Efficient Transfer Learning with Pretrained Language Models

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

eingereicht von
Mengjie Zhao

München, den 02. Januar 2022

Erstgutachter: Prof. Dr. Hinrich Schütze
Zweitgutachter: Prof. Dr. Marie-Francine Moens
Drittgutachter: Prof. Dr. Minlie Huang

Tag der Einreichung: 02. Januar 2022
Tag der mündlichen Prüfung: 26. April 2022

Eidesstattliche Versicherung
(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5.)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig ohne unerlaubte Beihilfe angefertigt ist.

München, den 02. Januar 2022.

_____
Mengjie Zhao

Abstract

Pretrained language models (PLMs) like BERT have been shown to encode rich linguistic information and prolific world knowledge. Through transfer learning, the PLMs significantly benefit a wide range of NLP tasks in diverse languages. However, two notable disadvantages come with the performance gains. First, plain transfer learning is *parameter-inefficient*, i.e., each downstream task requires a saved checkpoint for inference. This is problematic as PLMs often contain hundreds of millions of parameters and this amount is still fast growing. Second, plain transfer learning is *label-inefficient*, i.e., thousands of labeled annotations are still a crucial request of PLMs to perform well. This thesis thoroughly analyzes PLMs, exploring parameter- and label-efficient transfer learning methods.

The first publication investigates a widely adopted tokenization method in PLMs: Byte-Pair Encoding (BPE). We apply BPE in a language-agnostic way to tokenize texts of more than one thousand languages, and then create an embedding space accommodating them. We then transfer sentiment information from English to other languages to create sentiment lexicons for them.

The second publication investigates the contextualization procedures of words in BERT. We quantify the amount of contextualization by studying the extent to which semantic classes of a word can be accurately inferred from contextualized embeddings. Importantly, we show that pretrained knowledge about contextualization is still well preserved after finetuning BERT on downstream tasks.

Inspired by the second publication, we devise, in the third publication, an efficient method of transferring PLMs' knowledge to downstream tasks. We learn selective binary masks for pretrained weights in lieu of modifying them through finetuning. The new method achieves comparable performance to finetuning yet has a much smaller memory footprint when several tasks need to be inferred. Analyses of loss landscapes confirm the correctness of the new method.

The fourth publication investigates a label-efficient method, i.e., prompting, for crosslingual transfer with multilingual PLMs. Prompting reformulates classification tasks into cloze-style queries, better matching the pretraining objective of PLMs. We demonstrate that prompting outperforms finetuning in both few-shot crosslingual transfer and in-language training scenarios.

The fifth publication highlights a fundamental risk of conducting crosslingual transfer learning in few-shot scenarios: PLMs exhibit a high degree of sensitivity to the selection of few shots. We provide sampled few shots as a step towards standardizing few-shot crosslingual experiments.

The last publication exploits the utility of PLM-based few-shot learners. We propose LMTurk, which leverages PLMs to annotate resources for training an efficient model deployable in practical scenarios to solve a task. LMTurk is an important step towards making effective use of PLM-based few-shot learners.

