q3a1, Q3a2, Q3b1, Q3b2	8399	72
7	Office Recognition Question: Gordon Brown	J3c	2096	9
8	Office Recognition Question: Dick Cheney	J3b	2094	11
9	Office Recognition Question: Nancy Pelosi	J3a	2094	14
10	Office Recognition Question: John Roberts	J3d	2092	9

2011) in conjunction with `gensim` (Radim Rehurek, 2010) for including the `fasttext`-vectors.

Transfer Learning Architectures. As representatives for the class of transfer learning models we use existing `cased`⁸ implementations of BERT-base (Devlin et al., 2018), RoBERTa-base (Liu et al., 2019) and XLNet-base (Yang et al., 2019) via `simpletransformers`, which is based on the `transformers` module (Wolf et al., 2019). The basic structure of the models is complemented by a multilabel-classification head⁹. The used loss function is `BCEWithLogitsLoss` from `pytorch` *per node* in order to account for the multi-label structure of the targets. We do not intend to perform excessive tuning of hyperparameters, but rather want to evaluate the performance of these models when used "out-of-the-box" for a much more difficult task than the common ones. This approach is also largely in line with recent works extending BERT to multi-label problems (Lee and Hsiang, 2019; Chang et al., 2019). All models were fine-tuned on the data sets for three epochs with a maximum sequence length of 128 tokens and a batch size of eight sequences. (Peak) learning rate for fine-tuning was set to $2e-05$ for every model.

4.3 Evaluation Metrics

Generally, metrics commonly used for the evaluation of machine learning methods in binary or multi-class classification tasks cannot be used for multi-label learning without some further considerations (Tsoumakas and Katakis, 2007). This is mainly due to the fact that the performance of a given classifier should be evaluated over all labels and the partial correctness of a prediction must be taken into account.

⁸Since RoBERTa only exists in a `cased` version, we had to choose the other models analogously.

⁹For implementation details of this head see <https://github.com/ThilinaRajapakse/simpletransformers>

Thus, we here utilize a set of multi-label evaluation metrics reported in overview articles by different authors (Tsoumakas and Katakis, 2007; Sorower, 2010; Gibaja and Ventura, 2014, 2015; Herrera et al., 2016) to assess various aspects of the performance of the classifiers we investigate.

For the following, we resume the previous notation. Let us assume that we have a multi-label test set $T = (\mathbf{x}_i, Y_i) | 1 \leq i \leq n$ with n instances and different label sets Y_i , representing the ground truth, at our disposal. Further, let P_i be the set of predicted labels for a given observation.

First, we will report the widely known F_1 score, which is the harmonic mean of *Precision* and *Recall*

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (1)$$

We report the micro- and macro-averaged versions of this score, as the F_1 score is a binary evaluation measure and one needs to choose an averaging approach in the multi-label case. By doing so, different performance aspects can be investigated (Gibaja and Ventura, 2015). Micro-averaging mainly tends to summarize the classifier performance on the most common categories, whereas macro-averaging tends to report performance on the rare categories of the test set. Values towards 1 are better, the minimum value is 0.

Additionally, we also report the sample-based F_1 score as this is also the central metric Card and Smith (2015) use and report in their paper¹⁰. This version of the F_1 score can be formally described as:

$$F_1^{\text{sample}} = \frac{1}{N} \sum_{i=1}^N \frac{2|Y_i \cap P_i|}{|Y_i| + |P_i|} \quad (2)$$

(cf. Gibaja and Ventura (2014)) where N is the total number of samples in the test set.

Second, we report the *subset accuracy*, often also referred to as *exact match ratio*. It computes the fraction of instances in the data for which the predicted

¹⁰Note that they did not use the same notation, but essentially used the same metric described in a vectorized form.

5. Evaluating pre-trained language models on applications in Social Sciences

ICAART 2021 - 13th International Conference on Agents and Artificial Intelligence

Table 2: Model performances (measured as micro- and macro-averaged F_1 -scores) for all considered architectures. Results are displayed separately for each data set with the best performance per data set in bold. We report F_1^{sample} to ensure comparability to the results reported by Card and Smith (2015).

Dataset-ID		1	2	3	4	5	6	7	8	9	10
n		238	288	4393	4672	2100	8399	2096	2094	2094	2092
#labels		34	29	33	34	26	72	9	11	14	9
F_1^{sample}	Baseline	0.44	0.51	0.57	0.54	0.68	0.88	0.92	0.95	0.90	0.91
	BERT	0.00	0.02	0.44	0.35	0.41	0.79	0.94	0.95	0.91	0.93
	RoBERTa	0.00	0.00	0.56	0.55	0.57	0.85	0.95	0.97	0.93	0.94
	XLNet	0.00	0.00	0.54	0.58	0.55	0.86	0.96	0.98	0.91	0.92
	Card and Smith (2015)	0.55	0.67	0.71	0.71	0.81	0.86	0.94	0.96	0.93	0.96
F_1^{micro}	Baseline	0.40	0.48	0.53	0.51	0.61	0.84	0.89	0.93	0.85	0.90
	BERT	0.00	0.03	0.51	0.44	0.46	0.79	0.94	0.95	0.91	0.93
	RoBERTa	0.00	0.00	0.60	0.60	0.62	0.85	0.96	0.97	0.94	0.95
	XLNet	0.00	0.00	0.59	0.61	0.61	0.85	0.96	0.97	0.90	0.93
F_1^{macro}	Baseline	0.23	0.29	0.33	0.34	0.47	0.46	0.62	0.51	0.56	0.71
	BERT	0.00	0.01	0.11	0.16	0.12	0.09	0.47	0.40	0.39	0.58
	RoBERTa	0.00	0.00	0.18	0.26	0.21	0.14	0.51	0.51	0.44	0.58
	XLNet	0.00	0.00	0.20	0.27	0.21	0.16	0.58	0.53	0.43	0.66

labels *exactly* match their corresponding true labels. This is a very harsh metric, as it does not distinguish between partially and completely incorrect predictions. It is defined as:

$$subset\ accuracy = \frac{1}{N} \sum_{i=1}^N \mathbb{1}(P_i = Y_i) \quad (3)$$

Next, we report the *Label Ranking Average Precision* (LRAP). This metric computes the fraction of labels ranked above a certain label $\lambda \in Y_i$ which belong to Y_i , averaged across all observations (Gibaja and Ventura, 2015). For this, a function $f: X \times Y \rightarrow \mathbb{R}$ is generated by a label-ranking algorithm which orders all possible labels for a given instance \mathbf{x}_i by their relevance (Gibaja and Ventura, 2014). If a given label $\lambda' \in Y_i$ is ranked higher than another label $\lambda \in Y_i$, then $f(\mathbf{x}_i, \lambda') > f(\mathbf{x}_i, \lambda)$ must hold. In the following we consider \hat{f}_λ to be a function which returns the ranking for a given label λ , generated by the used ranking algorithm. Here, the higher the obtained metric results are, the better. The best achievable value is 1. LRAP is defined as (Gibaja and Ventura, 2014):

$$LRAP = \frac{1}{N} \sum_{i=1}^N \frac{1}{|Y_i|} \sum_{\lambda \in Y_i} \frac{|\{\lambda' \in Y_i | \hat{f}_{\lambda'} \leq \hat{f}_\lambda\}|}{\hat{f}_\lambda} \quad (4)$$

The LRAP favors classifiers which can rank the relevant labels associated with each observation higher than the irrelevant ones.

5 RESULTS

We report all of the above mentioned metrics for the baseline model as well as for the three mentioned pre-

trained architectures on the test set. The results will be structured as follows: In Tab. 2 we report macro- and micro-averaged F_1 scores, additionally the sample-based F_1 scores (cf. Card and Smith 2015) will be reported as well. Tab. 3 shows the label ranking average precision LRAP and the *subset accuracy*.

Considering the F_1^{sample} scores from Tab. 2, it becomes clear that all used models can hardly outperform the previous best results. Note that the best model from Card and Smith (2015) on almost all data sets has been a thoroughly tuned logistic regression model with a battery of different features. Overall, the best logistic regression model has outperformed even much more advanced architectures in 7 out of 10 cases, establishing that this kind of model can handle multi-label text classification problems surprisingly well. In line with this, we observe that our baseline can beat the transfer learning architectures on 5 out of 10 data sets. Only RoBERTa and XLNet can beat the previous best results on two data sets by a small margin. On all other data sets the previously set benchmark results remain largely unchallenged.

When focussing on the F_1^{micro} measure, we can see that the more advanced models, especially RoBERTa and XLNet, outperform the baseline as soon as the data set size gets bigger, even if they sometimes demonstrate only a slightly better performance. BERT still performs relatively poorly, and even gets beaten by the baseline on five out of ten data sets. RoBERTa also shows only slightly better performance than the baseline on the data set 5 containing the question on terrorism and the data set 6 on Important Issues. On the remaining data sets, how-

Table 3: Model performances (measured as *LRAP* and *subset accuracy*) for all considered architectures. Results are displayed separately for each data set with the best performance per data set in bold.

Dataset-ID		1	2	3	4	5	6	7	8	9	10
<i>LRAP</i>	Baseline	0.59	0.65	0.70	0.70	0.75	0.93	0.95	0.98	0.92	0.95
	BERT	0.09	0.10	0.41	0.32	0.40	0.71	0.94	0.95	0.90	0.93
	RoBERTa	0.09	0.09	0.51	0.49	0.55	0.79	0.95	0.97	0.93	0.94
	XLNet	0.09	0.09	0.49	0.52	0.53	0.80	0.95	0.97	0.90	0.92
<i>subset acc.</i>	Baseline	0.00	0.10	0.20	0.17	0.35	0.70	0.80	0.89	0.76	0.80
	BERT	0.00	0.00	0.16	0.08	0.20	0.41	0.89	0.90	0.81	0.87
	RoBERTa	0.00	0.00	0.22	0.20	0.32	0.54	0.91	0.94	0.87	0.89
	XLNet	0.00	0.00	0.22	0.22	0.31	0.58	0.92	0.94	0.80	0.87

ever, it can clearly outperform the baseline. XLNet also mostly outperforms the baseline, with the exception of the data set concerning terrorism. On the very small and thus very challenging data sets 1 and 2 which contain questions on the General and Primary Election outcomes, the baseline model still is the best.

Finally, when considering the F_1^{macro} score, we observe that the baseline model is the single best model across almost all data sets. Only for data set 8, the larger RoBERTa and XLNet can match or outperform it. While this might be quite surprising, it proves again that a binary relevance approach with a logistic regression as a base learner can be a quite competitive model – which is exactly the same finding Card and Smith (2015) have reported.

Regarding *LRAP* (cf. Tab. 3), RoBERTa and XLNet can partially match the baseline model especially on the last four data sets, which have a small label set and are reasonably large. But XLNet and RoBERTa also hardly outperform the baseline on all remaining data sets, which makes the baseline model a powerful ranking algorithm. BERT, however, cannot beat the baseline at any of the data sets. For the strict measure *subset accuracy* the baseline is not a strictly superior competitor, as it outperforms the more advanced models only on 3 out of 10 data sets. This is also why it is important to compare several evaluation metrics in multi-label classification, as each metric focuses different performance characteristics (Nam, 2019). Unfortunately, Card and Smith (2015) have not provided any results beyond the F_1^{sample} metric.

After these comparisons we conclude that concerning data sets 1 and 2, which contain 238 and 288 observations respectively, BERT, RoBERTa and XLNet cannot obtain any results above zero. Additionally, these models outperform the baseline only marginally on the data sets regarding the Party (Dis)Likes, Person (Dis)Likes and the Office-Recognition-Question for Dick Cheney. Nonetheless, they can outperform the baseline as soon as the data

sets get larger and the label sets remain relatively small.

6 DISCUSSION

Transfer learning has, in this specific use case, not turned out to be a strong alternative compared to previous research. BERT, RoBERTa and XLNet can not generally outperform previous best results obtained on the same data. Additionally, we observed just like the previous authors that a binary relevance approach with logistic regression can be a strong competitor, sometimes even outperforming advanced models. On small data sets, however, no model achieved good results with respect to the subset accuracy, our harshest metric. This is most certainly due to the size, as the data does not contain much information for automated classifiers to learn from. In this case, relying on hand-coding by humans might still be a good option.

Our findings are somewhat contrary to previously reported results, where BERT was used quite successfully in multi-label classification (Adhikari et al., 2019; Chang et al., 2019; Lee and Hsiang, 2019), even yielding new SOTA results. The data sets these authors have used to train their models, however, were much larger than the ones we can utilize here. As noted previously, we try to generate a benchmark regarding the usability of these models in the context of scarce data, which is common in the social sciences. In the light of the good performance of the baseline model, the bigger models also might not be the best choice if computational efficiency is the goal. As social scientists typically do not have unlimited computing power at their disposal, a model which can be trained to obtain reasonable levels of, for example, subset accuracy, in a short amount of time might be especially interesting for future research. Additionally, this model also can handle smaller data sets significantly better and does not break down on bigger

ones. This might be an indicator to look at smaller, more problem-specific algorithms like feature-based transfer learning to advance the research on automatic coding in the future.

7 CONCLUSION

In this work, we provided an extension to the collection of commonly used benchmark data sets used for evaluation transfer learning models for NLP. The full-text data set encompasses a different task than most of the others and thus widens the opportunities for carefully evaluating pre-trained models on a different kind of challenge. Furthermore we propose a unified pre-processing of the data set going along with a fixed train-test split enabling a valid comparison against our baselines. We evaluated the performance of state-of-the-art transfer learning models on the ANES 2008 data set and compared them to a simple baseline model. Our comparison illustrates that, despite the extremely good performances of those models on binary, multi-class and previous multi-label classification tasks, there is still a lot of room for improvement concerning the performance on challenging multi-label classification tasks on small to mid-sized data sets.

ACKNOWLEDGEMENTS

We want to express our sincere gratitude to Christian Heumann for his guidance and support during the process of this research project. We would like to thank Jon Krosnick and Matt Berent for their insightful explanations via e-mail regarding the *Open Ended Coding Project*. This helped us to develop a better understanding for the initial data format. A special thanks also goes to Dallas Card for his explanations regarding the data splits from Card and Smith (2015).

REFERENCES

- Adhikari, A., Ram, A., Tang, R., and Lin, J. (2019). Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.
- Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. Safari Books Online. O'Reilly Media Inc, Sebastopol.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Card, D. and Smith, N. A. (2015). Automated coding of open-ended survey responses.
- CESSDA Training Team (2020). Cessda data management expert guide.
- Chang, W.-C., Yu, H.-F., Zhong, K., Yang, Y., and Dhillon, I. (2019). X-bert: extreme multi-label text classification using bidirectional encoder representations from transformers. *arXiv preprint arXiv:1905.02331*.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Gibaja, E. and Ventura, S. (2014). Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(6):411–444.
- Gibaja, E. and Ventura, S. (2015). A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, 47(3):1–38.
- Herrera, F., Charte, F., Rivera, A. J., and Del Jesus, M. J. (2016). Multilabel classification. In *Multilabel Classification*, pages 17–31. Springer.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hoyle, L., Vardigan, M., Greenfield, J., Hume, S., Ionescu, S., Iverson, J., Kunze, J., Radler, B., Thomas, W., Weibel, S., and Witt, M. (2016). Ddi and enhanced data citation. *IASSIST Quarterly*, 39(3):30.
- Inter-University Consortium For Political And Social Research (ICSPR) (2012). Guide to social science data preparation and archiving: Best practice throughout the data life cycle.
- Krosnick, J. A., Lupia, A., and Berent, M. K. (2012). 2008 open ended coding project.
- Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. (2017). Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.
- Lee, J.-S. and Hsiang, J. (2019). Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124*.
- Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pre-training approach. *arXiv preprint arXiv:1907.11692*.
- Lupia, A. (2018a). Coding open responses. In Vannette, D. L. and Krosnick, J. A., editors, *The Palgrave Handbook of Survey Research*, pages 473–487. Springer International Publishing, Cham.
- Lupia, A. (2018b). How to improve coding for open-ended survey data: Lessons from the anes. In *The Palgrave Handbook of Survey Research*, pages 121–127. Springer.
- Mencia, E. L. and Fürnkranz, J. (2008). Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference*

on *Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer.

Nam, J. (2019). *Learning Label Structures with Neural Networks for Multi-label Classification*. PhD thesis, Technische Universität.

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., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Rehurek, P. S. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S. K., Albertson, B., and Rand, D. G. (2014). Structural topic models for open-ended survey responses. *American Journal of Political Science*, 58(4):1064–1082.

Sechidis, K., Tsoumakas, G., and Vlahavas, I. (2011). On the stratification of multi-label data. In Gunopulos, D., Hofmann, T., Malerba, D., and Vazirgiannis, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 145–158, Heidelberg. Springer.

Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18:1–25.

Szymański, P. and Kajdanowicz, T. (2017a). A network perspective on stratification of multi-label data. In *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 22–35.

Szymański, P. and Kajdanowicz, T. (2017b). A scikit-based python environment for performing multi-label classification. *arXiv preprint arXiv:1702.01460*.

The American National Election Studies (2015). *ANES 2008 Time Series Study*. Inter-university Consortium for Political and Social Research [distributor].

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Vardigan, M., Heus, P., and Thomas, W. (2008). Data documentation initiative: Toward a standard for the social sciences. *International Journal of Digital Curation*, 3(1):107–113.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized au-

toressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Züll, C. (2016). Open-ended questions. *GESIS Survey Guidelines*, 3.

APPENDIX

7.1 Pre-processed Data Set

Table 4: Multi-label descriptive statistics for our data preparation approach.

ID	Word count		Avg. Words/ Verbatim	Cardinality	Label Density	Powerset Size	Powerset/ Examples			Observations per Label				
	Total	Unique					Min.	Avg.	Max.	Min.	Avg.	Max.		
1	3775	872	15.86	2.895	0.085	238	1.00	1	20.26	73	1.00	1	20.26	73
2	3176	752	11.03	2.205	0.076	288	1.00	1	21.90	114	1.00	1	21.90	114
3	60405	5046	13.75	2.291	0.069	4393	1.00	2	305.00	1507	1.00	2	305.00	1507
4	67659	5136	14.48	2.397	0.070	4672	1.00	1	329.32	1549	1.00	1	329.32	1549
5	26502	2442	12.62	1.947	0.075	2100	1.00	8	157.27	618	1.00	8	157.27	618
6	37652	3548	4.48	2.329	0.032	8399	1.00	1	271.64	8398	1.00	1	271.64	8398
7	9512	742	4.54	1.374	0.153	512	0.24	5	319.89	1363	0.98	1	319.89	1363
8	6711	576	3.20	1.204	0.109	2048	0.98	1	229.18	1384	1.00	3	204.71	899
9	10378	720	4.96	1.369	0.098	2094	0.24	10	310.78	1232	0.24	10	310.78	1232
10	9157	762	4.38	1.337	0.149	512	0.24	10	310.78	1232	0.24	10	310.78	1232

6. Detecting stances of Fake News using pre-trained language models

Chapter 6 addresses the detection of (potential) fake news articles using pre-trained language models, which is in this case a *multi-class classification* task. Further the sensitivity towards a set of selected hyperparameters is evaluated and thus concluding statements about the necessity of tuning these hyperparameters are made. Additional insights are provided by comparing two different data sets as well as considering class-wise performance measures and linking those to the characteristics of the different architectures.

Contributing article:

Guderlei, M. and Aßenmacher, M. (2020). Evaluating Unsupervised Representation Learning for Detecting Stances of Fake News. *Proceedings of 28th International Conference on Computational Linguistics (COLING), Barcelona, Spain (Online), December 8-11, 2020: 6339–6349*. <https://doi.org/10.18653/v1/2020.coling-main.558>.

Copyright information:

This article is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>).

Author contributions:

The paper was jointly written and reworked by both authors. Besides the two authors, Christian Heumann and Matthias Wissel substantially contributed to the extensive task of selecting appropriate hyperparameter combinations for setting up the experiments. Maike Guderlei was responsible for writing most of the code, supported by continuous revision and support of Matthias Aßenmacher. Both of the authors finally revised and proofread the manuscript.

Supplementary material available at:

- Code and data set: <https://github.com/magud/fake-news-detection>
- Poster: <https://www.misoda.statistik.uni-muenchen.de/forschung/coling.pdf>

Evaluating Unsupervised Representation Learning for Detecting Stances of Fake News

Maïke Guderlei

Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
maïke@guderlei.de

Matthias Aßenmacher

Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
matthias@stat.uni-muenchen.de

Abstract

Our goal is to evaluate the usefulness of unsupervised representation learning techniques for detecting stances of Fake News. Therefore we examine several pretrained language models with respect to their performance on two Fake News related data sets, both consisting of instances with a headline, an associated news article and the stance of the article towards the respective headline. Specifically, the aim is to understand how much hyperparameter tuning is necessary when finetuning the pretrained architectures, how well transfer learning works in this specific case of stance detection and how sensitive the models are to changes in hyperparameters such as batch size, learning rate (schedule), sequence length as well as the freezing technique. The results indicate that the computationally more expensive autoregression approach of XLNet (Yang et al., 2019) is outperformed by BERT-based models, notably by RoBERTa (Liu et al., 2019). While the learning rate seems to be the most important hyperparameter, experiments with different freezing techniques indicate that all evaluated architectures had already learned powerful language representations that pose a good starting point for finetuning them.

1 Introduction

With the rise of social media, exchange of opinions and news happens faster than ever. News circulation is therefore less and less bound to traditional print journalism that usually requires extensive research, fact checking and accurate coverage in order to be a reliable news resource. It is relatively easy to share opinions that are either not supported by researched facts or simply wrong. In the worst case a large amount of people can be targeted by propaganda in order to shift societal discussions in favor of a wanted agenda. Human resources are limited to identify such Fake News, since they cover a wide range of topics and linguistic writing styles (Shu et al., 2017, p.2). Automated Fake News detection (FND) has therefore proven to be an important challenge for NLP researchers in recent years. To this day, a variety of approaches dealing with FND exists (Khan et al., 2019).

In 2017, the *Fake News Challenge Stage 1* was introduced which tackles FND as a stance detection task (Pomerleau and Rao, 2017), where the idea is to determine the stance of a news article to a given headline (Hanselowski et al., 2018, p.1). We now evaluate the five models BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019), ALBERT (Lan et al., 2019) and XLNet (Yang et al., 2019) which were developed and enhanced recently. All architectures were pretrained on large unlabeled corpora using a self-supervised objective and can be finetuned on a desired task at hand. Our main focus is to evaluate the necessity of hyperparameter tuning as well as the general performance.¹ Besides this, it is of special interest to examine the differences between auto-encoding (BERT-based models) and auto-regressive architectures (XLNet).

In this context, the term Fake News is defined as a text piece that is verifiably wrong and spread with a malicious intention. In doing so, other media sources such as video, images or audio are excluded. Since the intention has to be malicious all sorts of entertainment related false news such as hoaxes and april fools are excluded. The definition is similar to the narrow definition that Shu et al. (2017) undertake.

This work is licensed under a Creative Commons Attribution 4.0 International License.

License details: <http://creativecommons.org/licenses/by/4.0/>.

¹Code and data sets available on GitHub: <https://github.com/magud/fake-news-detection>

2 The Fake News Challenge (FNC-1)

In 2017, the *Fake News Challenge Stage 1* (FNC-1) was published. Organizers from industry and academia created an online challenge accessible via <http://www.fakenewschallenge.org/> (Pomerleau and Rao, 2017). The FNC-1 is conceptualized as an important pre-step in identifying Fake News and exploring how artificial intelligence tools can be leveraged in combatting them. Given a certain claim about a topic, what are different news agencies reporting about this claim? If most news agencies agree with a claim, this can be interpreted as an indicator of the truthfulness of the claim. On the contrary, if a lot of news disagree with the claim, the claim is likely Fake News. Statistically speaking this idea is translated into a stance detection task with the claim being treated as a headline and the stance of the article body being either *Agree*, *Disagree*, *Discuss* or *Unrelated*. FNC-1 is thus treated as a classification task with four categories which are interpreted as the stance of an article body towards a given headline claim. Along with a baseline model, a training and test set was published. According to Hanselowski et al. (2018), 50 teams participated in the challenge.

3 Related Work

In general, FNC-1 can be interpreted as a binary classification task where a text document is classified as either *Fake News* or *No Fake News* or as a multi-class problem often with ordinal labels. One major distinction in different classification algorithms can be drawn in the consideration of auxiliary information. Most contributions take a purely feature-oriented approach, while others try to incorporate information revolving around spreading Fake News and the respective dissemination process. The feature-based approach focuses on extracting relevant linguistic peculiarities associated with Fake News. Typical examples are characters n-grams, words or a measure of the readability and syntax of an article body (Pérez-Rosas et al., 2018, p.5). Other authors included the average word length, counts of exclamation marks or the sentiment of an article (Khan et al., 2019, p.6). After finding appropriate features, traditional machine learning algorithms as well as (deep) neural networks are proposed to classify the text instance given the extracted features. Other feature-based approaches directly rely on deep learning, using neural networks for learning good representations of the text input by a stack of hidden layers which is then fed in a last classification layer. This approach is used by Yang et al. (2018) who simultaneously train a CNN on text and image data to classify fake entities and by Dong et al. (2019) who implement a two-step approach of using supervised and unsupervised learning with a CNN as well. But despite the fact that feature-based approaches are fairly popular within the FNC-1 research, Shu et al. (2017) argue that they are not sufficient. Approaches that use auxiliary information e.g. revolve around modeling the dissemination process of Fake News by incorporating spatio-temporal information about users who like, share or publish (potential) Fake News (Ruchansky et al., 2017; Ren and Zhang, 2020). The FNC-1 organizers use a feature-based approach by proposing a baseline model that extracts various features from the headlines and article bodies. The approach of specifically modeling the relationship between a headline and a respective article body was also exploited by Yoon et al. (2019) in the context of clickbait detection while the FNC-1 takes this approach as a pre-step to FNC-2.