Zusammenfassung

Es wurde gezeigt, dass vortrainierte Sprachmodelle (pretrained language models, PLMs) wie BERT reichhaltige sprachliche Informationen und Weltwissen kodieren können. Durch Transferlernen profitiert eine große Bandbreite von NLP-Aufgaben in vielen unterschiedlichen Sprachen von diesen PLMs. Allerdings geht diese Leistungssteigerung mit zwei wichtigen Nachteilen einher. Erstens ist einfaches Transferlernen *parameter-ineffizient*, d.h. jede Downstream-Aufgabe benötigt einen gespeicherten Modellcheckpoint für die Inferenz. Dies ist problematisch, da PLMs oft Millionen von Parametern haben und diese Zahl immer noch steil ansteigt. Zweitens ist einfaches Transferlernen *label-ineffizient*, d.h. es werden immer noch tausende annotierte Daten benötigt, damit das Modell gut funktionieren kann. Diese Dissertation analysiert PLMs gründlich und beschreibt parameter- und label-effiziente Methoden des Transferlernens. Die erste Veröffentlichung untersucht eine weit verbreitete Tokenisierungsmethode in PLMs, das Byte-Pair-Encoding (BPE). Wir wenden BPE in einer sprach-agnostischen Art an, um Texte aus mehr als eintausend Sprachen zu tokenisieren, und erstellen einen Embeddingraum für sie. Danach transferieren wir Sentiment-Information aus Englisch in die anderen Sprachen, um ihre Sentimentlexika zu erstellen. Die zweite Veröffentlichung untersucht die Kontextualisierungsprozeduren von Wörtern in BERT. Wir quantifizieren die Kontextualisierung, indem wir untersuchen, wie weit die semantische Klasse eines Wortes aus dem kontextualisierten Embedding vorhergesagt werden kann. Wir zeigen, dass Wissen aus dem Vortraining über die Kontextualisierung auch nach dem Finetunen auf Downstream-Aufgaben noch gut erhalten ist. Inspiriert von dieser Veröffentlichung entwickeln wir in der dritten Veroeffentlichung eine effiziente Methode, um das Wissen eines PLMs zu Downstream-Tasks zu transferieren. Wir lernen selektive binäre Maskierungen für die vortrainierten Gewichte, statt sie durch Finetuning zu modifizieren. Diese neue Methode erzielt zu Finetuning vergleichbare Performanz, benötigt jedoch deutlich weniger Speicherplatz, wenn mehrere Aufgaben bearbeitet werden müssen. Analysen der Loss-Landscapes bestätigen die Korrektheit der neuen Methode. Die vierte Veröffentlichung untersucht mit Prompting eine label-effiziente Methode für den crosslingualen Transfer mit mehrsprachlichen PLMs. Prompting formuliert Klassifikationsaufgaben in cloze-style Anfragen um, damit sie besser zu den Vortrainingszielen von PLMs passen. Wir zeigen, dass Prompting sowohl in few-shot crosslingualem Transfer als auch in innersprachlichen Trainingsszenarien besser funktioniert als Finetuning. Die fünfte Veröffentlichung hebt ein fundamentales Risiko von crosslingualem Transferlernen in few-shot Szenarien hervor: PLMs weisen eine hohe Sensitivität gegenüber der Auswahl der few shots auf. Wir stellen gesampelte few shots als einen Schritt in Richtung der standartisierten few-shot crosslingualen Experimente vor. Die letzte Veröffentlichung benutzt die Nützlichkeit

von auf PLMs basierenden few-shot Lernern. Wir stellen LMTurk vor, welches PLMs benutzt, um Ressourcen für das Training eines effizienten Modells, welches in praxisnahen Szenarien für die Lösung einer Aufgabe eingesetzt werden kann, zu annotieren. LMTurk ist ein wichtiger Schritt hin zu einer effektiven Verwendung von PLM-basierten few-shot Lernern.

Acknowledgement

I would like to thank my advisor Hinrich Schütze for his constant support throughout the past few years. For me, Hinrich is a gold standard advisor: He can quickly understand the problem that I am facing and provide practical suggestions. I appreciate the guidance and I've learned so much from him. I also want to thank Hinrich for allowing me to participate in many hiring processes. This largely improves my understanding of interviews, which I used to be bad at.

I would like to thank Professor Sien Moens and Professor Minlie Huang for spending time reading my thesis and filing the reports.

A big thank you goes to Peggy Hobmaier and Thomas Schäfer. I appreciate the invaluable help of processing the complex paperworks (especially for a foreigner!) and managing the computational resources.

I want to thank my great CIS colleagues: Philipp Dufter, Masoud Jalili, Martin Schmitt, Nina Poerner, Benjamin Roth, Nora Kassner, Timo Schick, Marina Sedinkina, Alena Moiseeva, Dietrich Trautmann, Sanjeev Kumar Karn, Viktor Hangya, Dario Stojanovski, Matthias Huck, Leonie Weissweiler, Latif Köksal, Silvia Severini, Kerem Senel, Antonios Maronikolakis, Sheng Liang, Ayyoob Imani, Peiqin Lin, and Lavine. I will miss our PhD seminars, reading groups, and excursions. Needless to say, thank you also goes to old EPFL friends Tao Lin, Fei Mi, Wei Ma, and Cong Wang.