4 Material and Methods

4.1 Model architectures

Representation learning can be seen as one of the crucial success factors of large pretrained models such as BERT or XLNet that yield outstanding performances on a variety of NLP tasks. With the introduction of BERT (Devlin et al., 2019) - short for Bidirectional Encoder Representations from Transformers - the possibility of *simultaneously* learning left and right word context was introduced. Up to this point, considering bidirectionality was only possible by modeling two separate networks for each direction that would later be combined (Peters et al., 2018). After learning deep bidirectional representations from unlabeled text, BERT can be used for either finetuning or feature extraction. The model is pretrained on the combined objective of masked language modeling (MLM) and next-sentence prediction. The main premise of RoBERTa (Liu et al., 2019) is the assumption that BERT was seriously undertrained during

pretraining. RoBERTa is thus trained on larger batches of longer sequences from a larger pre-training corpus for a longer time. In addition, Liu et al. (2019) argue that the second task of the next-sentence prediction does not improve BERT's performance in a way worth mentioning and therefore remove the task from the training objective. While RoBERTa focuses on improving pretraining, DistilBERT (Sanh et al., 2019) and ALBERT (Lan et al., 2019) were introduced as lighter versions of BERT applying parameter reduction techniques in order to be usable under constrained computational training and inference budgets. Sanh et al. (2019) criticize the idea of simply enlarging data sets and running more and more exhaustive pretraining since this does not consider computational costs, memory requirements and even environmental aspects that are often neglected for the sake of further enhancing performance (Strubell et al., 2019; Hao, 2019; Peng, 2019). DistilBERT ends up diminishing BERT's architecture by around 40% while retaining around 97% of its language understanding capabilities and being 60% faster by using a distinct knowledge distillation approach (Hinton et al., 2015), while ALBERT relies on factorized embedding parameterization and cross-layer parameter sharing. Lan et al. (2019) specifically point out that simply enlarging model architectures by using bigger hidden size dimensions or more layers can lead to unexpected model degradation². While RoBERTa, DistilBERT and ALBERT focus on either improving or reducing the size of BERT, the overall bidirectional encoder architecture with the MLM objective remains the same. With XLNet, Yang et al. (2019) propose an alternative approach that works in an auto-regressive manner. Since BERT's objective is to reconstruct a corrupted input, it can be described as a denoising autoencoder (DAE) approach. In contrast to the denoising autoencoder, auto-regressive (AR) language modeling uses a sequential token prediction that can only condition on either left or right context. XLNet combines the advantages of both approaches, namely the bidirectionality while capturing dependency structures among tokens better by employing a so called permutation language modeling objective (PLM).

The pretrained models are all implemented in PyTorch (Paszke et al., 2019) using the *huggingface transformers* library (Wolf et al., 2019) that makes a large variety of the SOTA models in NLP available and ready to use. Architecture-wise, the `base`-based implementations with the suitable head for sequence classification (i.e. `<model_name>ForSequenceClassification`) are used. For DistilBERT and XLNet the weights of the pooling layers are randomly initialized, while BERT, RoBERTa and ALBERT rely on pretrained pooler layers. The weights for the softmax classification layer are randomly initialized for all five models. For DistilBERT a model version that is distilled from the RoBERTa base model was chosen. All models are finetuned using an Ubuntu 18.04.3 LTS OS image with 40 CPUs (Intel Xeon, 2.4 GHz clock speed), 736 GB of RAM and a maximum of two Tesla V100-PCIE-16GB GPUs.

All models were finetuned using the Adam algorithm with default values for its hyperparameters as indicated by Kingma and Ba (2014). As warmup ratio, we chose a value of 0.06 complemented by a more extensive evaluation when it comes to the learning rate schedule, where we conduct small grid search experiments later on. In order to avoid gradient explosion, the norm of all gradients is clipped with a maximum value of 1. Gradients are not accumulated. For all models, the pretrained weights of the specified *huggingface* version are loaded once. The experiments and grid search steps thus use the same randomly initialized weights for the finetuning layers per model³.

4.2 Data

The FNC-1 data set was created from the Emergent data set (Ferreira and Vlachos, 2016) which was developed for an online journalism project about rumour debunking. The project is still running and a website with manually checked claims is available (Silverman, 2019). Rumours were extracted from websites such as *snopes.com* and twitter accounts such as *@Hoaxalizer*. Journalists then first identified the respective claim and searched for articles mentioning this claim. As a next step the journalists labeled

²However, the performance of BERT, RoBERTa and XLNet can only be exceeded when the model is upscaling its width again which is contrary to the authors initial idea of presenting a leaner model architecture.

³Please note that the term *finetuning layer* refers to the additional layers that are put on top of the main model architecture, while *pretraining layers* are those layers that are part of all models no matter which finetuning task is used. Which layers were updated during the finetuning is not indicated by these terms, but is rather defined by the specified freezing technique.

the article as *For*, *Against* or *Observing* and then summarized the article into a headline. As an additional step the veracity level of the claim was labeled as *True*, *False* or *Unverified*. In total, 300 rumoured claims and 2,595 associated news with an average ratio of 8.65 (sd = 7.31) articles per claim were considered. The data set thus contains real world data which was manually labeled by journalists with regard to their stance and veracity level. For the FNC-1, organizers matched every article body with its respective headline and additionally created the fourth class *Unrelated* by randomly matching headlines and article bodies that belonged to different topics. Furthermore, 266 instances were created in addition to prevent participation teams from deriving labels for the test set since the Emergent data set is publicly available. The class distribution over the four classes in the FNC-1 data set is heavily skewed towards the *Unrelated* class. The 49,972 instances each consist of a headline (i.e. the claim to be looked at), the respective article body and label. In total, there are 1,669 unique article bodies and 1,648 unique headlines. The 300 topics are divided into 200 topics for training and 100 topics for testing. Not every claim is associated with all four labels.

Hanselowski et al. (2018) introduce the extended data set FNC-1 ARC as follows: ARC consists of 188 manually selected debate topics of popular questions from the user debate section of the New York Times. For each of these debate topics those user posts were selected that were highly ranked by other users. These highly ranked user posts were processed by producing two opposing claims for them. Afterwards, crowd workers decided on the stance of the user posts with regard to the two opposing claims and labeled the post as either *Agree*, *Disagree* or *Discuss*. The *Unrelated* label was created by randomly matching user posts to different topics. As this data set is built on user posts as opposed to the online news articles of the original FNC-1 data set, it consists of shorter documents that tend to express one viewpoint only and are less balanced in their opinion as news articles. Using this additional data set, the robustness of the provided models can thus be tested. The extended FNC-1 ARC data set combines the FNC-1 with the ARC data set. It comprises 64,205 instances, 14,233 more than the FNC-1 data set. The label distribution is overall similar to the original data but the *Disagree* category has a bigger proportion.

4.3 Pre-Processing

As a first step, headlines and article bodies are concatenated into one long sequence with headline coming first and the respective article body following. In doing so, the models can be evaluated with respect to their capabilities of learning the semantic structures of one instance as a whole. All model architectures can be used with their own respective tokenizer. Since every tokenizer provides the possibility to process cased text, it was decided to not perform lower-casing. All tokenizers can detect and ignore control characters. It was thus not necessary to explicitly remove them. The removal of seemingly uninformative words, so called stop words, was kept as low as possible and consisted of a manually selected list containing the words *The*, *the*, *A*, *a*, *An*, *an*. In order to remove stop words, it was necessary to first tokenize the instances to be able to filter out the mentioned stop words. After tokenizing each instance, the remaining tokens were padded. Instances that have a longer sequence length than 512 were truncated. As a final result, all tokens have a sequence length of $T = 512$ including all necessary special tokens. In addition, the model is fed with an identifier of which token is padded and which is not. This is necessary to only place the attention over the span of "true" tokens that is of non-padded tokens.

5 Results

5.1 Initial Experiments

The goal of the exploration step is to evaluate the general performance and gain insights as to how useful the given recommendations for hyperparameter choices are. Therefore, the main hyperparameters of interest, namely the sequence length, batch size, learning rate and learning rate schedule are kept fixed to determine how well the models perform with respect to different freezing techniques. The drawn conclusions are then considered for the grid search. The exploration step takes the full maximum sequence length that is available for each model which consists of 512 tokens. The common assumption for batch sizes is that a higher batch size yields a more accurate estimate of the gradients (Goodfellow et al., 2016, p.276). For the given hardware resources this results in a batch size of 8 which is the best

possible value for the given sequence length and memory capacity of the hardware. The learning rate is set to $3e-5$ with a linear schedule. The number of epochs is 2 since it is not the goal of the experimental step to receive the best possible performance but rather to gain first insights into the general adaptability of transfer learning for the stance detection of Fake News.

Concerning freezing, different degrees can be considered: It is obviously possible to finetune all layers, that is to use the pretrained weights as starting point and then update all parameter weights during finetuning. The other extreme would be to only train the additional task-specific layers that are placed on top of the general model structure. For the setup of the initial experiments, three different versions are considered: The first one we call *Freeze*, meaning that all layers except for the last projection as well as the final classification layer are frozen. Second, *No Freeze* uses no freezing at all which means that all parameters are updated during finetuning, while the last approach *Freeze Embed* is done with frozen embedding layers only. For every freezing technique finetuning was performed three times. The results are given in Tab. 1 and reported with respect to the mean macro-averaged F_1 -m metric which is known to handle imbalanced class distributions more effectively compared to the accuracy.

	FNC-1			FNC-1 ARC		
	Freeze	No Freeze	Freeze Embed	Freeze	No Freeze	Freeze Embed
BERT	20.88	75.62	74.93	21.06	75.16	75.31
RoBERTa	20.88	79.27	81.72	21.00	78.89	77.64
DistilBERT	20.88	76.57	76.46	21.00	75.31	76.62
ALBERT	34.66	67.91	68.16	34.70	69.49	69.97
XLNet	27.51	80.95	82.18	25.72	80.20	78.88

Table 1: Mean F_1 -m metric over three runs for the exploration step. For *Freeze* only the last projection and classification layers are updated. For *No Freeze* all layers are updated, while for *Freeze Embed* all embedding-specific layers are excluded from updating. Results are on the dev set of a train/dev split of the actual training set. An overview on the runtimes can be found in Tab. 5 in Appendix A.

The results indicate the importance of not freezing too many layers. All models have problems to accurately learn when the main layers (id est the encoder layers for the BERT-based and the relative-attention with feed-forward layers for XLNet) are frozen. Most models still predict every instance as *Unrelated* after two epochs. ALBERT performs best in this setting. This is not surprising since all encoder layers in ALBERT share weights and the model thus learns less information in general. Freezing the encoder layers therefore leads to less information loss for ALBERT. XLNet performs slightly better compared to BERT, RoBERTa and DistilBERT. For both *No Freeze* and *Freeze Embed* all models are able to accurately learn to classify the given data sets. For the FNC-1 data set, the performances of XLNet and RoBERTa are very close. Overall, there is a slight tendency for models to perform better when only the embedding layers are frozen. The main takeaway is that only finetuning the task-specific layers is not sufficient. Between not freezing any layers and freezing the embedding related layers, the difference in performance with respect to the F_1 -m metric is often neglectable. For the grid search, all models are finetuned with frozen embedding layers since this bears the advantage of speeding up training. Furthermore, models are trained for 3 instead of 2 epochs for the grid search.

5.2 Detailed Grid Search

Since traditional research on hyperparameter optimization focuses on training a model from scratch, some of the generated insights and algorithms might not be appropriate for the setting of transfer learning and finetuning (Li et al., 2020). There is little research on the effectiveness and necessity of hyperparameter optimization when finetuning, especially for NLP tasks. The biggest difference between pretraining and finetuning lies in the initialization of the weights. In the case of finetuning this initialization relies on the pretrained model which hopefully captures some intrinsic knowledge, whilst the pretraining phase usually works with random initialization. The goal of this experimental setup is thus to

gain more insights on the effectiveness and necessity of hyperparameter tuning for finetuning in an NLP context. Therefore, a grid search for the hyperparameters batch size, maximal sequence length and type of learning rate schedule and learning rate is conducted as indicated in Tab. 2.

The learning rate is the most common hyperparameter that is tuned. It is so important that Goodfellow et al. (2016) state to only tune the learning rate, if one has only time/budget to address one hyperparameter. For the finetuning setting, it is usually assumed that a drastically smaller learning rate should be used in comparison to pretraining (Li et al., 2020, p.1). During pretraining the model already learns fairly good representations which are valuable for any downstream task. These should therefore not be destroyed by using a too big learning rate that strides away too fast from the already gained knowledge. Thus, learning rate values of 1e-5, 2e-5, 3e-5 and 4e-5 are considered.

Hyperparameter	Considered Configurations
Batch size/Sequence length	16 / 256; 32 / 256; 4 / 512; 8 / 512
Learning rate	1e-05; 2e-05; 3e-05; 4e-05
Learning rate schedule	constant (<i>cst</i>), linear (<i>lin</i>), cosine (<i>cos</i>)

Table 2: Search space over chosen hyperparameters. The sequence length and batch size depend on one another due to memory capacity reasons. For the longer sequence length only smaller batch sizes could be considered. All learning rate schedules use a warmup period of 6% of the total optimization steps.

In addition, the schedule that is used along with the learning rate itself plays an important role. In this context, three different schedule types are considered, namely a constant, linear and a cosine schedule. All three types of schedules use the same warmup strategy for which a lower learning rate is used at the start of training to overcome optimization difficulties. After a warmup period the targeted learning rate is reached. The schedules now differ in the successive handling of the learning rate. The constant schedule keeps it at the targeted value, while the linear and cosine schedule decay it accordingly.

For the sequence length the naive assumption is that using as much context as possible is beneficial for the performance. A longer sequence length that does not truncate input sequences might therefore perform better. On the other hand it is imaginable that news articles might not need to be fed fully to the model since they might contain redundant discussion parts. Often, the main arguments are already shared at the beginning of the article with the full article further elaborating on the initially made statements. It might be sufficient to look at the beginning of articles which translates to a shorter sequence length. For the grid search, sequence lengths of $T_1 = 256$ and $T_2 = 512$ are examined.

Lastly, a shorter sequence length bears the possibility to increase the batch size which was found to be rather low in the exploration step due to limited memory capacities. A larger batch size is usually affiliated with a more accurate estimate of the gradient (Goodfellow et al., 2016, p.276). On the contrary, smaller batch sizes often have a regularizing effect which might be due to the additional noise they add to the learning process. The values for the batch sizes are chosen such that a constant tokens per batch ratio is reached. Given the memory constraints, the highest possible batch sizes are 32 and 8 for a sequence length of $T_1 = 256$ and $T_2 = 512$ respectively. Given the defined search space in Tab. 2, 48 combinations are evaluated for each model and data set. The combinations are examined per learning rate and presented in Tab. 3.

When it comes to the sequence length, a value of 256 is preferred by most models for the FNC-1 data set, while for the FNC-1 ARC data set there is no preferred choice. This means that even though, the FNC-1 data set contains longer sequences on average, a shorter sequence length is often preferred. This can be interpreted as an indication that the similarity of instances is more important than the average sequence length. Apart from the fact that the FNC-1 ARC data set contains more instances compared to the FNC-1 data set, the biggest distinction is that the additional instances were generated from a different context. Hanselowski et al. (2018) specifically introduced the extended data set to test a proposed model’s robustness by using the more heterogeneous FNC-1 ARC data set. The general performance on the different data sets is elaborated at a later point when discussing the indications of Tab. 4.

6. Detecting stances of Fake News using pre-trained language models

For a sequence length of $T_2 = 512$ a batch size of 8 leads to the best performance more often. The only exceptions are found for a smaller learning rate of $1e-5$ or $2e-5$. This is not surprising since a smaller batch size introduces more uncertainty in estimating the gradient. This uncertainty is then compensated by a smaller learning rate. For a sequence length of $T_1 = 256$ the affiliated preferred batch size is more evenly distributed. For the FNC-1 data set a sequence length of 256 is chosen 14 times in total with eight configurations preferring a batch size of 16 and six configurations one of 32. Hence, for the FNC-1 data set a smaller batch size is preferred for a sequence length of 256. For the FNC-1 ARC and the same sequence length a similar even distribution can be reported, with the batch size 16 being chosen four and the batch size 32 being chosen five times. There is thus an indication that the batch size seems to be especially important for longer sequence lengths where a higher batch size is preferred unless the models are trained on a very small learning rate. For both data sets a higher batch size of 32 tends to occur along with a higher learning rate of $4e-5$.

	BERT			RoBERTa		DistilBERT		ALBERT		XLNet	
	LR	Winner	F_1 -m	Winner	F_1 -m	Winner	F_1 -m	Winner	F_1 -m	Winner	F_1 -m
FNC-1	1e-5	16,256,cos	62.46	4,512,lin	78.18	8,512,cst	65.72	4,512,lin	56.62	4,512,cos	73.47
	2e-5	16,256,cst	70.18	16,256,lin	76.54	16,256,lin	67.64	8,512,cos	59.74	16,256,cos	75.00
	3e-5	16,256,cst	69.36	32,256,cos	76.52	32,256,cst	69.64	16,256,lin	59.80	32,256,cos	73.27
	4e-5	8,512,lin	68.09	32,256,lin	74.84	32,256,cst	72.11	16,256,lin	58.33	32,256,lin	73.46
FNC-1 ARC	1e-5	8,512,lin	68.87	4,512,lin	78.19	8,512,lin	71.99	8,512,cst	63.40	4,512,lin	74.42
	2e-5	4,512,lin	72.20	8,512,lin	77.27	8,512,cst	73.59	8,512,cos	65.01	8,512,lin	75.47
	3e-5	8,512,cos	70.93	16,256,lin	77.54	32,256,lin	72.99	16,256,lin	64.67	16,256,lin	73.97
	4e-5	32,256,lin	70.83	32,256,lin	77.54	16,256,lin	73.13	32,256,lin	63.63	32,256,lin	75.57

Table 3: Overview over the results of the grid search with respect to the learning rate (LR) in the left. *Winner* denotes the winning configuration (chosen with respect to F_1 -m on the evaluation set) out of the 12 possible configurations per LR. The values in the *Winner* column indicate batch size, sequence length and LR schedule (abbreviated by *cst* for constant, *lin* for linear and *cos* for cosine) in this order. The F_1 -m of the winning configuration per model is indicated in bold, values in teal indicate the overall winning configuration per data set. The overall winning configuration over both data sets (RoBERTa; F_1 -m = 78.19) is additionally marked by a box. All reported values are obtained on the official test set.

Evaluating the learning rate schedule, the most surprising finding is that the linear schedule is chosen most frequently with being the preferred choice in 25 of all 40 reported winning configurations. The constant and the cosine schedule are very much on par with being chosen seven and nine times respectively. For the FNC-1 data set the distribution of the schedulers is more equal with nine configurations choosing the linear, five the constant and six the cosine schedule. For the FNC-1 ARC data set a clear preference towards the linear scheduler can be observed. It seems that a linear decay is sufficient but important for both data sets. There are no peculiarities when it comes to ALBERT which uses a different optimizer during pretraining.

When it comes to the learning rate, most models perform best for a smaller value of either $1e-5$ or $2e-5$. There is a difference in the two data sets however. For FNC-1 ARC the overall winning configuration is always either one of the two smaller learning rates for all models. Looking at the FNC-1 data set, DistilBERT and ALBERT yield an overall winning configuration with a learning of $4e-5$ and $3e-5$ respectively. For DistilBERT this can be explained by the fact that it only uses half the layers compared to its teacher RoBERTa which might require the model to stride away more from its pretrained version in comparison to the other models in order to adequately capture the given downstream task. For FNC-1 ARC this might not be observable since the heterogeneous data set requires a smaller learning rate in general. Both DistilBERT and ALBERT perform relatively bad for a learning rate of $1e-5$ on the FNC-1 data set. Given the data set of the finetuning task is relatively homogeneous in its instances, lighter BERT-based frameworks require a larger learning rate than $1e-5$ in order to achieve a better performance.

The general best performing learning rates are $1e-5$ and $2e-5$. It seems that a more cautious updating of the pretrained parameters is important which is a strong indication of how well the large pretrained models already perform. Li et al. (2020) note that "it is believed that adhering to the original hyperparameters for finetuning with small learning rate prevents destroying the originally learned knowledge or features." (Li et al., 2020, p.1). A small learning rate is seemingly the most important factor for correctly using large pretrained NLP models for downstream tasks. BERT and RoBERTa prefer a learning rate of $2e-5$ and $1e-5$ respectively and show the same tendencies in performance with respect to the learning rate for both data sets. Thus this recommendation is relatively stable. DistilBERT and ALBERT prefer a larger learning rate for the more homogeneous FNC-1 data set and a smaller learning rate for the more heterogeneous FNC-1 ARC data set. Later, it will become evident that ALBERT's better performance for the FNC-1 ARC data set can be explained largely by the improved prediction strength on the sparse category of *Disagree* instances. The more evenly distributed class labels of FNC-1 ARC have the biggest impact on BERT and ALBERT.

Looking at the overall performance, ALBERT performs the worst and even worse than BERT. This confirms the statements of Lan et al. (2019) who have reported that ALBERT can only outperform BERT for the xlarge and xxlarge variant. Surprisingly, DistilBERT can outperform BERT on the FNC-1 ARC data set and partly on the FNC-1 data set. When comparing the two different approaches of DAE versus AR, the best BERT-based model (RoBERTa) performs better than XLNet. XLNet still outperforms BERT, DistilBERT and ALBERT on both data sets and is thus the second-best overall model. However, XLNet bears the additional disadvantage of a much longer training time than any BERT-based model. While RoBERTa trains for around 59 minutes for one configuration, XLNet takes around 140 minutes which is more than twice the finetuning time of RoBERTa. All other BERT-based model train faster than RoBERTa. The reason why XLNet does not beat RoBERTa might originate from the specific advantage that XLNet introduces. Yang et al. (2019) criticize BERT for making the assumption that all masked tokens in a sequence are independent. By using the PLM objective, XLNet can avoid making such an assumption and is furthermore able to capture dependencies between tokens better than BERT-based architectures. The given task of stance detection is not a token-level but a segment-level task. The advantage of XLNet of better capturing the dependencies between tokens might thus be not of much use in this context. The overall winning model is RoBERTa with a batch size of 4, a sequence length of 512 and a learning rate of $1e-5$ with a linear schedule for both data sets.

The evaluation done so far focused on the F_1 -m metric. Since both data sets consist of unevenly distributed and heavily skewed class labels, the class-wise F_1 is now considered in order to gain further insights: In general, the performance improves for the extended data set FNC-1 ARC for all models. All models except for RoBERTa can drastically improve their prediction strength for the *Disagree* class which has the fewest training instances, precisely 1.7% (FNC-1) and 3.5% (FNC-1 ARC). RoBERTa can not push the performance on the extended data set as much on this category as other models. This could be interpreted as some sort of saturation effect, since using more evenly distributed data only helps to a certain degree. The FNC-1 ARC data set is still heavily skewed and in addition more heterogeneous than the FNC-1 data set. If the overall model architecture is already very powerful, as is the case for RoBERTa, the heterogeneity in the training instances might outweigh the positive factor of having more evenly distributed data.

XLNet can boost its performance for the extended data set more, in relative comparison to RoBERTa. This can be largely attributed to the boosted performance on the *Disagree* class. All models perform relatively well on the *Discuss* and extraordinarily well on the *Unrelated* class. The worst F_1 -UNR values occur for ALBERT with 96.65 for the FNC-1 and 96.83 for the FNC-1 ARC data set.

RoBERTa is the strongest model since it also outperforms XLNet. This better performance is not attributable to one specific class, since RoBERTa performs stronger on all classes. When considering F_1 -m, all models improve their performance when finetuned on more data, especially on the hardest to predict class *Disagree*.

In summary, the grid search showed a very strong performance of RoBERTa which also outperforms XLNet. This might be due to the specific FND task that is on a segment- rather than a token-level. In

6. Detecting stances of Fake News using pre-trained language models

Metric	BERT		RoBERTa		DistilBERT		ALBERT		XLNet	
	FNC-1	+ ARC	FNC-1	+ ARC	FNC-1	+ ARC	FNC-1	+ ARC	FNC-1	+ ARC
F_1-m	70.18	72.20	78.18	78.19	72.11	73.59	59.80	65.01	75.00	75.57
F_1 -AGR	60.31	63.48	70.69	70.57	61.95	65.29	53.19	53.97	68.00	68.57
F_1 -DSG	41.76	48.28	56.15	58.92	45.09	50.46	13.21	34.07	49.47	53.69
F_1 -DSC	80.36	78.82	86.78	84.16	82.83	80.22	76.16	75.18	83.73	81.43
F_1 -UNR	98.28	98.22	99.10	99.09	98.58	98.38	96.65	96.83	98.80	98.60

Table 4: Model performances with respect to class-wise F_1 as well as F_1 -m in comparison for FNC-1 and FNC-1 ARC. For better readability we indicate the columns for FNC-1 ARC just with "+ ARC".

in addition to performing better, RoBERTa also trains much faster than XLNet. The learning rate is the most important hyperparameter which is in line with current research. For BERT and RoBERTa a clear recommendation for the learning rate can be drawn, while DistilBERT, ALBERT and XLNet exhibit their best performances for different values of the learning rate. The sequence length is not as important and dominated by the similarity of training instances within a data set. If the instances are more similar, the models manage finetuning well with shorter sequences. A more heterogeneous data set requires the model to consider more context and thus a longer sequences. As a learning rate schedule, a linear choice is sufficient and a cosine schedule is not necessary per se.

6 Conclusion

The overall conclusion that can be drawn from this work is how powerful and strong the evaluated architectures are for the examined task related to Fake News Detection. Even with minimal hyperparameter tuning and only finetuning for 3 epochs, the models already performed considerably well on both data sets. With respect to the considered hyperparameters the two most important conclusions are to not only finetune the classification and pooling layers that are stacked on top of the pretrained models⁴. Secondly, the most important hyperparameter is the learning rate. The recommendation of Goodfellow et al. (2016) to focus on the learning rate, when one has only time to address one single hyperparameter can be confirmed. At last, the models are relatively robust with respect to the learning rate schedule, the batch size, as long as it is adjusted to the learning rate and to a certain degree also the sequence length. Furthermore, the excessive pretraining approach of RoBERTa can outperform the permutation language model objective of XLNet.

Fake News detection itself is often time treated as a inherently binary task⁵ (i.e. a document either contains Fake News or it does not). The underlying truth, however, might be more complicated, since most of the times a document can not unambiguously be assigned to either of these two classes. This is why our experiments on the FNC-1 data sets, which comprehend Fake News detection as a multi-class classification problem, might be more beneficial for this field of research compared to experiments on data sets with a binary target. We would not go as far as concluding that these findings are without further ado transferable to other Fake News related data sets using different tasks or label sets, but nevertheless think that our findings can serve as starting points for further experiments in related fields.

Acknowledgements

We want to express our sincere gratitude to Christian Heumann (Ludwig-Maximilians-Universität) and Matthias Wissel (Capgemini Deutschland GmbH) for their help and constant support and advice during the process of conducting this research project. We would also like to thank the three anonymous reviewers for their insightful comments and their feedback on our work.

⁴For some models the pooling layer is already incorporated in the main architecture but still it does not suffice to only finetune the last pooling and classification layer.

⁵See for example the *Shared Task 2 (Fake news detection)* of the *KDD 2020 TrueFact Workshop*, where the texts are labeled either as "reliable" or "unreliable".

References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Xishuang Dong, Uboho Victor, Shanta Chowdhury, and Lijun Qian. 2019. Deep two-path semi-supervised learning for fake news detection. *ArXiv*, 1906.05659.
- William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data set for stance detection. *Proceedings of NAACL-HLT 2016*, pages 1163–1168.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
- Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance detection task. *arXiv preprint arXiv:1806.05180*.
- Karen Hao. 2019. Training a single ai model can emit as much carbon as five cars in their lifetimes. <https://www.technologyreview.com/s/613630/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/>. Accessed: 2020-02-18.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.
- Junaed Younus Khan, Md Khondaker, Tawkat Islam, Anindya Iqbal, and Sadia Afroz. 2019. A benchmark study on machine learning methods for fake news detection. *arXiv preprint arXiv:1905.04749*.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ArXiv*, 1412.6980.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotik, and Stefano Soatto. 2020. Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*.
- 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:1907.11692*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.
- Tony Peng. 2019. The staggering cost of training sota ai models. <https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/>. Accessed: 2020-02-20.
- Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.
- Dean Pomerleau and Delip Rao. 2017. Webpage of the fake news challenge 1 (fnc-1). <http://www.fakenewschallenge.org/>. Accessed: 2020-10-23.
- Yuxiang Ren and Jiawei Zhang. 2020. Hgat: Hierarchical graph attention network for fake news detection. *ArXiv*, 2002.04397.
- Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806.

6. Detecting stances of Fake News using pre-trained language models

- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.
- Craig Silverman. 2019. Emergent: A real-time tracker. <http://www.emergent.info/>. Accessed: 2020-10-23.
- Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, 1910.03771.
- Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. 2018. Ti-cnn: Convolutional neural networks for fake news detection. *ArXiv*, 1806.00749.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.
- Seunghyun Yoon, Kunwoo Park, Joongbo Shin, Hongjun Lim, Seungpil Won, Meeyoung Cha, and Kyomin Jung. 2019. Detecting incongruity between news headline and body text via a deep hierarchical encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 791–800.

Appendix

A Finetuning time for different freezing techniques

	FNC-1			FNC-1 ARC		
	Freeze	No Freeze	Freeze Embed	Freeze	No Freeze	Freeze Embed
BERT	00:27:11 (sd = 4 sec)	01:01:45 (sd = 6 sec)	00:57:49 (sd = 6 sec)	00:32:40 (sd = 2 sec)	01:17:33 (sd = 6 sec)	01:12:37 (sd = 11 sec)
RoBERTa	00:28:24 (sd = 11 sec)	01:03:01 (sd = 7 sec)	00:58:59 (sd = 6 sec)	00:33:50 (sd = 5 sec)	01:18:49 (sd = 5 sec)	01:13:40 (sd = 5 sec)
DistilBERT	00:17:05 (sd = 1 sec)	00:35:19 (sd = 2 min)	00:34:08 (sd = 0 sec)	00:19:03 (sd = 2 sec)	00:43:30 (sd = 5 sec)	00:38:33 (sd = 25 sec)
ALBERT	00:20:01 (sd = 3 sec)	00:49:28 (sd = 3 sec)	00:47:36 (sd = 2 sec)	00:25:25 (sd = 3 sec)	01:03:43 (sd = 2 sec)	01:01:14 (sd = 3 sec)
XLNet	01:26:46 (sd = 50 sec)	02:26:01 (sd = 10 sec)	02:20:44 (sd = 18 sec)	01:49:07 (sd = 26 sec)	03:06:21 (sd = 53 sec)	02:58:49 (sd = 25 sec)

Table 5: Mean finetuning time in hh:mm:ss over three runs for the exploration step. For *Freeze* only the last projection and classification layers are updated. For *No Freeze* all layers are updated, while for *Freeze Embed* all embedding-specific layers are excluded from updating. The learning rate was kept fixed at $3e-5$ with a linear schedule. The batch size was 8 with a sequence length of 512 tokens.

7. Pre-trained language models as knowledge bases for automated complaint analysis

Chapter 7 showcases the use of pre-trained language models as potential substitutes for traditional knowledge bases by transferring domain-specific knowledge into the architecture’s weights. It shows how this can be achieved by continual pre-training, explains the creation of an appropriate test set for probing the models and finally performs experiments using several state-of-the-art pre-trained architectures. This contribution shows rather promising results which indicate that this approach might (in the future) present a valid alternative to expensive handcrafted knowledge bases for some applications.

Contributing article:

Viellieber, V. D. and Aßenmacher, M. (2020). Pre-trained language models as knowledge bases for Automotive Complaint Analysis. *arXiv preprint arXiv:2012.02558*. <https://arxiv.org/abs/2012.02558>.

Copyright information:

This article is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>).

Author contributions:

The paper was jointly written and reworked by both authors. Matthias Aßenmacher mostly contributed by bringing up the idea of continual pre-training, designing the experimental setup of interleaved evaluation on the test set and selecting the models to be evaluated. Vanessa Viellieber brought in her extensive in-domain knowledge from the automotive sector by designing the test set for probing the pre-trained model’s ability to act as a knowledge base.

A similar split can be reported when it comes to the writing and the coding: While Matthias Aßenmacher wrote the theoretical parts and put the results into context, Vanessa Viellieber wrote the practical parts and did the programming. Furthermore, Christian Heumann contributed substantially by discussing the paper’s content, providing valuable hints regarding its structure and jointly revising the code with both of the authors. Both authors proofread the paper.

PRE-TRAINED LANGUAGE MODELS AS KNOWLEDGE BASES FOR AUTOMOTIVE COMPLAINT ANALYSIS

A PREPRINT

Vanessa Deborah Viellieber
Data Science and Artificial Intelligence
MHP - A Porsche Company
Ludwigsburg, Germany
vanessa.viellieber@mhp.com

Matthias Aßenmacher
Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
matthias@stat.uni-muenchen.de

December 4, 2020

ABSTRACT

Recently it has been shown that large pre-trained language models like BERT (Devlin et al., 2018) are able to store commonsense factual knowledge captured in its pre-training corpus (Petroni et al., 2019). In our work we further evaluate this ability with respect to an application from industry creating a set of probes specifically designed to reveal technical quality issues captured as described incidents out of unstructured customer feedback in the automotive industry. After probing the out-of-the-box versions of the pre-trained models with fill-in-the-mask tasks we dynamically provide it with more knowledge via continual pre-training on the Office of Defects Investigation (ODI) Complaints data set. In our experiments the models exhibit performance regarding queries on domain-specific topics compared to when queried on factual knowledge itself, as Petroni et al. (2019) have done. For most of the evaluated architectures the correct token is predicted with a $Precision@1$ ($P@1$) of above 60%, while for $P@5$ and $P@10$ even values of well above 80% and up to 90% respectively are reached. These results show the potential of using language models as a knowledge base for structured analysis of customer feedback.

1 Introduction

Recently researchers developed some interest in the knowledge stored in the large pre-trained models. Petroni et al. (2019) investigated BERT (Devlin et al., 2018) and other architectures with respect to their ability of storing commonsense factual knowledge. As the stored knowledge depends heavily on the pre-training corpus, we are curious about whether one can "teach" these kinds of models further knowledge by exposing them to texts from specific domains, like customer complaints in the automotive industry.

Especially for product-driven organizations as car manufacturers, customer feedback provides a precious source of information for product improvements, e.g. in terms of potential security risks identified and mentioned by customers. However, the structured use of this data is an open problem in industry, despite numerous investigations with advanced NLP methods (Choe et al., 2013; Lee et al., 2015; Akella et al., 2017; Liang et al., 2017; Joung et al., 2019). Handling this fuzzy data and satisfying the demand for detailed information extraction in an intelligent manner remains challenging.

The recent developments in NLP lead us to the idea of evaluating the ability of pre-trained language models to act as a domain-specific knowledge base. We investigate if a language model, further pre-trained on customer feedback, is able to store customer opinions about products, features, and services as knowledge in model parameters.

2 Related work

Besides the challenges of vast customer complaints with free flow writing, different languages, abbreviations, misspellings, domain-specific entities, what makes the analysis of customer complaints so heavily complex is that customer issues occur in so many different forms and combinations. With supervised learning methods the limits are that customer feedback analysis only works along a certain number of categories (Liang et al., 2017; Akella et al., 2017). With these methods it is impossible to automatically identify newly emerging, possibly security-relevant, risks. The lack of labeled training data as well as imbalanced data is a hurdle in the development of NLP models in industrial customer feedback analysis (Choe et al., 2013; Akella et al., 2017). Furthermore, it is relevant for product manufacturers to be able to identify implied connections in customer feedback. Lee et al. (2015) used Web Ontology Language (OWL) to express semantic relations in the customer complaint analysis. But schema engineering strongly needs human supervision and is very time-consuming (Lee et al., 2015; Wang et al., 2016). Therefore this depicts an expensive process (for industrial application) that some product manufacturers shy away from and thus leave a great deal of information unused.

As recently observed by Petroni et al. (2019) and Zhang et al. (2019), language models can store implicit knowledge after pre-training and thus act as a kind of knowledge base, which could be a solution for the predominant challenges in customer complaint analysis. Therefore we use these findings to create domain specific probes for automotive industry in the style of Petroni et al. (2019). They used a general corpus of facts representing knowledge statements like "iPod Touch is produced by Apple". After converting these facts into cloze statements they query the language model asking it to fill in a masked token, which was always the *object* of a fact triple (*subject - predicate - object*). For the evaluation of the models they determined how high the ground truth token was ranked against every other word in a fixed candidate vocabulary.

3 Materials and Methods

We evaluate a selection of pre-trained language models for the English language which recently achieved state-of-the-art results on frequently used benchmarks, for which we use the stable implementations via the unified API of the `transformers`¹ module (Wolf et al., 2019). As we want to infer the effect of continual pre-training on a domain-specific corpus, we use the architectures with the respective heads (e.g. BertForMaskedLM²).

As in-domain corpus for the continual pre-training we use a collection of roughly 500.000 publicly available e-mails containing customer complaints of various vehicle makes. Since this corpus should inherit many details which indicate deeper knowledge of the original equipment manufacturers (OEMs) products, we suspect that continual pre-training transfers this knowledge into the model's weights.

The evaluation set of probes is handcrafted from a held-out set from the same corpus which we utilize for continual pre-training. We created a dictionary containing technical terms of automotive industry and subsequently filtered the corpus using this dictionary in order to obtain suitable sentences where model parts could be masked for probing the model.

3.1 Pre-trained language models

BERT (Devlin et al., 2018) is a bidirectionally contextual transformer encoder model, which is available in a *base* (~110M parameters) and a *large* (~340M parameters) variant. The *base*, *uncased* variant will be used as baseline model. The *cased* variants for both model sizes are not considered since the raw data is only available capital letters and is thus lower-cased during pre-processing. The competing RoBERTa model (Liu et al., 2019) is an optimized, but architecturally alike, version of BERT pre-trained on a significantly larger corpus. We will also compare BERT to DistilBERT (Sanh et al., 2019) and ALBERT (Lan et al., 2019), two models which employ parameter reduction to the original BERT architecture. While DistilBERT relies on knowledge distillation (Hinton et al., 2015), ALBERT primarily makes use of cross-layer parameter sharing techniques (Lan et al., 2019).

3.2 Data for continual pre-training

We use data from the ODI³ of the National Highway and Traffic Safety Administration (NHTSA). NHTSA-ODI's data set consists, amongst other sources, of vehicle owner's complaints regarding different manufacturers and is used to

¹<https://github.com/huggingface/transformers>

²https://huggingface.co/transformers/model_doc/bert.html#bertformaskedlm

³<https://catalog.data.gov/dataset/nhtsas-office-of-defects-investigation-odi-complaints>

7. Pre-trained language models as knowledge bases for automated complaint analysis

A PREPRINT

identify safety issues that warrant investigation and to determine if a safety-related defect trend exists. Other sources are e.g. consumer action groups or insurance companies. In our work, we filtered the data base only for direct customer complaints in order to obtain a data set most alike to customer complaints as they are addressed to OEMs and documented in their CRM system.

The final corpus consists of 502.445 distinct complaints and has a size of 142.776 kilo bytes. We divide the data with a ratio of 90 to 10 into training and held-out set. The whole corpus has a vocabulary of 139.982 distinct words on which we, besides lower-casing, did not perform any other pre-processing steps. Descriptive statistics are displayed in Tab. 1, we use the respective tokenizers for the models.

	avg. length	min	25%	50%	75%	max
raw data	35.94	1	30	41	43	71
tokenized (BERT base uncased)	45.47	1	38	51	54	91
tokenized (BERT large uncased)	45.55	1	38	51	54	106
tokenized (RoBERTa base)	49.33	3	42	54	58	124
tokenized (DistilBERT base uncased)	45.55	1	38	51	54	106
tokenized (ALBERT xxlarge)	48.70	3	40	54	58	130

Table 1: Descriptives for the raw and the tokenized sequences of the train set. BERT uses a word-piece tokenizer, while ALBERT uses sentence-piece. RoBERTa and DistilBERT rely on a BPE-tokenizer.

3.3 Set of evaluation probes

We extracted the affected components from the variable `compdesc` (*description of affected components*) in ODI’s data set. To stick to the one-token logic of [Petroni et al. \(2019\)](#), we split up compounds and thus get a dictionary of 364 distinct, mainly technical, terms related to the vehicle. Due to the decomposition of the compounds, it is sometimes the case that individual terms do not necessarily describe a vehicle component unambiguously, but are nevertheless mainly of a technical nature or relevant in automotive customer complaints. An excerpt of the most frequent ten entries can be seen in Tab. 2.

term	engine	hydraulic	service	brakes	system	cooling	antilock	air	power	seat
frequency	42903	31861	30058	30044	27874	17396	16391	16301	16154	15874

Table 2: Most frequent technical terms to be replaced for creating the probes.

We identified the sentences which contain the relevant terms and masked one of these terms per sequence to prepare for the masked language modeling task. Tab. 3 shows exemplary evaluation probes.

original	<i>gear shift cable failure in auto transmission</i>
masked	<i>[CLS] [MASK] shift cable failure in auto transmission [SEP]</i>
masked	<i>[CLS] gear [MASK] cable failure in auto transmission [SEP]</i>

Table 3: Exemplary probe from our test set. One original sentence can lead to multiple probes if it contains more than one affected component. Multi-token components are not replaced by one MASK.

4 Results

We evaluate the models at regularly spaced intervals defined by the number of in-domain examples seen by a model. Performance values for all five different models are displayed in Tab. 4.

Overall we can see that across all different models there is only little ability for successfully performing this task when simply used ”out-of-the-box”, only ALBERT ([Lan et al., 2019](#)) already consistently predicts domain relevant terms. E.g. for the second masked sequence from Tab. 3 ALBERT’s ”out-of-the-box” version predicts the following top five tokens: (1) ’wrench’, (2) ’axle’, (3) ’shifter’, (4) ’shift’ (ground truth), (5) ’switch’ which contain the ground truth on rank four. The other models do not predict solely technical terms in such a consistent way. When pre-trained on our domain-specific corpus in a continual fashion, we can see that performance steadily increases for each of the examined

	#in-domain examples	out-of-the-box	100k	200k	300k	400k
BERT	base, uncased	12.5 (25.8/ 30.2)	44.5 (71.5/ 78.3)	49.5 (74.2/ 80.5)	58.2 (80.5/ 85.6)	59.0 (81.6/ 86.4)
	large, uncased	27.1 (57.1/ 65.0)	49.7 (74.1/ 80.6)	55.0 (78.3/83.9)	64.6 (85.2/ 89.0)	63.8 (84.8/ 88.8)
Others	RoBERTa (base)	35.8 (61.8/ 71.1)	52.2 (79.9/ 87.7)	51.8 (80.0/ 88.6)	51.2 (80.2/ 89.1)	51.5 (80.8/ 89.2)
	DistilBERT (base)	25.5 (47.8/ 57.4)	56.5 (78.4/ 84.0)	58.1 (79.4/ 84.8)	60.0 (81.1/ 86.1)	61.1 (82.1/ 86.9)
	ALBERT (xxlarge)	43.3 (70.0/ 78.3)	60.4 (83.7/ 89.8)	59.3 (83.7/ 90.7)	60.6 (84.4/ 91.2)	59.0 (84.0/ 91.2)

Table 4: Model performances on the test set measured as Precision@k $P@1$ ($P@5/P@10$) in percent for all considered architectures. Results are displayed separately for different amounts of additional in-domain examples used for continual pre-training. Best performance per column in bold.

models, partly by a large margin. After having seen 100k domain-specific examples, the values for $P@1$ stagnate for all of the architectures. For RoBERTa and ALBERT this happens rather early (after between 100k and 200k examples), while both BERT variants as well as DistilBERT still show smaller improvements until having seen all 400k examples. When relaxing the performance measure by considering $P@5$ and $P@10$ we observe similar behavior for the BERT variants and for DistilBERT. RoBERTa and ALBERT (contrary to before) are now also showing steady, decreasing improvements until the end of continual pre-training. The superiority of ALBERT, which we already observe for the "out-of-the-box" version, mostly prevails over the course of the evaluation intervals. Only the `large` variant of BERT is able to compete when having seen many (300k – 400k) in-domain examples. Concerning the choice of a suitable performance measure, $P@1$ might be a little bit too harsh, since in some cases plausible alternative are ranked above the ground truth. This is a point which requires further evaluation of qualitative nature.

5 Discussion

There are several limitations which we have not yet been able to address. First and foremost multi-token words represent a problem, especially in our domain of application. A significant amount of car parts (e.g. *coolant system*) and key issues (e.g. *not working properly*) are not represented by single tokens, but instead have multi-token representations. Following Petroni et al. (2019), probing *multi-tokens* and *predicates* remains an open challenge. Further extensions of this work will try to address this shortcoming. In addition, we would like to investigate the effects on the model when focal points of different components are described by the same defect patterns, under which conditions the model is then able to differentiate. Moreover we are interested in the time trend within the language models: Can knowledge for a certain temporal period be mapped or queried as such? It is also necessary to investigate the velocity with which new technical issues can be identified in a knowledge base.

6 Conclusion

We took an approach already investigated by Petroni et al. (2019) on some tentative probes for commonsense factual knowledge and extended it via continual pre-training on a corpus from a specific domain. Our experiment shows promising results which indicate that pre-trained language models have the ability of representing focal points gathered from customer complaints and are therefore able to represent domain-specific knowledge. Concluding, language models could be an innovative approach to handle the predominant problems in industry to utilize the full potential of unstructured information in customer feedback at economically acceptable expense. According to our industry experience, both the problem and the solution can be transferred from the automotive domain to other product-driven industries. Nevertheless, there are still some severe limitations to the use of language models as a customer opinion base. These limitations require further investigation and have to be overcome eventually in order to replace scheme-based knowledge bases with these self-supervised approaches.

References

- Akella, K., Venkatachalam, N., Gokul, K., Choi, K., and Tyakal, R. (2017). Gain customer insights using nlp techniques. *SAE International Journal of Materials and Manufacturing*, 10(3):333–337.
- Choe, P., Lehto, M. R., Shin, G.-C., and Choi, K.-Y. (2013). Semiautomated identification and classification of customer complaints. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 23(2):149–162.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Joung, J., Jung, K., Ko, S., and Kim, K. (2019). Customer complaints analysis using text mining and outcome-driven innovation method for market-oriented product development. *Sustainability*, 11(1):40.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Lee, C.-H., Wang, Y.-H., and Trappey, A. J. (2015). Ontology-based reasoning for the intelligent handling of customer complaints. *Computers & Industrial Engineering*, 84:144–155.
- Liang, R., Guo, W., and Yang, D. (2017). Mining product problems from online feedback of chinese users. *Kybernetes*.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. (2019). Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.
- Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Wang, Y., Cheng, X., Xu, L., Guan, J., Zhang, T., and Mu, M. (2016). A novel complaint calls handle scheme using big data analytics in mobile networks. In *Signal and Information Processing, Networking and Computers: Proceedings of the 1st International Congress on Signal and Information Processing, Networking and Computers (ICSINC 2015), October 17-18, 2015 Beijing, China*, page 347. CRC Press.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv, abs/1910.03771*.
- Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. (2019). Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

8. Applying pre-trained language models for measuring Customer Centricity

Chapter 8 showcases how multiple pre-trained language models can be seamlessly integrated in a pipeline-like structure with the goal to measure and to visualize a specific concept from the field of Insurance Marketing. The chapter describes the necessary pre-processing and data annotation steps which had to be taken prior to the application of two different state-of-the-art pre-trained architectures, namely DistilBERT (Sanh et al., 2019) and LCF-BERT (Zeng et al., 2019). This contribution depicts an interesting use case that indicates how such architectures can easily be used in a fast and productive manner in real world applications.

Contributing article:

Lebmeier, E., Hou, N., Spann, K., and Aßenmacher, M. (2021). Creating a “Customer Centricity Graph” from unstructured customer feedback. *Applied Marketing Analytics, Vol. 6(3): 221–229*. <https://www.henrystewartpublications.com/ama/v6>.

Copyright information:

Henry Stewart Publications, Applied Marketing Analytics, 2021.

Author contributions:

Korbinian Spann approached Matthias Aßenmacher and brought up the idea of jointly supervising a student consulting project on this topic. Elisabeth Lebmeier and Naiwen Hou worked on this project under close academic guidance and supervision of Christian Heumann and Matthias Aßenmacher. Korbinian Spann supervised the project on behalf of the external project Partner, Insaas GmbH. He also provided the data set and the keyword dictionaries as well as the expertise regarding the automotive insurance sector. Matthias Aßenmacher contributed by advising both of the students regarding architecture choice, programming decisions and other technical issues. After finishing the project, the article was jointly written and reworked by all of the authors.

Supplementary material available at:

- Demo of the developed dashboard: [Insaas Vector](#)
- Follow-Up article on the platform LinkedIn: [Know your customer 2.0](#)

Creating a ‘customer centricity graph’ from unstructured customer feedback

Received (in revised form): 14th December, 2020



Elisabeth Lebmeier

Master's Student, LMU Munich, Germany

Elisabeth Lebmeier is a master's student of statistics at LMU Munich, where she is currently working on her thesis about different approaches to aspect-based sentiment analysis. Her research interests include machine learning, deep learning and natural language processing.

Department of Statistics, LMU Munich, Ludwigstr. 33, München 80539, Germany
E-mail: e.lebmeier@gmx.de



Naiwen Hou

Master's Student, LMU Munich, Germany

Naiwen Hou is a master's student of statistics at LMU Munich with an economic and social science background.

Department of Statistics, LMU Munich, Ludwigstr. 33, München 80539, Germany
E-mail: hounaiwen@hotmail.com



Korbinian Spann

Managing Director, Insaas, Germany

Korbinian Spann is Managing Director and Head of Data at Insaas, a software company focusing on artificial intelligence and natural language processing. Dr Spann is responsible for the development of dictionaries and the data pipeline.

Insaas GmbH, Floßmannstr. 20, München 81245, Germany
E-mail: korbinian.spann@insaas.ai



Matthias Aßenmacher

PhD Student, LMU Munich, Germany

Matthias Aßenmacher researches natural language processing at LMU Munich, where he is currently studying for a PhD. His main research areas include *inter alia* the comparability as well as possible applications/use cases of large pre-trained language models.

Department of Statistics, LMU Munich, Ludwigstr. 33, München 80539, Germany
E-mail: matthias@stat.uni-muenchen.de

Abstract Certain industries, such as car insurance, do not have many customer touch points and do not offer a great deal of differentiation in the market. Marketers in such industries must therefore analyse vast amounts of customer-generated feedback in order to analyse customer preference in a quantitative manner. At present, this is done via market research or manual work, as an automated tool for summarising unstructured texts is as yet unavailable for certain European languages, including German. This paper discusses how Insaas and LMU Munich have used publicly available feedback on car insurance in Germany to develop a dedicated pipeline for the computation and visualisation of customer opinions. This paper provides an overview of the various steps of the procedure.

KEYWORDS: customer centricity, NLP, AI, dashboard, B2C

INTRODUCTION

Business-to-consumer (B2C) industries, like car insurance companies, have to focus on their customers' needs in order to provide them with the desired product. As touch points between companies and customers are infrequent, companies must get the most they can out of customer feedback. At present, such feedback is mainly found in unstructured texts that are publicly available on the internet, for instance in comparison portals. Star ratings, in particular, are a popular way for consumers to comment on the overall quality of an insurance company. But this is only a very general approach to a complex issue. More differentiating information can be found in the review texts themselves. In order to avoid manual analysis of this vast amount of data, an automated approach for information extraction and visualisation is needed. Working together, Insaas and LMU Munich have developed a multi-step procedure to solve this problem. The solution can detect topics and their polarity and group this information in such a way that customer opinions can be represented in the form of a graph, known as the *customer centricity graph*. This graph helps companies to identify those areas in which areas they perform better than their competitors and where there is room for improvement.

APPROACH

Based on state-of-the-art methods in the field of natural language processing (NLP), Insaas and LMU Munich have developed a pipeline that takes an arbitrary amount of review data as input and compresses that information into a customer centricity graph. This approach is targeted at the car insurance industry and at German review texts. The novelty of this work is the combination of several building blocks to produce a multi-step procedure that can be run automatically.

Its main features are pre-trained German language models from the BERT¹-family. BERT is a Transformer²-based language representation model, ie a model that is trained to represent words in a meaningful way, based on their bi-directional context. As pre-training such models requires massive amounts of computational power (typically Tensor Processing Units) and time, it is common practice to use pre-trained versions of such models. The present research used a variant pre-trained on German texts, so had only to fine-tune the model for the task at hand, ie the classification of reviews with respect to topics or polarities.