I came to Europe in 2015 and the last seven years are beyond my imagination: Syrian refugee crisis, Brexit, COVID-19 pandemie, Ukraine crisis. I would like to thank Europe for her devotion in dignity, freedom, and equality – these are the key takeaways I've learned besides research.

Publications and Declaration of Co-Authorship

Chapter 2 corresponds to the following publication:

> **Mengjie Zhao**, Hinrich Schütze. *A Multilingual BPE Embedding Space for Universal Sentiment Lexicon Induction.* In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 3506–3517. 2019.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 3 corresponds to the following publication:

> **Mengjie Zhao**, Philipp Dufter, Yadollah Yaghoobzadeh, and Hinrich Schütze. *Quantifying the Contextualization of Word Representations with Semantic Class Probing.* In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP): Findings, pp. 1219–1234. 2020.

I conceived of the original research contributions and performed all implementations and evaluations except for the evaluation of different context window sizes in Table 7 (conducted by Philipp Dufter). I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my co-authors who assisted me in improving the draft.

Chapter 4 corresponds to the following publication:

> **Mengjie Zhao***, Tao Lin*, Fei Mi, Martin Jaggi, Hinrich Schütze. *Masking as an Efficient Alternative to Finetuning for Pretrained Language Models.* In Proceedings of the 2020 Conference on Empirical

Methods in Natural Language Processing (EMNLP), pp. 2226–2241. 2020. (*equal contribution).

I conceived of the original research contributions. I implemented the models and designed the experiments. I conducted evaluations on nine tasks; Tao Lin performed evaluations of AG, SWAG, and mode connectivity. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my co-authors who assisted me in improving the draft.

Chapter 5 corresponds to the following publication:

Mengjie Zhao, Hinrich Schütze. *Discrete and Soft Prompting for Multilingual Models.* In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8547-8555. 2021.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 6 corresponds to the following publication:

Mengjie Zhao*, Yi Zhu*, Ehsan Shareghi, Ivan Vulić, Roi Reichart, Anna Korhonen, Hinrich Schütze. *A Closer Look at Few-shot Crosslingual Transfer: The Choice of Shots Matters.* In Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL) (*equal contribution), pp. 5751–5767. 2021.

I conceived of the original research contributions. I implemented the models and designed the experiments. I conducted experiments for identifying and analyzing the variance issue, as well as testing the importance of lexical features and different adaptation methods. Yi Zhu performed evaluations on part of the 40 languages and the analysis of language features in Table 3. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my co-authors who assisted me in improving the draft.

Chapter 7 corresponds to the following publication:

Mengjie Zhao, Fei Mi, Yasheng Wang, Minglei Li, Xin Jiang, Qun Liu, Hinrich Schütze. *LMTurk: Few-Shot Learners as Crowdsourcing Workers in a Language-Model-as-a-Service Framework.* In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Findings. 2022.

I conceived of the original research contributions. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my co-authors who assisted me in improving the draft.

München, den 02. Januar 2022

Mengjie Zhao

Contents

Chapter 1

Introduction

Representing raw text data in a format that a computational model can work with has always been a big challenge in machine learning.

The invention of large-scale pretrained language models (PLMs) such as GPT (Radford et al., 2019b), BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), and T5 (Raffel et al., 2020) has largely advanced the field of natural language processing (NLP) in the past few years.

Equipped with hundreds of millions of parameters and trained on massive amounts of text corpora via self-supervised learning, PLMs effectively capture syntactic, semantic, commonsense, and factual knowledge in languages, providing informative representations for downstream NLP tasks. The learned representations are highly transferable: Straightforward transfer learning methods are already capable of achieving state-of-the-art (StoA) performance on numerous natural language understanding and generation tasks. It is almost a consensus that most of the modern NLP systems are built upon the PLMs, instead of training neural networks from scratch (Qiu et al., 2020; Bommasani et al., 2021).