In what follows, this paper will provide an overview of the various steps of the procedure (Figure 1). In order to make subsequent explanations easier to understand, two exemplary reviews are added. As a first step, aspects (ie topics) are extracted from the reviews. Aspects highlight different facets

8. Applying pre-trained language models for measuring Customer Centricity

Creating a 'customer centricity graph' from unstructured customer feedback

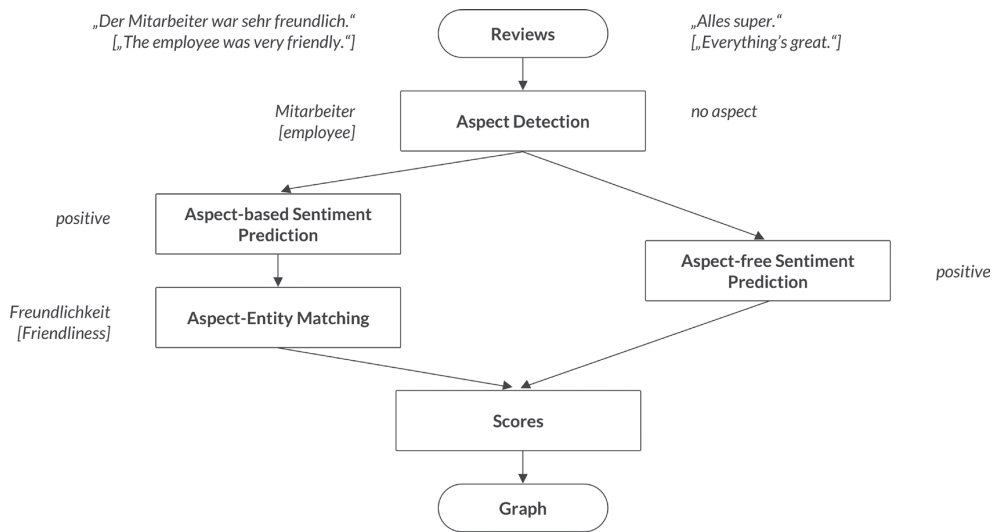


Figure 1: Overview on the steps conducted throughout the pipeline

of a review and may be either explicit or implied; in the latter case, the aspect may be identified from the context. For example, in the case of ‘*Der Mitarbeiter war sehr freundlich*’ [‘The employee was very friendly’], ‘*Mitarbeiter*’ [‘employee’] is the aspect, while in ‘*Alles super*’ [‘Everything’s great’], no aspect can be detected. Depending on the results of this first step, the data may be split into two groups: those reviews *with* identified aspects and those *without*.

Aspect-based sentiment analysis is then performed on the data with aspects, meaning that the sentiment, which is basically the emotion or the polarity, is determined separately for each aspect. In the case of ‘*Mitarbeiter*’ [‘employee’], the context suggests a positive sentiment for this aspect. Second, the aspects must be matched to their corresponding entities. Entities are categories in which the aspects can be grouped to reduce complexity. They are *a priori* defined to be ‘*Beratung*’, ‘*Erreichbarkeit*’, ‘*Freundlichkeit*’, ‘*Kompetenz*’, ‘*Qualität*’, ‘*Problemlösung*’, ‘*Preis*’ and ‘*Leistung*’, which can be translated as ‘guidance’, ‘availability’, ‘friendliness’, ‘expertise’, ‘quality’, ‘problem

solving’, ‘price’ and ‘benefit’. These entities, which can be grouped into either product or service-related, will dominate the definition of the resulting visualisation.

In the example, it makes sense to assign ‘*Mitarbeiter*’ [‘employee’] to *Freundlichkeit* [‘friendliness’], ie entities and sentiments are connected via aspects. For that part of the data without any aspects, a sentiment is predicted for the whole review; this will be called aspect-free sentiment. For the example ‘*Alles super*’ [‘Everything’s great’], this should clearly be positive. All this extracted information is then turned into entity-wise scores by calculating the mean of the sentiments of all aspects belonging to each entity. These are depicted in a radar chart, also known as a spider diagram, with one corner point per entity. For the reviews without any aspects, an aspect-free score is calculated in a similar manner.

THE DATA

The data used for training the different building blocks of the multi-step approach was collected from publicly available web

pages such as comparison portals such as Trustpilot (<https://de.trustpilot.com/>). From the original data set, called the review-wise labelled data, Insaas derived several smaller pieces of data that were targeted for special parts of the pipeline. Short descriptions for these are provided in the following.

Review-wise labelled data

After excluding irrelevant and duplicate reviews, a total of 93,543 samples of data were obtained. Each sample comprised seven variables, namely: feedback, date, source, company, rating, aspect and sentiment. As consumer review text falls under the heading of 'feedback', the present study considers this to be the most important variable. All review texts were written in German and varied in length from just a few words to multiple sentences.

The time stamp in the 'date' variable was initially used solely to identify duplicates. As this paper will discuss, however, including time as an additional dimension in the pipeline can provide an interesting extension to the analysis, hence it was also used to split up reviews by year and thereby create separate graphs.

The variables of 'source' and 'company' are used to indicate the source of the data (ie which comparison portal) and the company being commented upon, respectively. Due to the different sources being used, company names initially differed in spelling and it was necessary to consolidate them in order to group the data by company.

If the source of the review provided star ratings, this information was recorded under the heading of 'rating', on a scale of either 1–5 or 0–10. This information was used by Insaas to correct the predictions of the sentiment model, which predicted the reviews to be 'negative', 'neutral' or 'positive' (stored in the corresponding 'sentiment' column). These sentiments were used as true labels for the aspect-free sentiment prediction.

The 'aspect' variable describes the aspects predicted by the Insaas aspect detection model and serves as the ground truth for aspect prediction.

Aspect-wise labelled data

The second sub data set, which was constructed for the purpose of training one of the building blocks of the pipeline, will be referred to as *aspect-wise labelled data* in order to distinguish it from the *review-wise labelled data*. It was annotated this way because several reviews include more than one aspect. As for aspect-based sentiment classification and aspect-entity matching, the sentiment and the corresponding entity were required for each aspect. This created the need to construct a further data source. Thus, the new labels include aspect-based sentiments and entities for up to three aspects per review. The data comprise a subset of 584 observations of the review-wise labelled data which were manually annotated during the course of the project.

Lemmatisation list

A lemmatisation list was used to efficiently cope with the huge amount of different aspects in the *review-wise labelled data*. Lemmatisation entails grouping inflected words according to their lemma; for example, 'Beiträge' ['insurance premiums'], 'Beitrages' and 'Beitrags' are all assigned to the lemma 'Beitrag' ['insurance premium'].

Initially, there were over 1,000 different aspects. Not only was this too complex for the model to handle, but there was also the problem of multiple aspects referring to the same underlying construct. To address this, the researchers manually created a list where all aspects were assigned to a so-called *lemmatised aspect*. The approach was extended by grouping words with similar meaning to the lemmatised aspects; for example, aspects like 'Vertragsabschluss' ['completion of contract'], 'Vertragsformular'

8. Applying pre-trained language models for measuring Customer Centricity

Creating a 'customer centricity graph' from unstructured customer feedback

['contract form'], 'Unterlagen' ['documents'] and 'Vertragswechsel' ['change of contract'] were allocated to the lemmatised aspect 'Vertrag' ['contract']. While this procedure retained the meaning of the aspects, the generalisation made the task less complex. The lemmatisation list held 197 lemmatised aspects, which were utilised during aspect-based sentiment classification and aspect-entity matching.

Entity synonyms

For the task of aspect-entity matching, a list of synonyms for the entities was created. This list was created using ConceptNet,³ a semantic network that connects potentially related words with one another. For each entity, there were 50–75 synonyms both with respect to meaning, eg 'Hilfsbereitschaft' ['helpfulness'] for 'Freundlichkeit' ['friendliness'], as well as spelling, eg 'Qualität' for 'Qualität'. Also note that some synonyms are not unique for one entity; for example, 'Qualität' ['quality'] can also be used as a synonym for 'Leistung' ['benefit'].

PIPELINE BASED ON BERT MODELS

The goal of the project was to create a code pipeline to transform data from one company (serving as input) into a comprehensive visualisation that can be compared with data from other companies. This paper demonstrates the pipeline using data from Allianz and HUK, and will comprehensively discuss the results of each step inside the pipeline.

A total of 10,680 reviews of Allianz and 11,932 reviews of HUK were obtained.

The first step in the pipeline was aspect detection. This entailed training a classifier for so-called multi-label classification, so that reviews may (potentially) be assigned to more than one label (ie aspect). After removing very rare aspects and applying the lemmatisation list, a list of 198 aspects (including a 'no aspect' label) was obtained.

For this task, the German DistilBERT⁴ model (<https://huggingface.co/distilbert-base-german-cased>) was employed on a subset of the review-wise labelled data. DistilBERT is a smaller version of BERT that was created to address BERT's memory and computational issues. The key technique for reducing the model size is knowledge distillation, which is discussed in depth elsewhere.^{5,6} An in-depth theoretical explanation is beyond the scope of this paper, but the basic idea behind this technique is to train a small(er) *student* model to mimic the predictions of a large(r) *teacher* model.

For the next steps, the data were separated into samples with and without aspects. On the reviews with aspects, aspect-based sentiment classification methods were used to predict one sentiment per aspect. This was necessary as there were reviews like 'Der Mitarbeiter war sehr freundlich, aber der Versicherungsbeitrag zu hoch' ['The employee was very friendly, but the insurance premium was too high'], where the sentiments of 'Mitarbeiter' ['employee'] and 'Versicherungsbeitrag' ['insurance premium'] contradicted each other. LCF-BERT⁷ was selected for this task and trained on the aspect-wise labelled data set, which introduces a local-context-focus (LCF) mechanism. This means that, in order to identify the sentiment of an aspect, an additional focus is set on words that are close to the aspect. The basic BERT model used here was the *bert-base-german-cased* from the huggingface transformers library⁸ (<https://huggingface.co/bert-base-german-cased>). If a review had no predicted sentiment for any aspect, this review was removed from the data set with aspects and added to the one without aspects.

For the task of matching aspects and entities, the list of synonyms for each entity was employed together with FastText⁹ embeddings on aspects, entities and entity synonyms. Subsequently, each aspect was paired with all entities as well as entity synonyms. For each pair of embeddings

(aspect, entity/entity synonym), the cosine similarity was calculated. In order to obtain an entity for each aspect, the researchers took the ten most similar entities or entity synonyms, looked up the entities of the entity synonyms and chose their mode as the final entity for this aspect. If there was no unique mode, the knowledge from the aspect–entity list that had been extracted from the aspect-wise labelled data set was included. If the entity was still undecided, the entity with the highest similarity was added to the list and another attempt to take the mode was taken. Following this process, there remained two aspects that had no unique entity assigned. For these, the number of entity synonyms was reduced until it was possible to calculate a mode. This list of aspects and entities was used in the pipeline.

If no aspects were found within a review text, it was not possible to employ aspect-based methods for sentiment classification. In such instances, a multi-class classifier was used to predict the review’s aspect-free sentiment, thus effectively obtaining the sentiment of the entire review. This was done similarly with aspect detection, but with multi-class instead of multi-label prediction as the target variable contained exactly one label per review. For this task, a German DistilBERT model was fine-tuned on those parts of the review-wise labelled data that were not labelled with any aspects.

To visualise the extracted information, sentiments had to be converted into numbers that could be depicted. The researchers devised multiple scores to deal with special subgroups of reviews and to see which one best showed the results. For reviews without any aspects, aspect-free sentiments were predicted. The absolute values were used to calculate an aspect-free score with the following formula where *review* corresponds to a review without aspects and *N_without_aspects* is the total number of them:

$$aspect_free_score = \frac{1}{N_without_aspects} \sum_{\substack{review \\ without \\ aspects}} sentiment(review)$$

Sentiment (review) ∈ {-1; 0; 1} indicates the sentiment of each review, encoded for negative, neutral and positive, respectively. This score is basically the mean of the sentiments with a lower bound of -1 and an upper bound of 1. For Allianz data, the aspect-free scores are 0.5112 and 0.4504 for the years 2016 and 2020, respectively; for HUK data, they are 0.6780 and 0.4268, respectively. This means that the reviews without aspects were more positive in 2016 than in 2020. Comparing both companies, one may observe that HUK obtained a significantly higher value than Allianz in 2016, but that Allianz scored marginally better in 2020.

For the remaining number of reviews with aspects, the researchers calculated a score for each entity. This formula is actually the same as the one for the aspect-free score, but in this case, one takes into account only those sentiments that correspond to the aspects linked to the respective entity. As the connecting point between sentiments and entities is the aspect, the corresponding aspects may be summed as

$$score(entity) = \frac{1}{N_entity} \sum_{\substack{aspect \\ assigned \\ to\ entity}} sentiment(aspect)$$

where *N_entity* is the number of aspects assigned to the entity and the *sentiment (aspect) ∈ {-1; 0; 1}* is the sentiment belonging to an aspect of this entity. As these scores do not consider the varying values of *N_entity*, it may also be interesting for future work to include weights to account for this issue in a meaningful way.

THE DASHBOARD

For the final visualisation in the Insaas dashboard, data can be filtered by company

8. Applying pre-trained language models for measuring Customer Centricity

Creating a 'customer centricity graph' from unstructured customer feedback

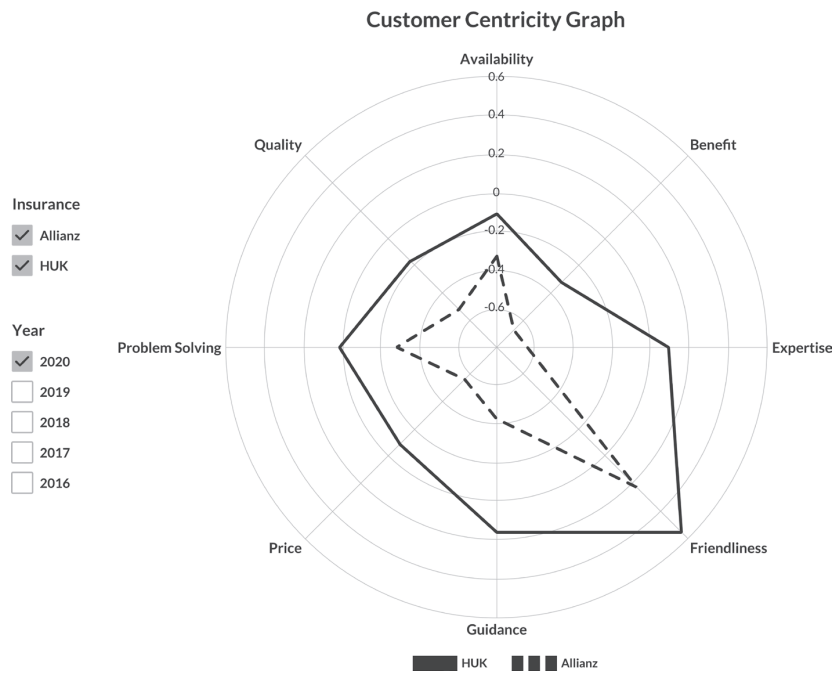


Figure 2: Customer centricity graphs for Allianz and HUK for 2020

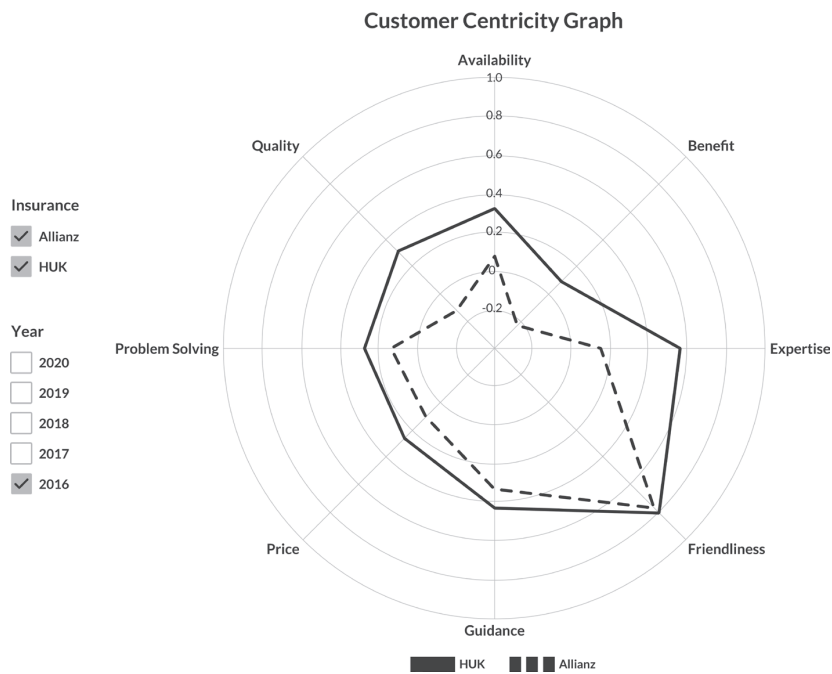


Figure 3: Customer centricity graphs for Allianz and HUK for 2016

and by year. The companies of interest in the present use case were Allianz and HUK. In addition to the entity-wise scores, the dashboard also includes time as a dimension. In Figure 2, which depicts the 2020 customer centricity graphs for Allianz and HUK, one can clearly see that HUK receives higher scores compared with Allianz with respect to all entities. Nevertheless, it is interesting that they both receive the highest values for ‘*Freundlichkeit*’ [‘friendliness’]. This is also the only entity for which Allianz has reached a positive value, unlike HUK, which obtained a positive value for three entities.

By comparison, Figure 3 shows customer centricity graphs for both companies for the year 2016. Already back then, ‘*Freundlichkeit*’ [‘friendliness’] was the highest ranked entity with respect to the sentiment, but besides this, many things appear to be different. Both companies had far better ratings in 2016: while HUK obtained positive sentiment scores for all entities, Allianz did so for all but two. These year-wise scores can be used to evaluate the impact of certain changes, for example in customer service. Note that the scales differ between 2016 and 2020, as the dashboard automatically adjusts its scaling according to obtained scores.

In Figure 4, absolute frequencies of the sentiments per entity (aggregated over all years) showcase yet another visualisation option of the versatile dashboard. The right side shows values for a company of interest (here, Allianz), while on the left, a so-called ‘industry benchmark’, consisting here of HUK and Allianz, serves for comparison. Note that the user can configure the composition of the industry benchmark by checking or unchecking the respective boxes. On the x-axis, the total amounts of the predicted sentiments are displayed, scaled to a similar width in order to allow for better visual comparability. Clearly one can see that on both sides ‘*Kompetenz*’ [‘expertise’] receives the lowest absolute frequency of sentiments whereas ‘*Beratung*’ [‘guidance’] is discussed most frequently. These quantities

must also be taken into account when interpreting the customer centricity graphs in Figures 2 and 3 as they make entity-wise scores more or less reliable.

CONCLUSION

This study has described the development of an automated approach for the analysis and visualisation of customer opinions from feedback texts that employs state-of-the-art methods from the field of natural language processing. Nevertheless, there are still several issues that could be improved. First of all, each of the steps can potentially perform better. In particular, a context-based approach may be applied to take the aspect-entity matching to the next level. Furthermore, the amount of entities, as shown in Figure 4, could be added to the customer centricity graph, for example by adjusting the angles of the entities according to their proportion of all entities. Another way of visualisation could be to take the height as a new dimension of the radar chart. The higher a score is placed in this dimension, the more entities it is based on.

Despite these issues, the researchers have established a working infrastructure for extracting valuable and differentiating information from review data. As the pipeline is built in a modular fashion, its building blocks can be easily modified or improved without the need to change everything else.

With respect to developing this research, future studies could integrate the quality of each review into the scores. Following the hypothesis that reviews with good grammar and spelling show a more fine-grained and reliable opinion, this might obtain interesting new results. This information could be added in the form of weights. Emojis and emoticons are also related to the style of writing. These can be used to further improve the sentiment predictions.

Another idea would be to extend the set of entities, as the given set of eight

8. Applying pre-trained language models for measuring Customer Centricity

Creating a 'customer centricity graph' from unstructured customer feedback

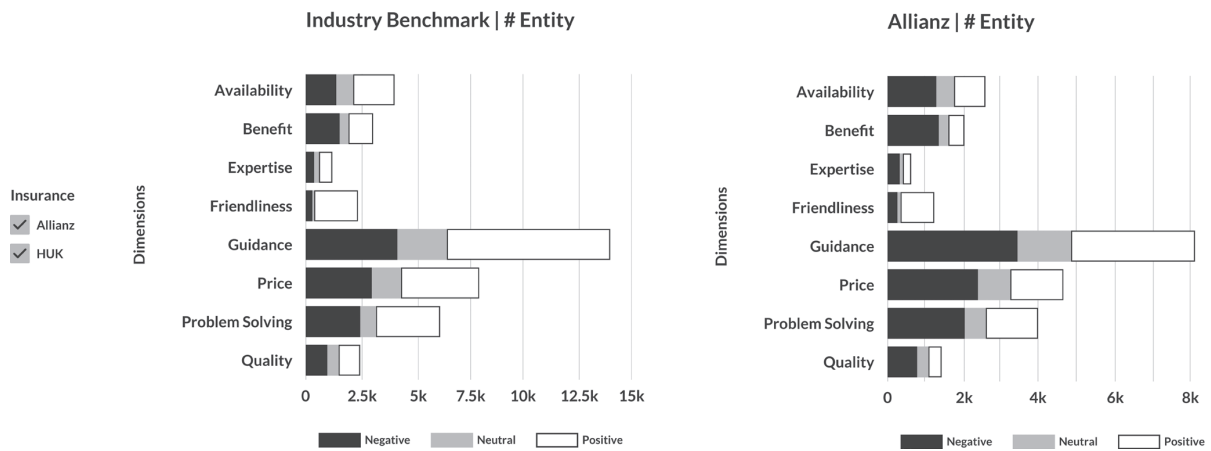


Figure 4: Sentiment frequencies per entity for Allianz data versus an industry benchmark built from HUK and Allianz data; numbers are summed up over all years

entities is not always sufficient to categorise all the various categories that people discuss. As such, it might be of benefit to add new entities, for example from the field of marketing and sales. Generalising these entities may also make the approach applicable to other business sectors.

References

1. Devlin, J., Chang, M. W., Lee, K. and Toutanova, K. (2019) 'BERT: Pre-training of deep bidirectional transformers for language understanding', in 'Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, 2nd–7th June', Vol. 1, pp. 4171–4186.
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017) 'Attention is all you need', *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008.
3. Chen, M., Tian, Y., Yang, M. and Zaniolo, C. (2016) 'Multilingual knowledge graph embeddings for cross-lingual knowledge alignment', arXiv preprint arXiv:1611.03954.
4. Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019) 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter', arXiv preprint arXiv:1910.01108.
5. Bucilua, C., Caruana, R. and Niculescu-Mizil, A. (2006) 'Model compression', in 'Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, 20th–23rd August', pp. 535–541.
6. Hinton, G., Vinyals, O. and Dean, J. (2015) 'Distilling the knowledge in a neural network', arXiv preprint arXiv:1503.02531.
7. Zeng, B., Yang, H., Xu, R., Zhou, W. and Han, X. (2019) 'LCF: A local context focus mechanism for aspect-based sentiment classification', *Applied Sciences*, Vol. 9, No. 16, p. 3389.
8. Wolf, T., Chaumond, J., Debut, L., Sanh, V., 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., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q. and Rush, A. M. (2020) 'Transformers: State-of-the-art natural language processing', in 'Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 16th–20th November', pp. 38–45.
9. Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017) 'Enriching word vectors with subword information', *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 135–146.

9. Re-Evaluating GermEval17 using German pre-trained language models

Chapter 9 is an attempt to estimate the improvements on the task of Aspect-based Sentiment Analysis (ABSA) for German texts induced by the use of pre-trained architectures. The GermEval17 shared task (Wojatzki et al., 2017) was conducted before the broad application of transfer learning architectures, such that the best performance values were also achieved without these more powerful models. Since the huggingface `transformers` module (Wolf et al., 2020) provides us with a multitude of German as well as multilingual models, a comprehensive re-evaluation of this task is made possible. Furthermore, parallels to the improvements in English ABSA due to the introduction of pre-trained models are drawn by considering the developments of SOTA performance on related tasks for the English language (Pontiki et al., 2014).

Contributing article:

Aßenmacher, M., Corvonato, A., and Heumann, C. (2021). Re-Evaluating GermEval17 Using German Pre-Trained Language Models. *Proceedings of the Swiss Text Analytics Conference, Winterthur, Switzerland (Online), June 14-16, 2021*. <http://ceur-ws.org/Vol-2957/paper1.pdf>.

Copyright information:

This article is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>).

Author contributions:

Matthias Aßenmacher drafted the idea of systematically re-evaluating the GermEval17 tasks with newly available German pre-trained language models and comparing the results to developments for the English language. All authors jointly chose a set of German and multilingual models to evaluate during this research project. Alessandra Corvonato was responsible for writing most of the code, continuously supported by Matthias Aßenmacher. The paper was jointly written and reworked by Matthias Aßenmacher and Alessandra Corvonato, with Christian Heumann also contributing valuable input. All three of the authors finally revised and proofread the manuscript. The authors received the [Best Presentation Award](#) at SwissText 2021 for presenting this article.

Supplementary material available at:

- Code: https://github.com/ac74/reevaluating_germeval2017

Re-Evaluating GermEval17 Using German Pre-Trained Language Models

Matthias Aßenmacher¹✉ Alessandra Corvonato¹✉ Christian Heumann¹✉

¹ Department of Statistics, Ludwig-Maximilians-Universität, Munich, Germany

✉{matthias, chris}@stat.uni-muenchen.de, ✉alessandracorvonato@yahoo.de

Abstract

The lack of a commonly used benchmark data set (collection) such as (Super) GLUE (Wang et al., 2018, 2019) for the evaluation of non-English pre-trained language models is a severe shortcoming of current English-centric NLP-research. It concentrates a large part of the research on English, neglecting the uncertainty when transferring conclusions found for the English language to other languages. We evaluate the performance of German and multilingual BERT models currently available via the `huggingface transformers` library on four subtasks of Aspect-based Sentiment Analysis (ABSA) from the GermEval17 workshop. We compare them to pre-BERT architectures (Wojatzki et al., 2017; Schmitt et al., 2018; Attia et al., 2018) as well as to an ELMo-based architecture (Biesialska et al., 2020) and a BERT-based approach (Gühr et al., 2020). The observed improvements are put in relation to those for a similar ABSA task (Pontiki et al., 2014) and similar models (pre-BERT vs. BERT-based) for the English language and we check whether the reported improvements correspond to those we observe for German.

1 Introduction

(Aspect-based) Sentiment Analysis is often used to transform reviews into helpful information on how a product or service of a company is perceived among the customers. Until recently,

Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

Sentiment Analysis was mainly conducted using traditional machine learning and recurrent neural networks, like LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014). Those models have been practically replaced by language models relying on (parts of) the Transformer architecture, a novel framework proposed by Vaswani et al. (2017). Devlin et al. (2019) developed a Transformer-encoder-based language model called BERT (Bidirectional Encoder Representations from Transformers), achieving state-of-the-art (SOTA) performance on several benchmark tasks - mainly for the English language - and becoming a milestone in the field of NLP.

Up to now, only a few researchers have focused on sentiment related problems for German reviews, despite language-specific evaluation is a crucial driving force for a more universal model development and improvement. Unique characteristics of the different languages present different challenges to the models, which is why sole evaluation on English data is a severe shortcoming.

The first shared task on German ABSA, which provides a large annotated data set for training and evaluation, is the *GermEval17 Shared Task* (Wojatzki et al., 2017). The participating teams back then analyzed the data using mostly standard machine learning techniques such as SVMs, CRFs, or LSTMs. In contrast to 2017, today, different pre-trained BERT models are available for a variety of different languages, including German. We re-analyzed the complete GermEval17 Task using seven pre-trained BERT models suitable for German provided by the `huggingface transformers` library (Wolf et al., 2020). We evaluate which one of the models is best suited for the different GermEval17 subtasks by comparing their performance values. Furthermore, we compare our findings on whether (and how much) BERT-based models are able to improve the pre-

BERT SOTA in German ABSA with the SOTA developments for English ABSA by the example of SemEval-2014 (Pontiki et al., 2014).

We first give an overview on the GermEval17 tasks (cf. Sec. 2) and on related work (cf. Sec. 3). Second, we present the data and the models (cf. Sec. 4), while Section 5 holds the results of our re-evaluation. Sections 6 and 7 conclude our work by stating our main findings and drawing parallels to the English language.

2 The GermEval17 Task(s)

The GermEval17 Shared Task (Wojatzki et al., 2017) is a task on analyzing aspect-based sentiments in customer reviews about "Deutsche Bahn" (DB) - the German public train company. The main data was crawled from various social media platforms such as Twitter, Facebook and Q&A websites from May 2015 to June 2016. The documents were manually annotated, and split into a training (**train**), a development (**dev**) and a synchronic (**test_{syn}**) test set. A diachronic test set (**test_{dia}**) was collected the same way from November 2016 to January 2017 in order to test for temporal robustness. The task comprises four subtasks representing a complete classification pipeline. Subtask A is a binary Relevance Classification task which aims at identifying whether the feedback refers to DB. Subtask B aims at classifying the Document-level Polarity ("negative", "positive" and "neutral"). In Subtask C, the model has to identify all the aspect categories with associated sentiment polarities in a relevant document. This multi-label classification task was divided into Subtask C1 (*Aspect-only*) and Subtask C2 (*Aspect+Sentiment*). For this purpose, the organizers defined 20 different aspect categories, e.g. Allgemein (*General*), Sonstige Unregelmäßigkeiten (*Other irregularities*). Finally, Subtask D refers to the Opinion Target Extraction (OTE), i.e. a sequence labeling task extracting the linguistic phrase used to express an opinion. We differentiate between exact match (Subtask D1) and overlapping match, tolerating errors of +/- one token (Subtask D2).

3 Related Work

Already before BERT, many researchers focused on (English) Sentiment Analysis (Behdenna et al., 2018). The most common architectures were traditional machine learning classifiers and recurrent neural networks (RNNs). SemEval14 (Task 4; Pon-

tiki et al., 2014) was the first workshop to introduce Aspect-based Sentiment Analysis (ABSA) which was expanded within SemEval15 Task 12 (Pontiki et al., 2015) and SemEval16 Task 5 (Pontiki et al., 2016). Here, restaurant and laptop reviews were examined on different granularities. The best model at SemEval16 was an SVM/CRF architecture using GloVe embeddings (Pennington et al., 2014). However, many works recently focused on re-evaluating the SemEval Sentiment Analysis task using BERT-based language models (Hoang et al., 2019; Xu et al., 2019; Sun et al., 2019; Li et al., 2019; Karimi et al., 2020; Tao and Fang, 2020).

In comparison, little research deals with German ABSA. For instance, Barriere and Balahur (2020) trained a multilingual BERT model for German Document-level Sentiment Analysis on the SB-10k data set (Cieliebak et al., 2017). Regarding the GermEval17 Subtask B, Guhr et al. (2020) considered both FastText (Bojanowski et al., 2017) and BERT, achieving notable improvements. Biesialska et al. (2020) made use of ensemble models: One is an ensemble of ELMo (Peters et al., 2018), GloVe and a bi-attentive classification network (BCN; McCann et al., 2017), achieving a score of 0.782, and the other one consists of ELMo and a Transformer-based Sentiment Analysis model (TSA), reaching a score of 0.789 for the synchronic test data set. Moreover, Attia et al. (2018) trained a convolutional neural network (CNN), achieving a score of 0.7545 on the synchronic test set. Schmitt et al. (2018) advanced the SOTA for Subtask C by employing biLSTMs and CNNs to carry out end-to-end Aspect-based Sentiment Analysis. The highest score was achieved using an end-to-end CNN architecture with FastText embeddings, scoring 0.523 and 0.557 on the synchronic and diachronic test data set for Subtask C1, respectively, and 0.423 and 0.465 for Subtask C2.

4 Materials and Methods

Data The GermEval17 data is freely available in `.xml`- and `.tsv`-format¹. Each data split (train, validation, test) in `.tsv`-format contains the following variables:

- document id (URL)
- document text
- relevance label (`true`, `false`)

¹The data sets (in both formats) can be obtained from <http://ldata1.informatik.uni-hamburg.de/germeval2017/>.

9. Re-Evaluating GermEval17 using German pre-trained language models

- document-level sentiment label
(negative, neutral, positive)
- aspects with respective polarities
(e.g. Ticketkauf#Haupt:negative)

For documents which are annotated as irrelevant, the sentiment label is set to `neutral` and no aspects are available. Visibly, the `.tsv`-formatted data does not contain the target expressions or their associated sequence positions. Consequently, Subtask D can only be conducted using the data in `.xml`-format, which additionally holds the information on the starting and ending sequence positions of the target phrases.

The data set comprises $\sim 26k$ documents in total, including the diachronic test set with around 1.8k examples. Further, the main data was randomly split by the organizers into a train data set for training, a development data set for validation and a synchronic test data set. Table 1 displays the number of documents for each split.

	train	dev	test _{syn}	test _{dia}
	19,432	2,369	2,566	1,842

Table 1: Number of documents per split of the data set.

While roughly 74% of the documents form the train set, the development split and the synchronic test split contain around 9% and around 10%, respectively. The remaining 7% of the data belong to the diachronic set (cf. Tab. 1). Table 2 shows the relevance distribution per data split. This unveils a pretty skewed distribution of the labels since the relevant documents represent the clear majority with over 80% in each split.

Relevance	train	dev	test _{syn}	test _{dia}
true	16,201	1,931	2,095	1,547
false	3,231	438	471	295

Table 2: Relevance distribution for Subtask A.

The distribution of the sentiments is depicted in Table 3, which shows that between 65% and 69% (per split) belong to the neutral class, 25–31% to the negative and only 4–6% to the positive class.

Table 4 holds the distribution of the 20 different aspect categories assigned to the documents². It

²Multiple annotations per document are possible; for a detailed category description see <https://sites.google.com/view/germeval2017-absa/data>.

shows the number of documents containing certain categories without differentiating between how often a category appears within a given document.

Sentiment	train	dev	test _{syn}	test _{dia}
negative	5,045	589	780	497
neutral	13,208	1,632	1,681	1,237
positive	1,179	148	105	108

Table 3: Sentiment distribution for Subtask B.

The relative distribution of the aspect categories is similar between the splits. On average, there are ~ 1.12 different aspects per document. Again, the label distribution is heavily skewed, with `Allgemein` (*General*) clearly representing the majority class, as it is present in 75.8% of the documents with aspects. The second most frequent category is `Zugfahrt` (*Train ride*) appearing in around 13.8% of the documents. This strong imbalance in the aspect categories leads to an almost Zipfian distribution (Wojatzki et al., 2017).

Category	train	dev	test _{syn}	test _{dia}
Allgemein	11,454	1,391	1,398	1,024
Zugfahrt	1,687	177	241	184
Sonstige Unregelmäßigkeiten	1,277	139	224	164
Atmosphäre	990	128	148	53
Ticketkauf	540	64	95	48
Service und Kundenbetreuung	447	42	63	27
Sicherheit	405	59	84	42
Informationen	306	28	58	35
Connectivity	250	22	36	73
Auslastung und Platzangebot	231	25	35	20
DB App und Website	175	20	28	18
Komfort und Ausstattung	125	18	24	11
Barrierefreiheit	53	14	9	2
Image	42	6	0	3
Toiletten	41	5	7	4
Gastronomisches Angebot	38	2	3	3
Reisen mit Kindern	35	3	7	2
Design	29	3	4	2
Gepäck	12	2	2	6
QR-Code	0	1	1	0
total	18,137	2,149	2,467	1,721
# documents with aspects	16,200	1,930	2,095	1,547
∅ different aspects/document	1.12	1.11	1.18	1.11

Table 4: Aspect category distribution for Subtask C. Multiple mentions of the same aspect category in a document are only considered once.

Pre-trained architectures BERT was initially introduced in a base (110M parameters) and a large (340M) variant, Sanh et al. (2019) proposed an even smaller BERT model (DistilBERT, 60M parameters) trained via knowledge distillation

Model variant	Pre-training corpus	Properties
bert-base-german-cased	12GB of German text (deepset.ai)	L=12, H=768, A=12, 110M parameters
bert-base-german-dbdmz-cased	16GB of German text (dbmdz)	L=12, H=768, A=12, 110M parameters
bert-base-german-dbdmz-uncased	16GB of German text (dbmdz)	L=12, H=768, A=12, 110M parameters
bert-base-multilingual-cased	Largest Wikipedias (top 104 languages)	L=12, H=768, A=12, 179M parameters
bert-base-multilingual-uncased	Largest Wikipedias (top 102 languages)	L=12, H=768, A=12, 168M parameters
distilbert-base-german-cased	16GB of German text (dbmdz)	L=6, H=768, A=12, 66M parameters
distilbert-base-multilingual-cased	Largest Wikipedias (top 104 languages)	L=6, H=768, A=12, 134M parameters

Table 5: Pre-trained models provided by huggingface `transformers` (version 4.0.1) suitable for German. For all available models, see: https://huggingface.co/transformers/pretrained_models.html.

([Hinton et al., 2015](#)). The exact model specifications regarding number of layers (L), number of attention heads (A) and embedding size (H) for available German BERT models are depicted in the last column of Table 5. Both architectures were pre-trained on the Masked Language Modeling task as well as on the auxiliary Next Sentence Prediction task (only BERT) and can subsequently be fine-tuned on a task at hand.

We include three German (Distil)BERT models pre-trained by DBMDZ³ and one by Deepset.ai⁴. The latter one is pre-trained using German Wikipedia (6GB raw text files), the Open Legal Data dump (2.4GB; [Ostendorff et al., 2020](#)) and news articles (3.6GB). DBMDZ combined Wikipedia, EU Bookshop ([Skadiņš et al., 2014](#)), Open Subtitles ([Lison and Tiedemann, 2016](#)), CommonCrawl ([Ortiz Suárez et al., 2019](#)), ParaCrawl ([Esplà-Gomis et al., 2019](#)) and News Crawl ([Haddow, 2018](#)) to a corpus with a total size of 16GB with $\sim 2,350$ M tokens. Besides this, we use the three multilingual (Distil)BERT models included in the `transformers` module. This amounts to five BERT and two DistilBERT models, two of which are "uncased" (i.e. every character is lower-cased) while the other five models are "cased" ones.

5 Results

For the re-evaluation, we used the latest data provided in .xml-format. Duplicates were not removed, in order to make our results as comparable as possible. We tokenized the documents and fixed single spelling mistakes in the labels⁵. For Subtask D, the BIO-tags were added based on the provided

³MDZ Digital Library team at the Bavarian State Library. Visit <https://www.digitale-sammlungen.de> for details and <https://github.com/dbmdz/berts> for their repository on pre-trained BERT models.

⁴Visit <https://deepset.ai/german-bert> for details.

⁵"positive" in `train` set was replaced with "positive", "negative" in `testdia` set was replaced with "negative".

sequence positions, i.e. one entity corresponds to at least one token tag starting with B- for "Beginning" and continuing with I- for "Inner". If a token does not belong to any entity, the tag O for "Outer" is assigned. For instance, the sequence "fährt nicht" (engl. "does not run") consists of two tokens and would receive the entity `Zugfahrt:negative` and the token tags `[B-Zugfahrt:negative, I-Zugfahrt:negative]` if it refers to a DB train which is not running.

The models were fine-tuned on one Tesla V100 PCIe 16GB GPU using Python 3.8.7. Moreover, the `transformers` module (version 4.0.1) and `torch` (version 1.7.1) were used⁶. The considered values for the hyperparameters for fine-tuning follow the recommendations of [Devlin et al. \(2019\)](#):

- Batch size $\in \{16, 32\}$,
- Adam learning rate $\in \{5e, 3e, 2e\} - 5$,
- # epochs $\in \{2, 3, 4\}$.

After evaluating the model performance for combinations⁷ of the different hyperparameters, all pre-trained architectures were fine-tuned with a learning rate of 5e-5 for four epochs, which turned out to be the most promising combination across the different models. The maximum sequence length was set to 256, which is sufficient since the evaluated data set consists of rather short texts from social media, and a batch size of 32 was chosen.

Other models Eight teams officially participated in the GermEval17 shared task, five of which analyzed Subtask A, all of them Subtask B and two respectively Subtask C and D. We furthermore consider the system by [Ruppert et al. \(2017\)](#) additionally to the participants' models from 2017, even

⁶Source code is available on GitHub: https://github.com/ac74/reevaluating_germeval2017. The results are fully reproducible for Subtasks A, B and C. For Subtask D, reproducibility could not be ensured. The micro F1 scores fluctuate across different runs between ± 0.01 around the reported values.

⁷Due to memory limitations, not every hyperparameter combination was applicable.

9. Re-Evaluating GermEval17 using German pre-trained language models

though they were the organizers and did not "officially" participate. They also tackled all four sub-tasks. Since 2017 several other authors analyzed (parts of) the GermEval17 subtasks using more advanced models, which we also consider for comparison here. Table 6 shows which authors employed which kinds of models to solve which task.

Subtask	A	B	C1	C2	D1	D2
Models from 2017 (Wojatzki et al., 2017; Ruppert et al., 2017)	X	X	X	X	X	X
Our BERT models	X	X	X	X	X	X
CNN (Attia et al., 2018)	-	X	-	-	-	-
CNN+FastText (Schmitt et al., 2018)	-	-	X	X	-	-
ELMo+GloVe+BCN (Biesialska et al., 2020)	-	X	-	-	-	-
ELMo+TSA (Biesialska et al., 2020)	-	X	-	-	-	-
FastText (Guhr et al., 2020)	-	X	-	-	-	-
bert-base-german-cased (Guhr et al., 2020)	-	X	-	-	-	-

Table 6: An overview on all the models discussed in this article, an "X" in a column indicates that the architecture was evaluated on the respective subtask.

Subtask A The Relevance Classification is a binary document classification task with classes `true` and `false`. Table 7 displays the micro F1 score obtained by each language model on each test set (best result per data set in bold).

Language model	test _{syn}	test _{dia}
Best model 2017 (Sayyed et al., 2017)	0.903	0.906
bert-base-german-cased	0.950	0.939
bert-base-german-dbdmz-cased	0.951	0.946
bert-base-german-dbdmz-uncased	0.957	0.948
bert-base-multilingual-cased	0.942	0.933
bert-base-multilingual-uncased	0.944	0.939
distilbert-base-german-cased	0.944	0.939
distilbert-base-multilingual-cased	0.941	0.932

Table 7: F1 scores for Subtask A on synchronic and diachronic test sets.

All the models outperform the best result achieved in 2017 for both test data sets. For the synchronic test set, the previous best result is surpassed by 3.8–5.4 percentage points. For the diachronic test set, the absolute difference to the best contender of 2017 varies between 2.6 and 4.2 percentage points. With a micro F1 score of 0.957 and 0.948, respectively, the best scoring pre-trained language model is the uncased German BERT-BASE variant by dbdmz, followed by its cased version. All the pre-trained models perform slightly better on the synchronic test data than on the diachronic data. Attia et al. (2018), Schmitt et al. (2018), Biesialska et al. (2020) and Guhr et al. (2020) did not evaluate their models on this task.

Subtask B Subtask B refers to the Document-level Polarity, which is a multi-class classification task with three classes. Table 8 demonstrates the performances on the two test sets:

Language model	test _{syn}	test _{dia}
Best models 2017 (test _{syn} : Ruppert et al., 2017) (test _{dia} : Sayyed et al., 2017)	0.767	0.750
bert-base-german-cased	0.798	0.793
bert-base-german-dbdmz-cased	0.799	0.785
bert-base-german-dbdmz-uncased	0.807	0.800
bert-base-multilingual-cased	0.790	0.780
bert-base-multilingual-uncased	0.784	0.766
distilbert-base-german-cased	0.798	0.776
distilbert-base-multilingual-cased	0.777	0.770
CNN (Attia et al., 2018)	0.755	-
ELMo+GloVe+BCN (Biesialska et al., 2020)	0.782	-
ELMo+TSA (Biesialska et al., 2020)	0.789	-
FastText (Guhr et al., 2020)	0.698 [†]	-
bert-base-german-cased (Guhr et al., 2020)	0.789 [†]	-

Table 8: Micro-averaged F1 scores for Subtask B on synchronic and diachronic test sets.

[†]Guhr et al. (2020) created their own (balanced & unbalanced) data splits, which limits comparability. We compare to the performance on the unbalanced data since it more likely resembles the original data splits.

All models outperform the best model from 2017 by 1.0–4.0 percentage points for the synchronic, and by 1.6–5.0 percentage points for the diachronic test set. On the synchronic test set, the uncased German BERT-BASE model by dbdmz performs best with a score of 0.807, followed by its cased variant with 0.799. For the diachronic test set, the uncased German BERT-BASE model exceeds the other models with a score of 0.800, followed by the cased German BERT-BASE model reaching a score of 0.793. The three multilingual models perform generally worse than the German models on this task. Besides this, all the models perform slightly better on the synchronic data set than on the diachronic one. The FastText-based model (Guhr et al., 2020) comes not even close to the baseline from 2017, while the ELMo-based models (Biesialska et al., 2020) are pretty competitive. Interestingly, two of the multilingual models are even outperformed by these ELMo-based models.

Subtask C Subtask C is split into *Aspect-only* (Subtask C1) and *Aspect+Sentiment* Classification (Subtask C2), each being a multi-label classification task⁸. As the organizers provide 20 aspect categories, Subtask C1 includes 20 labels, whereas Subtask C2 has 60 labels since each aspect category

⁸This leads to a change of activation functions in the final layer from softmax to sigmoid + binary cross entropy loss.

can be combined with each of the three sentiments. Consistent with Lee et al. (2017) and Mishra et al. (2017), we do not account for multiple mentions of the same label in one document. The results for Subtask C1 are shown in Table 9:

Language model	test _{syn}	test _{dia}
Best model 2017 (Ruppert et al., 2017)	0.537	0.556
bert-base-german-cased	0.756	0.762
bert-base-german-dbdmz-cased	0.756	0.781
bert-base-german-dbdmz-uncased	0.761	0.791
bert-base-multilingual-cased	0.706	0.734
bert-base-multilingual-uncased	0.723	0.752
distilbert-base-german-cased	0.738	0.768
distilbert-base-multilingual-cased	0.716	0.744
CNN+FastText (Schmitt et al., 2018)	0.523	0.557

Table 9: Micro-averaged F1 scores for Subtask C1 (*Aspect-only*) on synchronic and diachronic test sets. A detailed overview of *per-class* performances for error analysis can be found in Table 15 in Appendix A.

All pre-trained German BERTs clearly surpass the best performance from 2017 as well as the results reported by Schmitt et al. (2018), who are the only ones of the other authors to evaluate their models on this tasks. Regarding the synchronic test set, the absolute improvement ranges between 16.9 and 22.4 percentage points, while for the diachronic test data, the models outperform the previous results by 17.8–23.5 percentage points. The best model is again the uncased German BERT-BASE model by dbdmz, reaching scores of 0.761 and 0.791, respectively, followed by the two cased German BERT-BASE models. One more time, the multilingual models exhibit the poorest performances amongst the evaluated models. Next, Table 10 shows the results for Subtask C2:

Language model	test _{syn}	test _{dia}
Best model 2017 (Ruppert et al., 2017)	0.396	0.424
bert-base-german-cased	0.634	0.663
bert-base-german-dbdmz-cased	0.628	0.663
bert-base-german-dbdmz-uncased	0.655	0.689
bert-base-multilingual-cased	0.571	0.634
bert-base-multilingual-uncased	0.553	0.631
distilbert-base-german-cased	0.629	0.663
distilbert-base-multilingual-cased	0.589	0.642
CNN+FastText (Schmitt et al., 2018)	0.423	0.465

Table 10: Micro-averaged F1 scores for Subtask C2 (*Aspect+Sentiment*) on synchronic and diachronic test sets. A detailed overview of *per-class* performances for error analysis can be found in Table 16 in Appendix A.

Here, the pre-trained models surpass the best model from 2017 by 15.7–25.9 percentage points and 20.7–26.5 percentage points, respectively, for the

synchronic and diachronic test sets. Again, the best model is the uncased German BERT-BASE dbdmz model reaching scores of 0.655 and 0.689, respectively. The CNN models (Schmitt et al., 2018) are also outperformed. For both, Subtask C1 and C2, all the displayed models perform better on the diachronic than on the synchronic test data.

Subtask D Subtask D refers to the Opinion Target Extraction (OTE) and is thus a token-level classification task. As this is a rather difficult task, Wojatzki et al. (2017) distinguish between exact (Subtask D1) and overlapping match (Subtask D2), tolerating a deviation of \pm one token. Here, "entities" are identified by their BIO-tags. It is noteworthy that there are less entities here than for Subtask C since document-level aspects or sentiments could not always be assigned to a certain sequence in the document. As a result, there are less documents at disposal for this task, namely 9,193. The remaining data has 1.86 opinions per document on average. The majority class is now *Sonstige Unregelmäßigkeiten:negative* with around 15.4% of the true entities (16,650 in total), leading to more balanced data than in Subtask C.

	Language model	test _{syn}	test _{dia}
	Best model 2017 (Ruppert et al., 2017)	0.229	0.301
without CRF	bert-base-german-cased	0.460	0.455
	bert-base-german-dbdmz-cased	0.480	0.466
	bert-base-german-dbdmz-uncased	0.492	0.501
	bert-base-multilingual-cased	0.447	0.457
	bert-base-multilingual-uncased	0.429	0.404
	distilbert-base-german-cased	0.347	0.357
with CRF	distilbert-base-multilingual-cased	0.430	0.419
	bert-base-german-cased	0.446	0.443
	bert-base-german-dbdmz-cased	0.466	0.444
	bert-base-german-dbdmz-uncased	0.515	0.518
	bert-base-multilingual-cased	0.472	0.466
	bert-base-multilingual-uncased	0.477	0.452
	distilbert-base-german-cased	0.424	0.403
	distilbert-base-multilingual-cased	0.436	0.418

Table 11: Entity-level micro-averaged F1 scores for Subtask D1 (*exact match*) on synchronic and diachronic test sets. A detailed overview of *per-class* performances for error analysis can be found in Table 17 in Appendix B.

In Table 11, we compare the pre-trained models using an "ordinary" softmax layer to when using a CRF layer for Subtask D1.

The best performing model is the uncased German BERT-BASE model by dbdmz with CRF layer on both test sets, with a score of 0.515 and 0.518, respectively. Overall, the results from 2017 are outperformed by 11.8–28.6 percentage points

9. Re-Evaluating GermEval17 using German pre-trained language models

Language model		test _{syn}	test _{dia}	
Best models 2017 (test _{syn} : Lee et al., 2017) (test _{dia} : Ruppert et al., 2017)		0.348	0.365	
without CRF	bert-base-german-cased	0.471	0.474	
	bert-base-german-dbmdz-cased	0.491	0.488	
	bert-base-german-dbmdz-uncased	0.501	0.518	
	bert-base-multilingual-cased	0.457	0.473	
	bert-base-multilingual-uncased	0.435	0.417	
	distilbert-base-german-cased	0.397	0.407	
	distilbert-base-multilingual-cased	0.433	0.429	
with CRF	bert-base-german-cased	0.455	0.457	
	bert-base-german-dbmdz-cased	0.476	0.469	
	bert-base-german-dbmdz-uncased	0.523	0.533	
	bert-base-multilingual-cased	0.476	0.474	
	bert-base-multilingual-uncased	0.484	0.464	
	distilbert-base-german-cased	0.433	0.423	
		distilbert-base-multilingual-cased	0.442	0.427

Table 12: Entity-level micro-averaged F1 scores for Subtask D2 (*overlapping match*) on synchronic and diachronic test sets. A detailed overview of *per-class* performances for error analysis can be found in Table 18 in Appendix B.

on the synchronic test set and 5.6–21.7 percentage points on the diachronic test set.

For the overlapping match (cf. Tab. 12), the best system from 2017 are outperformed by 4.9–17.5 percentage points on the synchronic and by 4.2–16.8 percentage points on the diachronic test set. Again, the uncased German BERT-BASE model by dbmdz with CRF layer performs best with an micro F1 score of 0.523 on the synchronic and 0.533 on the diachronic set. To our knowledge, there were no other models to compare our performance values with, besides the results from 2017.

Main Takeaways For the first two subtasks, which are rather simple binary and multi-class classification tasks, the pre-trained models are able to improve a little upon the already pretty decent performance values from 2017. Further, we do not see large differences between the different pre-trained models. Nevertheless, the small differences we can observe, already point in the same direction as what can be observed for the primary ABSA tasks of interest, C1 and C2:

- Uncased models have a tendency of outperforming their cased counterparts for the monolingual models, for multilingual models this cannot be clearly confirmed.
- Monolingual models outperform the multilingual ones.
- There are no large performance differences between the two cased BERT models by DBMDZ and Deepset.ai, which suggests only a minor influence of the different corpora, which the models were pre-trained on.

- The monolingual DistilBERT model is pretty competitive, it consistently outperforms its multilingual counterpart as well as the multilingual BERT models on the subtasks A – C and is at least competitive to the monolingual BERT models.

For D1 and D2 we observe a rather clear dominance of the uncased monolingual model which is not observable to this extent for the other tasks.

6 Discussion

After having observed a notable performance increase for German ABSA when employing pre-trained models, the next step is to compare these observations to what was reported for the English language. Therefore, we examine the temporal development of the SOTA performance on the most widely adopted data sets for English ABSA, originating from the SemEval Shared Tasks (Pontiki et al., 2014, 2015, 2016). When looking at public leaderboards, e.g. <https://paperswithcode.com/>, Subtask SB2 (*aspect term polarity*) from SemEval-2014 is the task which attracts most of the researchers. This task is related, but not perfectly similar, to Subtask C2, since in this case, the *aspect term* is always a word which has to present in the given review. For this task, a comparison of pre-BERT and BERT-based methods reveals no big “jump” in the performance values, but rather a steady increase over time (cf. Tab. 13).

Language model		Laptops	Restaurants
pre-BERT	Best model SemEval-2014 (Pontiki et al., 2014)	0.7048	0.8095
	MemNet (Tang et al., 2016)	0.7221	0.8095
	HAPN (Li et al., 2018)	0.7727	0.8223
BERT-based	BERT-SPC (Song et al., 2019)	0.7899	0.8446
	BERT-ADA (Rietzler et al., 2020)	0.8023	0.8789
	LCF-ATEPC (Yang et al., 2019)	0.8229	0.9018

Table 13: Development of the SOTA Accuracy for the aspect term polarity task (SemEval-2014; Pontiki et al., 2014). Selected models were picked from <https://paperswithcode.com/sota/aspect-based-sentiment-analysis-on-semeval>.

Clearly more related, but unfortunately also less used, are the subtasks SB3 (*aspect category extraction*; comparable to Subtask C1) and SB4 (*aspect category polarity*; comparable to Subtask C2)

from SemEval-2014.⁹ Limitations with respect to comparability arise from the different numbers of categories: Subtask SB4 only exhibits five aspect categories (as opposed to 20 categories for GermEval17) which leads to an easier classification problem and is reflected in the already pretty high scores of the 2014 baselines. Table 14 shows the performance of the best model from 2014 as well as performance of subsequent (pre-BERT and BERT-based) models for subtasks SB3 and SB4.

Language model		Restaurants	
		SB3	SB4
pre-BERT	Best model SemEval-2014 (Pontiki et al., 2014)	0.8857	0.8292
	ATAE-LSTM (Wang et al., 2016)	—	0.840
BERT-based	BERT-pair (Sun et al., 2019)	0.9218	0.899
	CG-BERT (Wu and Ong, 2020)	0.9162 [†]	0.901 [†]
	QACG-BERT (Wu and Ong, 2020)	0.9264	0.904 [†]

Table 14: Development of the SOTA F1 score (SB3) and Accuracy (SB4) for the aspect category extraction/polarity task (SemEval-2014; Pontiki et al., 2014). [†]Additional auxiliary sentences were used.

In contrast to what can be observed for SB2, in this case, the performance increase on SB4 caused by the introduction of BERT seems to be kind of striking. While the ATAE-LSTM (Wang et al., 2016) only slightly increased the performance compared to 2014, the BERT-based models led to a jump of more than 6 percentage points. So when taking into account the potential room for improvement (0.16 for SB4 vs. 0.60 for C2), the improvements *relative* to the potential (0.06/0.16 for SB4 vs. 0.23/0.60 for C2) are quite similar.

Another issue is that (partly) highly specialized (T)ABSA architectures were used for improving the SOTA on the SemEval-2014 tasks, while we “only” applied standard pre-trained German BERT models without any task-specific modifications or extensions. This leaves room for further improvements on this task on German data which should be an objective for future research.

⁹Since the data sets (*Restaurants* and *Laptops*) have been further developed for SemEval-2015 and SemEval-2016, subtasks SB3 and SB4 are revisited under the names Slot 1 and Slot 3 for the in-domain ABSA in SemEval-2015. Slot 2 from SemEval-2015 aims at OTE and thus corresponds to Subtask D from GermEval17. For SemEval-2016 the same task names as in 2015 were used, subdivided into Subtask 1 (*sentence-level ABSA*) and Subtask 2 (*text-level ABSA*).

7 Conclusion

As one would have hoped, all the state-of-the-art pre-trained language models clearly outperform all the models from 2017, proving the power of transfer learning also for German ABSA. Throughout the presented analyses, the models always achieve similar results between the synchronic and the diachronic test sets, indicating temporal robustness for the models. Nonetheless, the diachronic data was collected *only* half a year after the main data. It would be interesting to see whether the trained models would return similar predictions on data collected a couple of years later.

The uncased German BERT-BASE model by dbmdz achieves the best results across all subtasks. Since Rönqvist et al. (2019) showed that monolingual BERT models often outperform the multilingual models for a variety of tasks, one might have already suspected that a monolingual German BERT performs best across the performed tasks. It may not seem evident at first that an uncased language model ends up as the best performing model since, e.g. in Sentiment Analysis, capitalized letters might be an indicator for polarity. In addition, since nouns and beginnings of sentences always start with a capital letter in German, one might assume that lower-casing the whole text changes the meaning of some words and thus confuses the language model. Nevertheless, the GermEval17 documents are very noisy since they were retrieved from social media. That means that the data contains many misspellings, grammar and expression mistakes, dialect, and colloquial language. For this reason, already some participating teams in 2017 pursued an elaborate pre-processing on the text data in order to eliminate some noise (Hövelmann and Friedrich, 2017; Sayyed et al., 2017; Sidarenka, 2017). Among other things, Hövelmann and Friedrich (2017) transformed the text to lower-case and replaced, for example, “S-Bahn” and “S Bahn” with “sbahn”. We suppose that in this case, lower-casing the texts improves the data quality by eliminating some of the noise and acts as a sort of regularization. As a result, the uncased models potentially generalize better than the cased models. The findings from Mayhew et al. (2019), who compare cased and uncased pre-trained models on social media data for NER, corroborate this hypothesis.

References

- Mohammed Attia, Younes Samih, Ali Elkahky, and Laura Kallmeyer. 2018. [Multilingual multi-class sentiment classification using convolutional neural networks](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Valentin Barriere and Alexandra Balahur. 2020. [Improving sentiment analysis over non-English tweets using multilingual transformers and automatic translation for data-augmentation](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 266–271, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Salima Behdenna, Fatiha Barigou, and Ghalem Belem. 2018. [Document level sentiment analysis: A survey](#). *EAI Endorsed Transactions on Context-aware Systems and Applications*, 4:154339.
- Katarzyna Biesialska, Magdalena Biesialska, and Henryk Rybinski. 2020. [Sentiment analysis with contextual embeddings and self-attention](#). *arXiv preprint arXiv:2003.05574*.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. [Learning phrase representations using rnn encoder-decoder for statistical machine translation](#). *arXiv preprint arXiv:1406.1078*.
- Mark Cieliebak, Jan Milan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. [A Twitter corpus and benchmark resources for German sentiment analysis](#). In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 45–51, Valencia, Spain. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- M. Esplà-Gomis, M. Forcada, Gema Ramírez-Sánchez, and Hieu T. Hoang. 2019. [ParaCrawl: Web-scale parallel corpora for the languages of the EU](#). In *MT-Summit*.
- Oliver Guhr, Anne-Kathrin Schumann, Frank Bahrmann, and Hans-Joachim Böhme. 2020. [Training a Broad-Coverage German Sentiment Classification Model for Dialog Systems](#). In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 1627–1632, Marseille, France.
- Barry Haddow. 2018. [News Crawl Corpus](#).
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *arXiv preprint arXiv:1503.02531*.
- Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouses. 2019. [Aspect-based sentiment analysis using BERT](#). In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, Turku, Finland. Linköping University Electronic Press.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural computation*, 9(8):1735–1780.
- Leonard Hövelmann and Christoph M. Friedrich. 2017. [Fasttext and Gradient Boosted Trees at GermEval-2017 Tasks on Relevance Classification and Document-level Polarity](#). In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.
- Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2020. [Adversarial training for aspect-based sentiment analysis with bert](#).
- Ji-Ung Lee, Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. [UKP TU-DA at GermEval 2017: Deep Learning for Aspect Based Sentiment Detection](#). In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.
- Lishuang Li, Yang Liu, and AnQiao Zhou. 2018. [Hierarchical attention based position-aware network for aspect-level sentiment analysis](#). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 181–189, Brussels, Belgium. Association for Computational Linguistics.
- Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. [Exploiting BERT for end-to-end aspect-based sentiment analysis](#). In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41, Hong Kong, China. Association for Computational Linguistics.
- Pierre Lison and Jörg Tiedemann. 2016. [OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles](#). In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.
- Stephen Mayhew, Tatiana Tsygankova, and Dan Roth. 2019. [ner and pos when nothing is capitalized](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

- 9th International Joint Conference on Natural Language Processing, pages 6256–6261, Hong Kong, China. Association for Computational Linguistics.
- Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. **Learned in translation: Contextualized word vectors.** In *Advances in Neural Information Processing Systems*, volume 30, pages 6294–6305. Curran Associates, Inc.
- Pruthwik Mishra, Vandan Mujadia, and Soujanya Lanka. 2017. GermEval 2017: Sequence based Models for Customer Feedback Analysis. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.
- Pedro Javier Ortíz Suárez, Benoît Sagot, and Laurent Romary. 2019. **Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures.** In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom. Leibniz-Institut für Deutsche Sprache.
- Malte Ostendorff, Till Blume, and Saskia Ostendorff. 2020. **Towards an Open Platform for Legal Information.** In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, JCDL '20*, pages 385–388, New York, NY, USA. Association for Computing Machinery.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee de clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeny Kotelnikov, Nuria Bel, Salud Maria Zafra, and Gülşen Eryiğit. 2016. **Semeval-2016 task 5: Aspect based sentiment analysis.** In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. **SemEval-2015 task 12: Aspect based sentiment analysis.** In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado. Association for Computational Linguistics.
- Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. **SemEval-2014 task 4: Aspect based sentiment analysis.** In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.
- Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2020. **Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification.** In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4933–4941, Marseille, France. European Language Resources Association.
- Samuel Rönnqvist, Jenna Kanerva, Tapio Salakoski, and Filip Ginter. 2019. Is Multilingual BERT Fluent in Language Generation? In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 29–36, Turku, Finland. Linköping University Electronic Press.
- Eugen Ruppert, Abhishek Kumar, and Chris Biemann. 2017. LT-ABSA: An Extensible Open-Source System for Document-Level and Aspect-Based Sentiment Analysis. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Zeeshan Ali Sayyed, Daniel Dakota, and Sandra Kübler. 2017. IDS-IUCL: Investigating Feature Selection and Oversampling for GermEval 2017. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.
- Martin Schmitt, Simon Steinheber, Konrad Schreiber, and Benjamin Roth. 2018. **Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks.** In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1114, Brussels, Belgium. Association for Computational Linguistics.
- Uladzimir Sidarenka. 2017. PotTS at GermEval-2017 Task B: Document-Level Polarity Detection Using Hand-Crafted SVM and Deep Bidirectional LSTM Network. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.
- Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of Parallel Words for Free: Building and Using the EU Bookshop Corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 1850–1855, Reykjavik, Iceland. European Language Resources Association (ELRA).

- Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.
- Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *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)*, pages 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.
- Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.
- Jie Tao and Xing Fang. 2020. Toward multi-label sentiment analysis: a transfer learning based approach. *Journal of Big Data*, 7:1.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, California, USA.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in neural information processing systems*, pages 3266–3280.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.
- Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.
- Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. *GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12, Berlin, Germany.
- 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. *Transformers: State-of-the-Art Natural Language Processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zhengxuan Wu and Desmond C Ong. 2020. Context-guided bert for targeted aspect-based sentiment analysis. *arXiv preprint arXiv:2010.07523*.
- Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis.
- Heng Yang, Biqing Zeng, JianHao Yang, Youwei Song, and Ruyang Xu. 2019. A multi-task learning model for chinese-oriented aspect polarity classification and aspect term extraction. *arXiv preprint arXiv:1912.07976*.

Appendix

A Detailed results (per category) for Subtask C

It may be interesting to have a more detailed look at the model performance for this subtask because of the high number of classes and their skewed distribution by investigating the performance on category-level. Table 15 shows the performance of the uncased German BERT-BASE model by dbmdz per test set for Subtask C1. The support indicates the number of appearances, which are also displayed in Table 4 in this case. Seven categories are summarized in *Rest* because they have an F1 score of 0 for both test sets, i.e. the model is not able to correctly identify any of these seven aspects appearing in the test data. The table is sorted by the score on the synchronic test set.

Aspect Category	test _{syn}		test _{dia}	
	Score	Support	Score	Support
Allgemein	0.854	1,398	0.877	1,024
Sonstige Unregelmäßigkeiten	0.782	224	0.785	164
Connectivity	0.750	36	0.838	73
Zugfahrt	0.678	241	0.687	184
Auslastung und Platzangebot	0.645	35	0.667	20
Sicherheit	0.602	84	0.639	42
Atmosphäre	0.600	148	0.532	53
Barrierefreiheit	0.500	9	0	2
Ticketkauf	0.481	95	0.506	48
Service und Kundenbetreuung	0.476	63	0.417	27
DB App und Website	0.455	28	0.563	18
Informationen	0.329	58	0.464	35
Komfort und Ausstattung	0.286	24	0	11
<i>Rest</i>	0	24	0	20

Table 15: Micro-averaged F1 scores and support by aspect category (Subtask C1). Seven categories are summarized in *Rest* and show each a score of 0.

The F1 scores for Allgemein (*General*), Sonstige Unregelmäßigkeiten (*Other ir-*

regularities) and `Connectivity` are the highest. 13 categories, mostly similar between the two test sets, show a positive F1 score on at least one of the two test sets. For the categories subsumed under `Rest`, the model was not able to learn how to correctly identify these categories.

Subtask C2 exhibits a similar distribution of the true labels, with the `Aspect+Sentiment` category `Allgemein:neutral` as majority class. Over 50% of the true labels belong to this class. Table 16 shows that only 12 out of 60 labels can be detected by the model (see Table 16).

Aspect+Sentiment Category	test _{syn}		test _{dia}	
	Score	Support	Score	Support
Allgemein:neutral	0.804	1,108	0.832	913
Sonstige Unregelmäßigkeiten:negative	0.782	221	0.793	159
Zugfahrt:negative	0.645	197	0.725	149
Sicherheit:negative	0.640	78	0.585	39
Allgemein:negative	0.582	258	0.333	80
Atmosphäre:negative	0.569	126	0.447	39
Connectivity:negative	0.400	20	0.291	46
Ticketkauf:negative	0.364	42	0.298	34
Auslastung und Platzangebot:negative	0.350	31	0.211	17
Allgemein:positive	0.214	41	0.690	33
Zugfahrt:positive	0.154	34	0	34
Service und Kundenbetreuung:negative	0.146	36	0.174	21
<i>Rest</i>	0	343	0	180

Table 16: Micro-averaged F1 scores and support by `Aspect+Sentiment` category (Subtask C2). 48 categories are summarized in `Rest` and show each a score of 0.

All the aspect categories displayed in Table 16 are also visible in Table 15 and most of them have negative sentiment. `Allgemein:neutral` and `Sonstige Unregelmäßigkeiten:negative` show the highest scores. Again, we assume that here, 48 categories could not be identified due to data sparsity. However, having this in mind, the model achieves a relatively high overall performance for both, Subtask C1 and C2 (cf. Tab. 9 and Tab. 10). This is mainly owed to the high score of the majority classes `Allgemein` and `Allgemein:neutral`, respectively, because the micro F1 score puts a lot of weight on majority classes. It might be interesting whether the classification of the rare categories can be improved by balancing the data. We experimented with removing general categories such as `Allgemein`, `Allgemein:neutral` or documents with sentiment `neutral` since these are usually less interesting for a company. We observe a large drop in the overall F1 score which is attributed to the absence of the strong majority class and the resulting data loss. Indeed, the classification for some single categories could be improved, but the

rare categories could still not be identified by the model.

B Detailed results (per category) for Subtask D

Similar as for Subtask C, the results for the best model are investigated in more detail. Table 17 gives the detailed classification report for the uncased German BERT-BASE model with CRF layer on Subtask D1. Only entities that were correctly detected at least once are displayed. The table is sorted by the score on the synchronic test set. The classification report for Subtask D2 is displayed analogously in Table 18.

Category	test _{syn}		test _{dia}	
	Score	Support	Score	Support
Zugfahrt:negative	0.702	622	0.729	495
Sonstige Unregelmäßigkeiten:negative	0.681	693	0.581	484
Sicherheit:negative	0.604	337	0.457	122
Connectivity:negative	0.598	56	0.620	109
Barrierefreiheit:negative	0.595	14	0	3
Auslastung und Platzangebot:negative	0.579	66	0.447	31
Connectivity:positive	0.571	26	0.555	60
Allgemein:negative	0.545	807	0.343	139
Atmosphäre:negative	0.500	403	0.337	164
Ticketkauf:negative	0.383	96	0.583	74
Ticketkauf:positive	0.368	59	0	13
Komfort und Ausstattung:negative	0.357	24	0	16
Atmosphäre:neutral	0.348	40	0.111	14
Service und Kundenbetreuung:negative	0.323	74	0.286	31
Informationen:negative	0.301	68	0.505	46
Zugfahrt:positive	0.276	62	0.343	83
DB App und Website:negative	0.232	39	0.375	33
DB App und Website:neutral	0.188	23	0	11
Sonstige Unregelmäßigkeiten:neutral	0.179	13	0.222	2
Allgemein:positive	0.157	86	0.586	92
Service und Kundenbetreuung:positive	0.115	23	0	5
Atmosphäre:positive	0.105	26	0	15
Ticketkauf:neutral	0.040	144	0.222	25
Connectivity:neutral	0	11	0.211	15
Toiletten:negative	0	15	0.160	23
<i>Rest</i>	0	355	0	115

Table 17: Micro-averaged F1 scores and support by `Aspect+Sentiment` entity with exact match (Subtask D1). 35 categories are summarized in `Rest`, each of them exhibiting a score of 0.

For Subtask D1, the model returns a positive score on 25 entity categories on at least one of the two test sets. The category `Zugfahrt:negative` can be classified best on both test sets, followed by `Sonstige Unregelmäßigkeiten:negative` and `Sicherheit:negative` for the synchronic test set and by `Connectivity:negative` and `Allgemein:positive` for the diachronic set. Visibly, the scores between the two test sets differ more here than in the classification report of the previous task.

The report for the overlapping match (cf. Tab. 18) shows slightly better results on some categories

9. Re-Evaluating GermEval17 using German pre-trained language models

Category	test _{syn}		test _{dia}	
	Score	Support	Score	Support
Zugfahrt:negative	0.708	622	0.739	495
Sonstige Unregelmäßigkeiten:negative	0.697	693	0.617	484
Sicherheit:negative	0.607	337	0.475	122
Connectivity:negative	0.598	56	0.620	109
Barrierefreiheit:negative	0.595	14	0	3
Auslastung und Platzangebot:negative	0.579	66	0.447	31
Connectivity:positive	0.571	26	0.555	60
Allgemein:negative	0.561	807	0.363	139
Atmosphäre:negative	0.505	403	0.358	164
Ticketkauf:negative	0.383	96	0.583	74
Ticketkauf:positive	0.368	59	0	13
Komfort und Ausstattung:negative	0.357	24	0	16
Atmosphäre:neutral	0.348	40	0.111	14
Service und Kundenbetreuung:negative	0.323	74	0.286	31
Informationen:negative	0.301	68	0.505	46
Zugfahrt:positive	0.276	62	0.343	83
DB App und Website:negative	0.261	39	0.406	33
DB App und Website:neutral	0.188	23	0	11
Sonstige Unregelmäßigkeiten:neutral	0.179	13	0.222	2
Allgemein:positive	0.157	86	0.586	92
Service und Kundenbetreuung:positive	0.115	23	0	5
Atmosphäre:positive	0.105	26	0	15
Ticketkauf:neutral	0.040	144	0.222	25
Connectivity:neutral	0	11	0.211	15
Toiletten:negative	0	15	0.160	23
Rest	0	355	0	112

Table 18: Micro-averaged F1 scores and support by *Aspect+Sentiment* entity with overlapping match (Subtask D2). 35 categories are summarized in *Rest* and show each a score of 0.

than for the exact match. The third-best score on the diachronic test data is now *Sonstige Unregelmäßigkeiten:negative*. Besides this, the top three categories per test set remain the same.

Apart from the fact that this is a different kind of task than before, one can notice that even though the overall micro F1 scores are lower for Subtask D than for Subtask C, the model manages to successfully identify a larger variety of categories, i.e. it achieves a positive score for more categories. This is probably due to the more balanced data for Subtask D than for Subtask C2, resulting in a lower overall score and mostly higher scores per category.

Part IV.

Benchmark Studies

10. Benchmarking down-scaled transformer-based architectures

Chapter 10 is an attempt to gain a better understanding for the inner workings of large pre-trained models. By using a pre-defined standardized setting (same pre-training corpus, same computational resources, similar size) the individual effects and interactions of pre-training objective, embedding dimension, number of layers as well as pre-training steps and batch size on model performance are evaluated. The acquired insights are subsequently transferred to larger models by attempting to transfer the concept of model scaling from Computer Vision to NLP.

Contributing article:

Aßenmacher, M., Schulze, P., and Heumann, C. (2021). Benchmarking down-scaled (not so large) pre-trained language models. *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021), Düsseldorf, Germany, September 6-9, 2021*. <https://aclanthology.org/2021.konvens-1.2>.

Copyright information:

This article is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>).

Author contributions:

Matthias Aßenmacher came up with the idea of systematically investigating the influence of different architectural dimensions on the model performance and transferring the concept of model scaling to Transformer-based language models. He designed the frame for all of the performed experiments, while Patrick Schulze worked on the implementation and also contributed with own ideas. Christian Heumann officially supervised this project and also contributed valuable input. The paper was jointly written and reworked by Matthias Aßenmacher and Patrick Schulze. All three of the authors finally revised and proofread the manuscript.

Supplementary material available at:

- Code: <https://github.com/PMSchulze/NLP-benchmarking>

Benchmarking down-scaled (not so large) pre-trained language models

Matthias Aßenmacher[♣]

Patrick Schulze[♣]

Christian Heumann[♣]

Department of Statistics
Ludwig-Maximilians-Universität
Ludwigstr. 33, D-80539 Munich, Germany

[♣]{matthias, chris}@stat.uni-muenchen.de, [♣]pa.schulze@campus.lmu.de

Abstract

Large Transformer-based language models are pre-trained on corpora of varying sizes, for a different number of steps and with different batch sizes. At the same time fundamental components, such as the pre-training objective or architectural hyperparameters, are modified. In total, it is therefore difficult to ascribe changes in performance to specific factors. Since searching the hyperparameter space over the full systems is too costly, we pre-train down-scaled versions of several popular Transformer-based architectures on a common pre-training corpus and benchmark them on a subset of the GLUE tasks (Wang et al., 2018). Specifically, we systematically compare three pre-training objectives for different shape parameters and model sizes, while also varying the number of pre-training steps and the batch size. In our experiments MLM + NSP (BERT-style) consistently outperforms MLM (RoBERTa-style) as well as the standard LM objective. Furthermore, we find that additional compute should be mainly allocated to an increased model size, while training for more steps is inefficient. Based on these observations, as a final step we attempt to scale up several systems using compound scaling (Tan and Le, 2019) adapted to Transformer-based language models.

1 Introduction

The introduction of the Transformer (Vaswani et al., 2017) together with the application of transfer learning (Thrun and Pratt, 1998) has led to major advances in Natural Language Processing (NLP). While many different lines of research exist, most attention is generally paid to the largest systems which often reach new state-of-the-art (SOTA) results. The current trend is to scale up such systems to ever new orders of magnitude: 213M parameters in the Transformer, 300M parameters in BERT

(Devlin et al., 2019), 1.5B parameters in GPT-2 (Radford et al., 2019) and 175B in GPT-3 (Brown et al., 2020). Since these models are pre-trained on corpora of widely varying sizes, for a different number of training steps and with different batch sizes, comparability suffers (Aßenmacher and Heumann, 2020). At the same time, new systems often apply fundamentally different methods, such as using a different pre-training objective or modified architectural hyperparameters. While altering multiple components simultaneously can help achieve new SOTA results, which is an important endeavor, it is difficult to disentangle the effects of the various factors. Though there exist various ablation studies, these often show only a small excerpt from the broad spectrum of experimental opportunities and can thus not provide a comprehensive picture. In this work, we conduct a systematic study of three Transformer-based architectures with respect to several pre-training hyperparameters.

2 Related work

One line of research empirically derives generalization results for large neural NLP systems. Rosenfeld et al. (2019) study how the generalization error of language models (LMs) depends on model and data set size. Regarding model size, they provide an approximation of the test loss, assuming that a LM is scaled with respect to a pre-defined scheme, such as increasing solely the embedding dimension. A related but more comprehensive study was conducted by Kaplan et al. (2020), examining power laws of the test loss when scaling large neural LMs with respect to a broad variety of different dimensions. These dimensions include architectural hyperparameters, model size, data set size, number of training steps and batch size. A central question in their work is how these factors can be combined to attain an optimal performance given a fixed amount of compute.

Compute efficient training is also investigated by Li et al. (2020), recognizing that an optimal allocation of computational resources is crucial for improving model performance. Considering Masked Language Modeling (MLM) pre-training, Li et al. (2020) examine the optimal choice of number of training steps and batch size in the relation to the model size. In a large-scale study, Raffel et al. (2019) cover an even broader variety of modeling scenarios than Kaplan et al. (2020), but train a much smaller number of systems per scenario. For instance, they include several variants of the Transformer, different pre-training objectives and various fine-tuning strategies in their analysis. Finally, based on their observations, Raffel et al. (2019) also scale-up a system to 11B parameters.

3 Materials and Methods

Pre-training data We pre-train all models on WikiText-103¹ (Merity et al., 2016), a large-scale text corpus for training and evaluating language models on long-range contexts, which has served as an evaluation data set (Radford et al., 2019; Dai et al., 2019; Shoeybi et al., 2019) as well as for pre-training (Howard and Ruder, 2018). We pre-train all models on the training set of WikiText-103, which allows for learning long-range dependencies (Rae et al., 2019). The validation set is employed to compare different architectures by their validation loss during pre-training. WikiText-103 is much smaller than most pre-training corpora of modern language models. For instance, Devlin et al. (2019) trained BERT on a 3,300M words corpus, which is approximately 32x the size of WikiText-103. Aside from this, pre-training data sets of different models often vary considerably in size, which makes fair comparisons difficult (Aßenmacher and Heumann, 2020). Pre-training on the same corpus allows us to exclude the amount and quality of pre-training data as confounding factors when evaluating the different model components.

Models We compare three different model types: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and GPT-2 (Radford et al., 2019). BERT is a bidirectional Transformer encoder which is trained with both MLM as well as Next Sentence Prediction (NSP). Its direct successor RoBERTa relies on the exact same architecture and differs from BERT solely in the pre-training procedure. Amongst other

¹www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

changes, Liu et al. (2019) abandoned the NSP objective and introduced a dynamic masking² procedure for the MLM objective³. GPT-2 is a Transformer decoder, and thus a unidirectional model, trained with the standard LM objective.

Since we train a multitude of down-scaled versions for each model type, thus modifying the specifications of the original models, we introduce the following conventions: We label models trained with MLM & NSP as *BERT-style*, models trained with MLM as *RoBERTa-style*, and models trained with LM as *GPT-2-style*. Alongside with the pre-training objectives, we also use the respective tokenizers of the different models. This means using byte-level BPE (Radford et al., 2019) for *RoBERTa-* and *GPT-2-style* and the WordPiece algorithm (Schuster and Nakajima, 2012) for *BERT-style* models, all of them exhibiting a uniform vocabulary size of 30,000 tokens

Fine-tuning data We fine-tune and evaluate our systems on GLUE (Wang et al., 2018). We mainly compare performances on MNLI (Williams et al., 2017), QQP (Shankar et al., 2017) and QNLI (Wang et al., 2018), which are the three largest GLUE tasks, since the results on these tasks are the most reliable. In particular, we therefore calculate the average score over the validation set performances of the three tasks, which we denote by *GLUE-Large*. For MNLI, we consider only the matched validation set when calculating this score. Whenever meaningful results for the two next largest data sets SST-2 (Socher et al., 2013) and CoLa (Warstadt et al., 2019) were achieved⁴, those will also be reported.

Training details Hyperparameters and the pre-training/fine-tuning procedure are largely adopted from the original models (cf. Appendix A and B).

4 Experiments

4.1 Comparison of different Shapes⁵

In computer vision it has been observed that the performance of a neural network strongly depends on the choice of architectural hyperparameters, such

²We also use dynamic masking throughout this study.

³There were further alterations, none of which are crucial for our experiments since we are using fixed pre-training data sets, batch sizes, learning rates, etc. for better comparability.

⁴For the smaller model sizes the performance on these smaller data sets did not significantly differ from zero.

⁵There exist several other choices, but examining the entire spectrum of possible shapes is out of the scope of this study.

as width or depth (Tan and Le, 2019). In contrast, Kaplan et al. (2020) observed a similar LM test loss over a wide range of shape parameters. Similarly, for MLMs, Li et al. (2020) found that the validation loss does not depend strongly on the model shape. This holds true also for the MNLI validation accuracy of fine-tuned systems.

In this study, we examine the impact of three different architectural hyperparameters in Transformer-based models: *depth*, *width* and the *number of attention heads*. Depth is given by the number of layers L . Stacking many layers in Transformer-based systems can be somewhat inefficient and does not always lead to a considerable increase in performance (Lan et al., 2019). Width corresponds to the embedding dimension H . Increasing H has in general produced slightly better results than increasing L in Transformer-based systems (Lan et al., 2019; Raffel et al., 2019; Li et al., 2020). Attention Heads are used to discriminate between different regions of the embedding space. In most applications of the Transformer, the number of attention heads A is set in fixed relation to H , such as $H = 64 \times A$. Decreasing performance has been reported for larger ratios (Vaswani et al., 2017; Brown et al., 2020).

4.2 Model Size, Training Steps and Batch size

Several recent studies have investigated the problem of compute efficient training of Transformer-based systems (Raffel et al., 2019; Li et al., 2020; Kaplan et al., 2020). The consensus among these studies is that, under a restricted budget, optimal performance is achieved by training very large models and stopping training well before convergence. Furthermore, additional compute should rather be used to increase the batch size instead of training for more steps. To examine convergence characteristics, we monitor the pre-training validation loss of several systems and test how this loss corresponds to different model sizes and shapes. Additionally, we conduct experiments regarding the effect of the batch size and the number of training steps. In particular, we evaluate how the training time and the model performance depend on both factors.

4.3 Definition of the Model Size

We follow Kaplan et al. (2020) and use the approximate number of non-embedding parameters to define the model size, which we denote as N_{model} . Since the share of embedding parameters decreases for larger models, similarly to Kaplan et al. (2020)

we expect that discarding the number of embedding parameters allows for better generalization of our results to large models. Another advantage of defining the model size as the number of non-embedding parameters is that it is closely linked to the number of (non-embedding related) floating point operations (FLOPs) per input token (Kaplan et al., 2020). This enables us to design benchmarking scenarios by training different models of comparable size, which at the same time require roughly similar amounts of computation.

Omitting biases and other sub-leading terms, the number of non-embedding parameters is given by

$$N_{\text{model}} = 12LH^2, \quad (1)$$

assuming that queries, keys and values are all transformed to dimension $\frac{H}{A}$ and the feed-forward dimension is $4H$. For a more in-depth explanation, please see Appendix E.

5 Results⁶

We start by evaluating how varying single shape dimensions affects the performance on GLUE-Large for the three different pre-training objectives (cf. Sec. 5.1). This aims at investigating whether the performance gain diminishes after a certain level, comparing how the performance changes when scaling different dimensions, and examining whether models with different pre-training objectives respond differently to single-dimension scaling. Subsequently in Section 5.2, we change multiple shape dimensions simultaneously to investigate whether the different dimensions depend on each other. In Sections 5.3 and 5.4 we study how to train efficiently by varying the model size, the number of training steps and the batch size. In Section 5.5 we put together our observations from the previous sections and scale networks to different sizes.

5.1 Scaling Single Shape Dimensions

In this section, we separately scale L and H , while holding all other dimensions constant. As shown in Figure 1, BERT-style systems perform significantly better than GPT-2-style and RoBERTa-style systems on GLUE-Large, contrary to the results of Liu et al. (2019) and in line with the original findings of Devlin et al. (2019).

Observation 1 *The pre-training objective has a large impact on the performance of a fine-tuned*

⁶Source code: <https://github.com/PMSchulze/NLP-benchmarking>

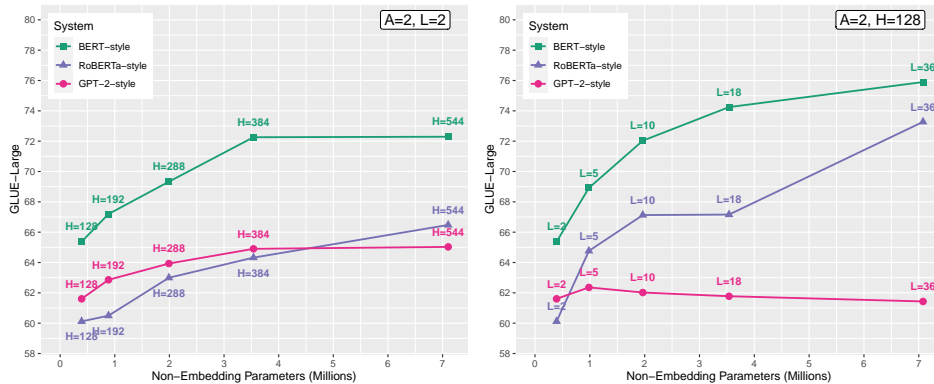


Figure 1: Average score on GLUE-Large, when varying H (left) vs. when varying L (right). For detailed performance values on the single tasks, see Table 5 and Table 6 in Appendix C.

system. Pre-training with the combination of MLM & NSP achieves the best results on sentence-pair tasks⁷, while pre-training with the unidirectional LM objective shows in general the worst performance.

Furthermore, for BERT-style systems the average performance is a relatively smooth function of the model size. Scaling up H results in an increasing performance, which saturates at approximately 72%, while for L we cannot clearly see this saturation (even not at 75%). For RoBERTa-style systems, the difference between scaling L and H individually is much larger. Furthermore, a saturation (as for BERT-style systems) can not be observed.⁸ For GPT-2-style systems, the average score slightly increases when scaling the embedding size, but interestingly, stacking more layers shows no positive effect at all. This suggests that GPT-2-style systems require more pre-training data compared to BERT-style and RoBERTa-style systems.

Observation 2 *In most cases, the performance of a fine-tuned system increases up to a certain level when scaling either width or depth, but the progression depends strongly on the pre-training objective.*

5.2 Scaling Multiple Shape Dimensions

We next examine whether the performance can be improved by scaling multiple dimensions at the same time. First, we increase both H and L and

⁷Note that this does not necessarily generalize to other languages or other types of tasks.

⁸Note that the relatively low average score for the 18-layer RoBERTa-style system, shown in the right plot of Figure 1, is due to a weak performance on the QNLI task.

compare the performance with the results from Section 5.1. Fig. 2 shows that for RoBERTa-style and BERT-style systems, scaling both dimensions significantly improves the performance on GLUE-Large.

Observation 3 *Scaling multiple shape dimensions can lead to a better performance than scaling single dimensions.*

Therefore, we conclude that the shape dimensions are not independent of each other. For GPT-2-style systems, however, we do not observe a performance increase, as shown in Table 1.

BERT-Style				Validation Set Performance	
A	H	L	N_{model}	GLUE-Large	
2	204	7	3,495,744	77.1	
2	256	9	7,077,888	78.6	
8	544	2	7,102,464	78.4	
GPT-2-Style				Validation Set Performance	
A	H	L	N_{model}	GLUE-Large	
2	204	7	3,495,744	63.6	
2	256	9	7,077,888	63.8	
8	544	2	7,102,464	66.0	
RoBERTa-Style				Validation Set Performance	
A	H	L	N_{model}	GLUE-Large	
2	204	7	3,495,744	72.9	
2	256	9	7,077,888	75.0	
8	544	2	7,102,464	70.9	

Table 1: Performance on GLUE-Large when increasing multiple shape dimensions at the same time.

So far, we did not increase A when scaling H and observed that, without using more attention heads, wide systems perform worse than deep systems (cf. Fig. 1). To evaluate whether a larger num-

10. Benchmarking down-scaled transformer-based architectures

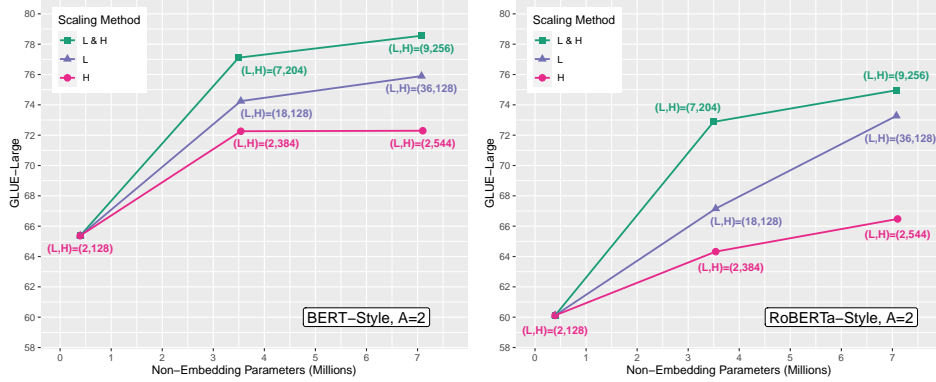


Figure 2: Performance on GLUE-Large when increasing multiple shape dimensions.

ber of attention heads can boost the performance of wide systems, we re-implement our widest systems with $A = 8$ attention heads, which corresponds to $\frac{H}{A} = 68$. We observe that the score of the widest system on GLUE-Large improved substantially by doing so (cf. Fig. 1 and Tab. 1). In particular, when using $A = 8$ instead of $A = 2$, the wide BERT-style system ($A = 8, H = 544, L = 2$) performs even better than the deep BERT-style system of comparable size ($A = 2, H = 128, L = 36$). Furthermore, as also shown in Table 1, the wide BERT-style system (with increased A) performs close to the balanced one ($A = 2, H = 256, L = 9$).

Observation 4 *The fine-tuning performance can be similar over a wide range of shapes. For BERT-style systems, wide systems perform slightly better than deep systems, if the number of attention heads is adapted to the embedding dimension.*

In contrast to BERT-style systems, deep RoBERTa-style systems still perform better than wide systems, even when increasing the number of attention heads. For GPT-2-style systems, adding more attention heads hardly increases the performance.

5.3 Monitoring the Validation Loss

In the previous sections, different models were made comparable by their number of non-embedding parameters. As stated in section 4.3, this number is related to the computational cost when evaluated as the number of FLOPs per token. Reporting the computational cost in FLOPs neglects, however, that some operations can be run in parallel, while others cannot. In order to assess the speed of convergence, following Li et al. (2020),

we therefore directly report the wall-clock time in seconds.

Figure 3 shows the validation loss for BERT-style systems of different shape, when pre-trained on the short sequences.⁹ The left plot depicts several pre-training loss curves corresponding to the single-dimension scaling experiments from Section 5.1. Interestingly, when comparing the validation loss with the GLUE-Large results (cf. Fig. 1), we find that, although increasing H (while holding A fixed) results in a lower validation loss than increasing L , the GLUE-Large score shows a higher increase in the latter case.

Observation 5 *The pre-training validation loss is not necessarily a good indicator for the performance of a fine-tuned system.*

Dependent on the downstream task some architectures presumably favor fine-tuning more than others, which can offset a relatively worse initialization point. This finding suggests that, although Kaplan et al. (2020) observe similar test losses for different shapes, benchmarking the corresponding fine-tuned versions may present a different picture.

In the left plot of Figure 3 we furthermore observe that shape has a significant effect on the pre-training time. In particular, stacking many layers requires much longer pre-training. It is also evident that increasing the size does not lead to a proportionate increase in the pre-training time. This holds true especially when scaling multiple dimensions, as depicted in the right plot of Figure 3. When doubling the number of pre-training parameters, the

⁹We do pre-training on short and long sequences. For a detailed description, see Appendix A and Appendix F.

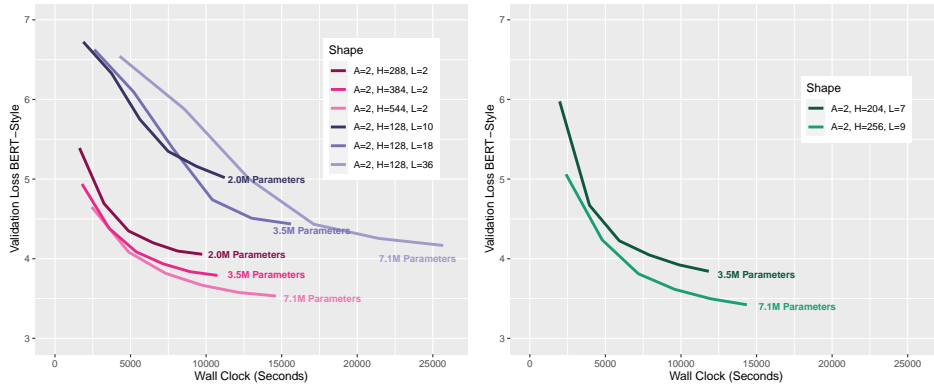


Figure 3: Loss curves of BERT-Style systems of different shape. All loss curves are associated with the first stage of pre-training, where we train on short sequences with a of 128 tokens (For the loss curves for the subsequent training on the long sequences, see Appendix D). The depicted parameter counts refer to the model size N_{model} .

training time only increases from approximately 11,800 seconds to approximately 14,400 seconds. In particular, the loss of the larger system is smaller at any measured point in time.

Observation 6 *Given a fixed time budget, training large systems for a relatively small number of steps is more efficient than training small systems for a large number of steps.*

The 9-layer system in the right plot of Figure 3 achieves a notably lower validation loss than the 7-layer system after 10,000 seconds, which corresponds to approximately 65,800 and 79,800 steps, respectively. Li et al. (2020) made a similar observation by showing that larger Transformer-based systems generally reach a lower pre-training validation perplexity in shorter time. A point of concern might be that larger systems overfit more easily during fine-tuning. However, Li et al. (2020) showed that, when stopping models of different size at the same pre-training validation perplexity, large systems generally achieve comparable downstream task performances to small systems, which contradicts the overfitting argument.

5.4 Number of Training Steps and Batch Size

The amount of processed data can be increased by increasing either the number of training steps or the batch size. In Table 2 we compare how halving the number of steps vs. halving the batch size impacts model performance. As baseline we use our best performing system thus far ($A = 2, H = 256, L = 9$), pre-trained RoBERTa- and BERT-style.

In both cases we find that reducing the number of training steps is more detrimental to the performance than reducing the batch size. Conversely, it follows that when scaling up a system, a better model performance can be achieved when doubling the amount of training steps than when doubling the batch size, which is consistent with the results of Raffel et al. (2019). On the other hand, we observe that the systems with the smaller batch size were trained for a significantly longer time than the systems with the reduced number of training steps. Therefore, increasing the batch size may result in a more favorable training duration than increasing the number of training steps. The modest drop in GLUE-Large performance, when halving the number of training steps is consistent with our findings from Section 5.3 and provides additional evidence that training for a large number of steps is inefficient.

Observation 7 *Doubling the number of training steps marginally increases the downstream task performance, whereas doubling the batch size significantly reduces the average training time of an input sequence.*

As stated, several other studies have shown that using a larger batch size is in general more efficient than training for more steps (Kaplan et al., 2020). This means that the reduction of training time by using larger batches dominates the marginal performance gains resulting from an increased number of training steps. However, for each specific model and training configuration there exists a critical batch size, after which the performance hardly im-

BERT-Style		Validation Set Performance				
Training Strategy	Total Time	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
Baseline	21, 358s	78.6	72.0/72.7	81.2	82.5	83.4
$\frac{1}{2}$ x steps, 1x batch	10, 736s	77.4	70.2/71.2	80.5	81.5	82.5
1x steps, $\frac{1}{2}$ x batch	14, 575s	78.2	71.5/71.9	80.9	82.3	83.9

RoBERTa-Style		Validation Set Performance				
Training Strategy	Total Time	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
Baseline	19, 760s	75.0	68.4/70.9	78.2	78.3	75.0
$\frac{1}{2}$ x steps, 1x batch	9, 906s	73.7	67.0/69.0	76.7	77.4	83.5
1x steps, $\frac{1}{2}$ x batch	13, 101s	75.6	68.2/70.0	79.5	78.9	84.4

Table 2: GLUE results and total pre-training time when halving batch size vs. number of training steps.

proves, if at all (Kaplan et al., 2020; Li et al., 2020). Our results suggest that this critical size is very small in our experiments, which we believe is due to the small size of the pre-training data set, as also observed by Kaplan et al. (2020).

5.5 Systematic Scaling

In this section we apply a modified version of the compound scaling method that was used to scale up EfficientNet (Tan and Le, 2019), a model that achieved a notably better accuracy on ImageNet (Deng et al., 2009) than previous approaches using less compute. For scaling, we only consider BERT-style systems and propose the following compound scaling method for Transformer-based systems:

$$L = \alpha^\phi, \quad H = \beta^\phi, \quad A \approx H/64, \quad (2)$$

s.t. $\alpha\beta^2 \approx 2$, with $\alpha \geq 1, \beta \geq 1$.

For suitable values of α and β , a system is scaled up by increasing the *compound coefficient* ϕ . Doubling L doubles N_{model} , while H leads to a fourfold increase. Since N_{model} dominates the amount of compute in a Transformer, the constraint $\alpha\beta^2 \approx 2$ thus ensures that when scaling the network from ϕ_{old} to ϕ_{new} , the amount of compute (which is approx. independent of A) approximately increases by the factor $2^{\phi_{\text{new}} - \phi_{\text{old}}}$. Following existing approaches and using Observation 4, we therefore set the number of attention heads to $A \approx H/64$.

Grid search To determine α and β , we follow Tan and Le (2019) and perform a grid search over a set of nine small networks of comparable size trained only on the short sequences. Subsequently, we select the three systems with the lowest validation loss. Based on Observation 5, we then fine-tune and evaluate these three systems on GLUE-Large, which leads to the best performing system

having $L = 3$ and $H = 104$ (cf. Tab. 7 in Appendix C). From the constraint in Eq. (2) it follows that the size of this system corresponds to a compound coefficient of $\phi = \log_2(LH^2) = 14.99 \approx 15$, such that we obtain $\alpha = 3^{\frac{1}{15}} \approx 1.076$, $\beta = 104^{\frac{1}{15}} \approx 1.363$. Note that the resulting coefficients favor scaling width over depth. In general, we believe that this is reasonable, especially in light of the much longer training times of deep networks compared to wide networks (cf. Fig. 3). However, we also want to emphasize that further research is needed, whether these scaling coefficients are suitable for BERT-style systems. For GPT-2-style systems, Kaplan et al. (2020) proposed to scale such that width/depth remains fixed. Importantly, however, Kaplan et al. (2020) did not study the effect of shape parameters on the GLUE-Large performance, but instead only monitored the LM test loss. In machine translation, on the other hand, Transformer-based systems are scaled preferably by increasing width (Shazeer et al., 2018; Li et al., 2020). Other approaches focus on increasing depth, while making modifications to the Transformer to allow for more efficient training (Al-Rfou et al., 2019).

Scaling Based on Observation 6, we successively increase the compound coefficient to scale three systems to larger sizes than all previously trained systems, but train for less steps. For our smallest system, we train for 5 epochs on both the long and the short sequences.¹⁰ The results are listed in Table 4. Furthermore, Table 3 shows a comparison of the smallest of the three systems to the best performing system so far, as well as to a modification of this system which fulfills the requirement

¹⁰Since validation loss on the long sequences did not further decrease after 3 epochs, the two larger systems were only trained for 3 epochs on these sequences (cf. Appendix D).

ϕ	BERT-Style				Total Time	Epochs	Validation Set Performance		
	A	H	L	N_{model}			GLUE-Large	Final Loss	
NA	2	256	9	7,0778,88	21,358s	6	78.6	3.24	
NA	4	256	9	7,0778,88	21,703s	6	78.9	3.29	
19.865	7	469	4	10,558,128	20,873s	5	79.4	3.13	

Table 3: Verification of the scaling method: The proposed modifications lead to a better GLUE score and a lower validation loss, while requiring less training time compared to previous best performing models.

ϕ	BERT-Style				Validation Set Performance					
	A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA
20.578	9	585	5	20,553,500	80.7	75.3/75.5	83.5	83.4	85.1	16.5
21.716	13	832	5	41,553,440	81.4	75.6/75.9	84.1	84.4	85.8	21.3

Table 4: GLUE results of BERT-style systems, scaled up based on the observations made in the previous sections.

$A \approx H/64$. As can be observed, both the performance on the large GLUE-Large tasks and the final validation loss are improved, while requiring less training time. For the two larger systems, each obtained by approximately doubling the model size, downstream performance and validation loss are further improved (cf. Tab. 4). Note that these systems are rather large compared to the amount of pre-training data. This demonstrates the remarkable robustness of these systems with respect to overfitting on the pre-training data, which is in line with the results of Kaplan et al. (2020).

6 Conclusion & Future work

Limitations The most severe limitation is the small pre-training data set. Based on the observations of Kaplan et al. (2020), systems train faster if more training examples are used. The small size of the pre-training data set might also be the cause of overfitting on smaller tasks. Therefore, for further experiments, we suggest to expand the amount of pre-training data. Furthermore, we did no hyperparameter tuning, but instead adopted the configurations from the original models. It would be advisable to adjust the hyperparameters accordingly (Li et al., 2020), especially since we used different batch sizes as the original models.

Directions for Further Research Kaplan et al. (2020) studied the effect of the amount of pre-training data, however, not with regard to downstream task performance. Due to the fact that current NLP systems are trained on vastly different amounts of pre-training data, we believe that this relationship should be explored further.

Although attempts have been made to study the

relationship between different pre-training objectives and the performance on downstream tasks (Arora et al., 2019), this relation is yet not well understood. Empirically, contrastive pre-training objectives, such as replaced token detection (Clark et al., 2020) have shown very promising results. It would be interesting to extend the study to such contrastive objectives. Since we observed that the NSP task is beneficial for learning sentence-pair relationships, comparing it to ALBERT’s SOP task (Lan et al., 2019) could yield further insights.

Finally, by fine-tuning on a larger variety of tasks we could break down in more detail how different modeling choices affect the performances on different tasks. We believe that further investigation of such relationships will open many opportunities for future research.

Conclusion In our experiments, BERT-style systems consistently outperform RoBERTa-style and GPT-2-style systems. We therefore conclude that, at least in case of a relatively small pre-training data set, the combination of MLM & NSP is preferable to MLM or LM. Although our experiments were conducted on a much smaller scale than other studies, we were able to reproduce many previous findings. For instance, we observed that, provided multiple dimensions are scaled, systems with very different shapes can achieve similar performances.

Consistent with previous studies (Kaplan et al., 2020; Li et al., 2020) we found that it is in general inefficient to train until convergence and that training for more steps improves the performance rather marginally. Instead, in accordance with Kaplan et al. (2020), we believe that increasing the batch size is more beneficial than training for more steps.

More importantly, also consistent with the results of Kaplan et al. (2020) and Li et al. (2020), we conclude that the model size is the key factor in Transformer-based systems. We observed that even for rather large systems, both the final pre-training validation loss and the GLUE performance benefit from further increasing the size. At the same time, the total pre-training time increases at a rather low rate. In particular, given a fixed time budget, large systems reach a lower loss than small systems. Therefore, we believe that additional compute should be allocated mainly to increase the model size.

Acknowledgements We would like to thank the three anonymous reviewers for their insightful comments and their feedback on our work.

References

- Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166.
- Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. 2019. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*.
- Matthias Aßenmacher and Christian Heumann. 2020. On the comparability of pre-trained language models. In *Proceedings of the 5th Swiss Text Analytics Conference and 16th Conference on Natural Language Processing*, Zurich, Switzerland (Online). CEUR Workshop Proceedings.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *OpenReview.net*.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*.
- 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:1907.11692*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.
- Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

-
- Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. 2019. [A constructive prediction of the generalization error across scales](#). *arXiv preprint arXiv:1909.12673*.
- Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.
- Iyer Shankar, Dandekar Nikhil, and Csernai Kornél. 2017. [First quora dataset release: Question pairs](#). Accessed: 2021-02-01.
- Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10414–10423.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. [Megatron-lm: Training multi-billion parameter language models using gpu model parallelism](#). *arXiv preprint arXiv:1909.08053*.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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*, pages 1631–1642.
- Mingxing Tan and Quoc V Le. 2019. [Efficientnet: Rethinking model scaling for convolutional neural networks](#). *arXiv preprint arXiv:1905.11946*.
- Sebastian Thrun and Lorien Pratt. 1998. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. [Glue: A multi-task benchmark and analysis platform for natural language understanding](#). *arXiv preprint arXiv:1804.07461*.
- Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. [A broad-coverage challenge corpus for sentence understanding through inference](#). *arXiv preprint arXiv:1704.05426*.
- 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. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Appendix

A Pre-training details

Training duration To ensure a fair comparison of the different pre-training objectives, we pre-train RoBERTa-style and GPT-2-style systems for 10 epochs, and BERT-style systems for 6 epochs, which in all cases equates to approximately 137,000 total training steps combined over both partitions.¹¹ Since the data is duplicated when training with MLM & NSP, it is natural to simply lower the number of epochs in relation to the amount of pre-training data. While the amount of pre-training data of RoBERTa-style and GPT-2-style systems amounts to more than 60% of the data of BERT-style systems, we found that, on the other hand, the average WordPiece token contains slightly more information than the average byte-level BPE token.

Optimization Apart from the experiments in section 5.4, we use a batch size of 64 when training on the short sequences and a batch size of 16 for the long sequences. We optimize all systems with Adam (Kingma and Ba, 2014) using the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-6$ and L_2 weight decay of 0.01. For BERT-style and RoBERTa-style systems we use a maximum learning rate of $1e-4$, and for GPT2-style systems the maximum learning rate is $2.5e-4$. In all cases we use a linear warmup for the first 1000 steps, which corresponds to approximately 1% of the total steps. Furthermore, for all systems we employ dropout with a rate of 0.1 on all layers. The activation function of all systems is the GELU (Hendrycks and Gimpel, 2016). The hyperparameters are in general chosen as in the original systems, except for RoBERTa-style systems, because RoBERTa was trained with significantly larger batches, which requires different hyperparameters. For RoBERTa-style systems we therefore choose the same hyperparameters as for BERT-style systems.

Implementation We pre-train all systems on a single NVIDIA 16GB V100 GPU, making use of the Hugging Face transformers library (Wolf et al., 2020). The same also holds true for fine-tuning.

Short and long sequences With our pre-training procedure we follow Devlin et al. (2019): The

¹¹In sections where we do not compare the different objectives the number of epochs may differ.

first 90% of the steps on short sequences (128 tokens), the remaining 10% on long ones (512 tokens). When inspecting the validation loss, we adjust the evaluation sequence lengths to the lengths of the training sequences, so ensure the same distribution for training and validation data. This causes the validation loss on the long sequences to start at a slightly higher point than the final validation loss on the short sequences (cf. Appendix D).

B Fine-tuning details

We follow Devlin et al. (2019) and train for three epochs on all GLUE tasks. We use a batch size of 16 and a learning rate of $2e-5$ for each task. Apart from these hyperparameter configurations, we apply the same fine-tuning procedures that were used by the original systems. For GPT-2-style systems, we implemented the fine-tuning approach of GPT (because GPT-2 was not fine-tuned).

However, we do make one small modification to the original implementations. In contrast to BERT-style systems, the pre-training objective of RoBERTa-style and GPT-2-style systems does not contain a classification task. When performing the NSP task, in the original BERT the contextualized representation of the CLS token is obtained by feeding the corresponding final hidden state through a linear layer with dropout and *tanh* activation. Subsequently, the contextualized representation is fed through another linear layer with dropout, which is the output layer mapping the contextualized representation to the class probabilities. Consequently, when fine-tuning BERT-style systems on a classification task, there are in fact two linear layers between the final hidden state and the output classes. However, RoBERTa and GPT in their original implementation use only one linear layer. In order to be as consistent as possible, in contrast, we use two linear output layers for all systems. The first linear layer is followed by a *tanh* activation and both layers are implemented with a dropout rate of 0.1. For more information regarding this issue see [huggingface’s discussion forum](#).

C Detailed performance values for single shape dimensions and results for the grid search

Performance values on GLUE-Large and SST-2 for scaling H (Tab. 5) and for scaling L (Tab. 6). Table 7 shows the results of the grid search.

BERT-Style				Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	65.4	59.0/60.2	72.3	64.8	78.0
2	192	2	884,736	67.2	62.1/62.8	74.0	65.4	82.6
2	288	2	1,990,656	69.3	63.7/65.2	76.0	68.3	82.0
2	384	2	3,538,944	72.3	65.7/66.6	77.8	73.2	81.1
2	544	2	7,102,464	72.3	66.8/68.1	78.0	72.0	83.3
GPT-2-Style				Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	61.6	56.3/56.2	66.1	62.3	79.8
2	192	2	884,736	62.9	58.0/58.4	68.7	61.9	79.7
2	288	2	1,990,656	63.9	58.7/58.7	70.9	62.2	81.7
2	384	2	3,538,944	64.9	59.8/59.6	71.9	63.0	81.2
2	544	2	7,102,464	65.0	59.8/59.7	72.4	62.9	82.5
RoBERTa-Style				Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	60.1	53.7/55.1	64.7	61.9	79.2
2	192	2	884,736	60.5	54.4/55.4	65.0	62.0	80.8
2	288	2	1,990,656	63.0	57.5/58.0	68.1	63.4	80.3
2	384	2	3,538,944	64.3	59.4/59.8	69.0	64.6	81.9
2	544	2	7,102,464	66.5	60.2/60.7	72.7	66.5	81.8

Table 5: Performance on GLUE when increasing only the embedding dimension.

BERT-Style				Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	65.4	59.0/60.2	72.3	64.8	78.0
2	128	5	983,040	68.9	62.1/64.2	75.0	68.6	79.8
2	128	10	1,966,080	72.0	65.3/66.9	76.7	74.1	81.8
2	128	18	3,538,944	74.2	67.2/68.6	77.8	77.7	82.2
2	128	36	7,077,888	75.9	69.7/70.4	79.7	78.3	83.3
GPT-2-Style				Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	61.6	56.3/56.2	66.1	62.3	79.8
2	128	5	983,040	62.4	57.6/56.1	67.4	62.0	80.5
2	128	10	1,966,080	62.0	56.9/57.0	67.7	61.5	81.4
2	128	18	3,538,944	61.8	56.1/56.4	66.8	62.4	80.6
2	128	36	7,077,888	61.4	56.6/56.7	66.6	61.1	80.7
RoBERTa-Style				Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	60.1	53.7/55.1	64.7	61.9	79.2
2	128	5	983,040	64.8	59.5/60.6	70.4	64.4	80.2
2	128	10	1,966,080	67.1	60.9/61.9	72.0	68.5	81.7
2	128	18	3,538,944	67.2	62.9/64.3	74.3	64.3	80.0
2	128	36	7,077,888	73.3	67.6/69.1	77.3	75.0	82.6

Table 6: Performance on GLUE when increasing only the number of layers.

BERT-Style				Validation Loss (WikiText-103)	Validation Performance (GLUE)
A	H	L	N_{model}	BERT-Style Loss	GLUE-Large
2	128	2	393,216	5.66	66.6
2	104	3	389,376	6.34	68.2
2	90	4	388,800	6.41	67.1
2	74	6	394,272	6.47	-
2	64	8	393,216	6.50	-
2	58	10	403,680	6.54	-
2	52	12	389,376	6.58	-
2	48	14	387,072	6.62	-
2	46	16	406,272	6.62	-

Table 7: Grid search over nine small BERT-style systems.

D Validation loss for scaled-up models

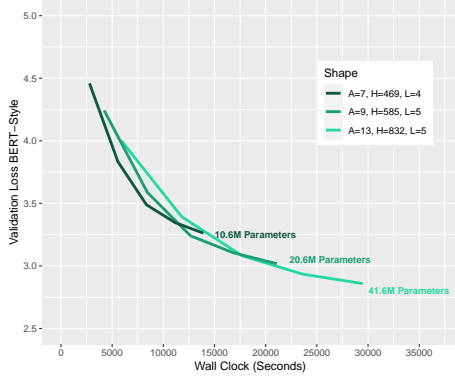


Figure 4: Validation loss of scaled-up BERT-style systems when pre-training on the short sequences. The depicted parameter counts refer to N_{model} .

E Definition of the model size

We follow Kaplan et al. (2020) and use the approximate number of non-embedding parameters to define the model size, which we denote as N_{model} . The embedding parameters consist of all token, position and (if present) segment embeddings. The number of embedding parameters does not depend on the network depth, and when scaling width and/or depth, it is a sub-leading term of the total number of parameters. Furthermore, the number of FLOPs related to embedding (and de-embedding) is also sub-leading term of the total number of FLOPs. Consistent with this is the observation of Kaplan et al. (2020) that discarding the number of embedding parameters when calculating model size and amount of compute results in significantly cleaner scaling laws. Since the share of embedding parameters decreases significantly for larger models, similarly to Kaplan et al. (2020) we expect that discarding the number of embedding parameters allows for a better generalization of our results to large models. Another advantage of defining the model size as the number of non-embedding parameters is that this number is closely linked to the number of (non-embedding related) FLOPs. This enables us to design benchmarking scenarios by training different models of comparable size, which at the same time require roughly similar amounts of computation.

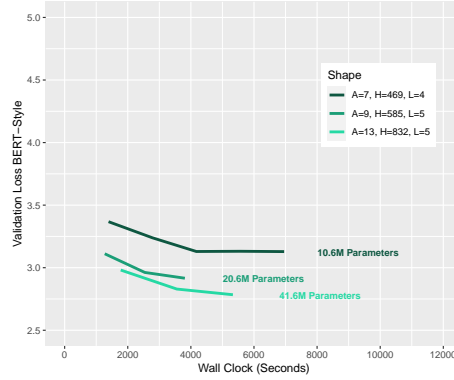


Figure 5: Validation loss of scaled-up BERT-style systems when pre-training on the long sequences. The depicted parameter counts refer to N_{model} .

Number of Non-Embedding Parameters

Omitting biases and other sub-leading terms, the number of non-embedding parameters, which is our definition of the model size, is given by

$$N_{model} := 12LH^2, \quad (3)$$

where we have assumed that $H_k = H_v = \frac{H}{A}$ and $H_{ff} = 4H$. Therefore, per layer there are approximately $12H^2$ non-embedding parameters. This number can be derived from the following three steps performed in each layer of a Transformer:

- 1. Input projection** For each attention head, the queries, keys and values of dimension $\frac{H}{A}$ are obtained with the three matrices \mathbf{W}_i^Q , \mathbf{W}_i^K , and \mathbf{W}_i^V , which are each of size $H \times \frac{H}{A}$. In total, the input projection thus consists of $3 \cdot A \cdot \frac{H^2}{A} = 3H^2$ parameters.

- 2. Output projection** First, note that performing attention on the projected inputs of dimension $\frac{H}{A}$ involves no additional parameters. The concatenated attention results are projected back to dimension H with the $H \times H$ matrix \mathbf{W}^O . Therefore, the output projection involves an additional set of H^2 parameters.

- 3. Feed-forward network** The last sub-layer of each layer consists of applying a feed-forward network to the output projections. There exist $H \cdot 4H$ connections between the output projections and the neurons of the inner-layer, and another $4H \cdot H$ connections from the inner-layer to the final output neurons. This step hence involves $8H^2$ parameters.

Note that the feed-forward network accounts for the majority of non-embedding parameters, followed by the input and output projections, respectively.

Relation to FLOPs

As stated, the number of non-embedding parameters is closely linked to the number of non-embedding related FLOPs. We start by deriving the number of FLOPs per token and forward pass for GPT-2-style systems, where sub-leading terms such as biases and layer normalization are again omitted.

1. Input projection The matrix-vector products of each per-layer input with \mathbf{W}_i^Q , \mathbf{W}_i^K , and \mathbf{W}_i^V involve approximately $3 \cdot 2 \cdot H \cdot \frac{H}{A}$ FLOPs per attention head. Considering all attention heads, the input projection thus requires approximately $6H^2$ FLOPs per token.

2. Attention The computation of the attention operation can be divided into two sub-components:

- **Computation of the weights:** On average, $\frac{N_{ctx}}{2}$ attention weights have to be computed per input token, since on average half of the tokens are masked for each input token. Computation of a dot-product attention weight requires approximately $2\frac{H}{A}$ FLOPs per head. In total, the computation of the attention weights hence involves approximately $N_{ctx}H$ FLOPs per token.
- **Computation of the weighted sum:** Since only half of the tokens are summed on average, given the attention weights, calculation of the weighted sum of the values has an average cost of approximately $N_{ctx}H$ FLOPs for each token.

3. Output projection The vector matrix product of the attention outputs with \mathbf{W}^O requires approximately $2H^2$ FLOPs for each token.

4. Feed-forward network The feed-forward network consists of two consecutive matrix multiplications, where each matrix contains $4H^2$ parameters. Thus, the feed-forward network requires approximately $2 \cdot 2 \cdot 4H^2 = 16H^2$ FLOPs per token.

The number of FLOPs per token and forward pass in GPT-2-style systems, which we denote by

$C_{forward}$, can hence be approximated as

$$\begin{aligned} C_{forward} &\approx L(6H^2 + N_{ctx}H + N_{ctx}H \\ &\quad + 2H^2 + 16H^2) \\ &= 24LH^2 + 2LN_{ctx}H \\ &= 2N_{model} + 2LN_{ctx}H. \end{aligned} \quad (4)$$

BERT-style and RoBERTa-style systems require slightly more FLOPs than GPT-2-style systems, because these systems have no autoregressive attention mask. Hence, in both steps of the attention operation above, the computational cost is approximately twice as much, i.e., $2N_{ctx}H$ in each step. Therefore, BERT-style and RoBERTa-style systems require approximately $2N_{model} + 4LN_{ctx}H$ FLOPs per token and forward pass. As mentioned by Kaplan et al. (2020), if $H > N_{ctx}/12$, the context-dependent term in Eq. (4) only accounts for a relatively small fraction of the compute of GPT-2-style systems. In particular, when increasing H , the importance of the context-dependent term diminishes. For BERT-style and RoBERTa-style systems the context-dependent term becomes small if $H > N_{ctx}/6$. Both constraints are satisfied by a large margin for all our systems, especially since we mainly train on rather short sequences. The backward pass requires approximately twice as much compute as the forward pass (Kaplan et al., 2020), such that the total amount of non-embedding related compute per token and training step can be approximated as

$$C := 6N_{model}. \quad (5)$$

F Sequence characteristics

The following Table 8 provides an overview on the number of tokens in short and long sequences.

System	Partition	Number of Tokens	
		Total	Average
BERT-Style	Short	110, 888, 186	110.04
	Long	43, 274, 856	375.52
RoBERTa-Style	Short	70, 025, 709	110.31
	Long	27, 692, 351	457.04
GPT-2-Style	Short	70, 564, 106	111.16
	Long	27, 729, 551	457.65

Table 8: Number of tokens for the short and the long sequences as well as the average sequence lengths resulting from the different tokenizers.

Part V.

Conclusion

11. Future Directions and Concluding Remarks

11.1. Few-Shot Learning

Since the time the latest model presented in Section 3.3 was proposed (ELECTRA, March 2020) the whole field of NLP has moved even further and different directions and approaches have been explored. ELECTRA seemed to be a suitable cut point, prior to including the Parts II – IV holding the publications constituting this thesis, since all architectures used in these publications were introduced prior to it. This last Chapter concludes this thesis by summing up what has happened since then and tries to give a tentative outlook into the future.¹

Already when introducing GPT-2, Radford et al. (2019) did set new standards with respect to model size, utilized computational power and size of the pre-training corpus at that time. But more importantly, they also already started to investigate the concept of *Zero-Shot Learning*, a transfer learning paradigm where the model is tested on its ability to execute tasks without any explicit supervision (i.e. no task-specific fine-tuning). The authors measured the zero-shot performance of their architecture on the language modeling task, the task they were initially pre-trained on, but for specific domains distinct from the domain of the pre-training corpus. Furthermore, they evaluated the model regarding its accuracy on the *Children’s Book Test* (CBT; Hill et al., 2015) and the *LAMBADA* (Paperno et al., 2016) benchmarks, both of which are cloze-style tasks.

Brown et al. (2020) went even further by training GPT-3, a model similar to GPT-2 regarding the basic architecture, but again by no means comparable regarding size and pre-training efforts. With a model size of up to 175B parameters and a pre-training corpus of 499 Billion tokens (approx. 50 times the size of the pre-training corpus from GPT-2), a new bar was set. Model performance is subsequently evaluated by transforming every evaluated task into a sequence-to-sequence format² and testing the model on it. Brown et al. (2020) relax the concept of *Zero-Shot Learning* by framing it more generally as *Few-Shot Learning*, where K denotes the number of example provided to the model. This in turn includes both *Zero-Shot* ($K = 0$) as well as *One-Shot Learning* ($K = 1$) as special cases. Model training of GPT-3 consists of two distinct phases:

Outer Loop: The model is pre-trained on the language modeling task (i.e. in a self-supervised fashion) via stochastic gradient descent. This is the only part of the training, where the model is subject to explicit (self-)supervision of the language modeling task. Regarding the expense of computational power, this part is extremely resource intensive due to the immense size of the pre-training corpus and the large number of parameters in the model to be updated.

¹Although, given the current speed and amount in which new research and new ideas are currently published, giving an outlook is pretty difficult and speculative.

²This procedure is comparable to what Raffel et al. (2019) do prior to fine-tuning T5 regarding task formulation. But differently from T5, GPT-3 is *not* fine-tuned, i.e. no gradient updates are performed using the task-specific data.

Inner Loop (In-Context Learning): The model is provided with a task description, followed by exactly K examples (i.e. source and target sequence) and eventually a single prompt (i.e. only a source sequence). This *In-Context Learning* does not happen via explicit supervision, which would include gradient updates, just like in the traditional pre-training + fine-tuning setting. Rather the model "learns" by the demonstration of K examples which task (and how) to perform and transfers this to the prompt. This makes the model input look as follows:

```
task description
source => target
      ⋮
source => target
prompt => ...
```

While its predecessor GPT-2 was only evaluated on pure language modeling and cloze-style tasks, GPT-3 also showed impressive performance on a wider range of benchmark tasks including (amongst others) NMT tasks, the SuperGLUE benchmark and various reading comprehension tasks. Already when using only a small number of labelled examples for demonstration ($K \leq 64$ in most cases), GPT-3 already achieves a pretty competitive performance compared to fine-tuned BERT models.

11.2. Final Thoughts

As already mentioned in Sec. 1.2, controversial discussions about recent developments and directions are pretty prominent in the current scientific debate, which is why I will conclude this thesis with three final thoughts:

- It remains an open question, which of these two paradigms (fine-tuning vs. few-shot) will prove to be superior in the future. Probably there will also be a hybrid of both, since the few-shot approach suffers from its inflexibility: GPT-3 was trained on a huge amount of internet data from a specific time period, which is why it will most likely have a hard time adopting to (future) texts with a large temporal distance to the time frame in which its training data was collected.
- Ever larger models set the bars higher and higher when it comes to performance and the amount of consumed data, but often lack explainability/interpretability and are difficult to deploy. It will be interesting to see where the field is heading to, whether model sizes are further increasing or if smaller models like e.g. iPET (Schick and Schütze, 2020b) will succeed.
- Clark et al. (2021) recently brought up the issue of why tokenization, a step preceding *every* large pre-trained language model, might be problematic. It will be interesting to see, if and to what extent their proposal of tokenization-free encoders will potentially shape the next generation of pre-trained language models.

With that said, there still are and will be a lot of open problems which pre-trained language models might help solving, for all of which a focus on model comparability and a proper benchmarking setup are important prerequisites.

Contributing Publications

- Aßenmacher, M. and Heumann, C. (2020). On the comparability of pre-trained language models. *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS), Zurich, Switzerland (Online), June 23-25, 2020*. <http://ceur-ws.org/Vol-2624/paper2.pdf>.
- Meidinger, M. and 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. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021), Vienna, Austria (Online), February 4-6, 2021, Vol. 2: 866-873*. <https://doi.org/10.5220/0010255108660873>.
- Guderlei, M. and Aßenmacher, M. (2020). Evaluating Unsupervised Representation Learning for Detecting Stances of Fake News. *Proceedings of 28th International Conference on Computational Linguistics (COLING), Barcelona, Spain (Online), December 8-11, 2020: 6339-6349*. <https://doi.org/10.18653/v1/2020.coling-main.558>.
- Viellieber, V. D. and Aßenmacher, M. (2020). Pre-trained language models as knowledge bases for Automotive Complaint Analysis. *arXiv preprint arXiv:2012.02558*. <https://arxiv.org/abs/2012.02558>.
- Lebmeier, E., Hou, N., Spann, K., and Aßenmacher, M. (2021). Creating a “Customer Centricity Graph” from unstructured customer feedback. *Applied Marketing Analytics, Vol. 6(3): 221-229*. <https://www.henrystewartpublications.com/ama/v6>.
- Aßenmacher, M., Corvonato, A., and Heumann, C. (2021). Re-Evaluating GermEval17 Using German Pre-Trained Language Models. *Proceedings of the Swiss Text Analytics Conference, Winterthur, Switzerland (Online), June 14-16, 2021*. <http://ceur-ws.org/Vol-2957/paper1.pdf>.
- Aßenmacher, M., Schulze, P., and Heumann, C. (2021). Benchmarking down-scaled (not so large) pre-trained language models. *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021), Düsseldorf, Germany, September 6-9, 2021*. <https://aclanthology.org/2021.konvens-1.2>.

Further References

- Paul H Algoet and Thomas M Cover. 1988. A sandwich proof of the shannon-mcmillan-breiman theorem. *The annals of probability*, pages 899–909.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.
- Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Jared Davis, Tamas Sarlos, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.
- Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. Canine: Pre-training an efficient tokenization-free encoder for language representation. *arXiv preprint arXiv:2103.06874*.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. ELECtra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Thomas M Cover and Joy A Thomas. 1991. Entropy, relative entropy and mutual information. *Elements of information theory*, 2(1):12–13.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. **Transformer-XL: Attentive language models beyond a fixed-length context**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.
- Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.
- Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.
- Eva Gibaja and Sebastián Ventura. 2014. Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(6):411–444.
- Eva Gibaja and Sebastián Ventura. 2015. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, 47(3):1–38.
- Aaron Gokaslan and Vanya Cohen. 2019. **Openwebtext corpus**.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Jan Gorodkin. 2004. Comparing two k-category assignments by a k-category correlation coefficient. *Computational biology and chemistry*, 28(5-6):367–374.

Further References

- Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children’s books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Jigsaw and ConversationAI. 2018. [Toxic comment classification challenge](#). Accessed: 2020-08-31.
- Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18, page 5.
- Cornelia Kiefer. 2019. Quality indicators for text data. *BTW 2019–Workshopband*.
- Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.
- Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.
- 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:1907.11692*.

- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.
- Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.
- Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.
- Eneldo Loza Mencia and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer.
- Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016a. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016b. [Wikitext-103](#). Accessed: 2020-02-10.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016c. [Wikitext-2](#). Accessed: 2020-02-10.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.
- Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.
- Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.
- Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197.
- Juri Opitz and Sebastian Burst. 2019. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*.

Further References

- Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambda dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *10th International Workshop on Semantic Evaluation (SemEval 2016)*.
- Maria Pontiki, Dimitrios Galanis, Harris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 486–495.
- Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. [SemEval-2014 task 4: Aspect based sentiment analysis](#). In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.
- Alec Radford. 2018. [Improving language understanding with unsupervised learning](#). Accessed: 2020-02-10.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).
- Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillcrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. [Know what you don't know: Unanswerable questions for SQuAD](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. <http://is.muni.cz/publication/884893/en>.
- Margaret E Roberts, Brandon M Stewart, and Edoardo M Airolidi. 2016. A model of text for experimentation in the social sciences. *Journal of the American Statistical Association*, 111(515):988–1003.
- Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.
- Sebastian Ruder. 2019. *Neural transfer learning for natural language processing*. Ph.D. thesis, NUI Galway.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few-shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.
- Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.
- Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.
- Dmitriy Selivanov, Manuel Bickel, and Qing Wang. 2020. *text2vec: Modern Text Mining Framework for R*. R package version 0.6.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural machine translation of rare words with subword units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Further References

- Iyer Shankar, Dandekar Nikhil, and Csernai Kornél. 2017. [First quora dataset release: Question pairs](#). Accessed: 2020-02-10.
- Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.
- 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*, pages 1631–1642.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.
- Wilson L Taylor. 1953. “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.
- Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.
- Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

- Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. [GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback](#). In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12, Berlin, Germany.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.
- Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.
- Biqing Zeng, Heng Yang, Ruyang Xu, Wu Zhou, and Xuli Han. 2019. Lcf: A local context focus mechanism for aspect-based sentiment classification. *Applied Sciences*, 9(16):3389.
- Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12. Juli 2011, § 8 Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig,
ohne unerlaubte Beihilfe angefertigt ist.

München, den 16.07.2021

Matthias Aßenmacher