Together with the outstanding task performances, energy consumption and environmental impacts imposed by PLMs become increasingly significant and can no longer be overlooked (Strubell et al., 2019; Schwartz et al., 2020). For instance, self-supervised pretraining of the base version of BERT yields 1,438 lbs carbon dioxide emissions, which is roughly equivalent to a trans-American flight (Strubell et al., 2019). Similarly, transfer learning with PLMs is not efficient either. It takes a large amount of disk memory to store the PLM checkpoints, e.g., the large version of BERT has checkpoints taking 1.34GB of disk memory (Devlin et al., 2019); T5-XXL (Raffel et al., 2020) checkpoints have size 41.5GB. In even worse cases, this number can grow linearly with the number of downstream tasks to be solved. One aspect of this dissertation studies the contextualization mechanisms of PLMs and shows that transfer learning brings relatively small changes to the pretrained weights. We then introduce a method addressing the memory efficiency

issue of applying transfer learning with PLMs.

The ultimate goal of artificial intelligence is to create human-like computational agents. One important aspect of human learning is few-shot learning: Unlike current computational models, humans require only a handful of training examples to conduct a task reasonably well. In other words, humans are more label-efficient than current computational models. Recently, it has been shown that when scaling up the number of parameters in the PLMs to 175 billion or leveraging gradient descent, PLMs can be converted to effective few-shot language learners when using priming (Brown et al., 2020) or prompting (Schick and Schütze, 2020). This dissertation reveals how label-efficient current PLMs are, by evaluating the few-shot learning ability of PLMs in a specific transfer learning scenario: Few-shot crosslingual transfer. We show that prompting outperforms straightforward transfer learning methods like finetuning in few-shot crosslingual transfer with natural language inference. Furthermore, we show that the performance of few-shot crosslingual transfer with PLMs sees large variances, depending on the choice of the few shots of training examples. Lastly, we discuss practical usages of few-shot learners built upon PLMs. We propose to consider the few-shot learners as crowdsourcing workers that annotate data for a downstream task. As a result, small and efficient machine learning models can be trained to solve the task in practical scenarios.

In this chapter, we briefly introduce representation learning for NLP and the backbone neural architecture utilized in most state-of-the-art representation learners: Transformer (Vaswani et al., 2017). Next, we introduce two main transfer learning schemes in NLP: Feature- and parameter-based transfer learning. After that, we introduce more parameter- and label-efficient transfer learning methods. We close this chapter by summarizing the contributions of this dissertation.

1.1 Distributed Representations

Building accurate machine learning models relies on abstract and informative data representations. Indeed, learning high-quality representations is now a central research topic of deep learning (Bengio et al., 2013).

The low-dimensional dense vectors learned from end-to-end training neural networks often show better quality and generalization ability than features designed manually according to human prior knowledge (LeCun et al., 2015). For example in computer vision, modern object detection systems (Bochkovskiy et al., 2020) largely rely on distributed representations extracted from ImageNet (Deng et al., 2009) pretraining rather than the histogram of oriented gradients feature (Dalal and Triggs, 2005).

In contrast to computer vision and speech processing (Purwins et al., 2019)

happy	car	suit
glad	vehicle	suits
pleased	cars	lawsuit
ecstatic	SUV	Suit
overjoyed	minivan	lawsuits

Table 1.1: Nearest neighbors measured by cosine similarity of three querying words in the Google News embedding space (Mikolov et al., 2013a).

in which data is expressed as real-valued signals, natural language is discrete and symbolic (Goldberg, 2017) such that the text corpus cannot be leveraged as inputs to neural networks directly. One straightforward method is to represent words in the text corpus with one-hot vectors (Hinton, 1984; LeCun et al., 2015): $\mathbf{w} \in S^{|\mathcal{V}|}$ where $S = \{0, 1\}$, $\mathbf{w}^\intercal \mathbf{1} = 1$, and $|\mathcal{V}|$ is the size of the corpus vocabulary $\mathcal{V}$. We can easily identify two immediate drawbacks: (i) $\forall\ \mathbf{w}_i, \mathbf{w}_j, i \neq j$, we have $\mathbf{w}_i^\intercal \mathbf{w}_j = 0$; (ii) $|\mathcal{V}|$ is a fixed large number, e.g., 1 million for the 6 billion Google News Corpus (Mikolov et al., 2013a). Point (i) implies that all words in the text corpus are orthogonal with each other, which is an undesirable property when working on languages. For example, we would like to have synonyms (e.g., "happy" and "glad") have high similarities while unrelated word pairs (e.g., "happy" and "mass") should be orthogonal. Point (ii) implies that directly processing such high-dimensional vectors is inefficient; this method cannot properly handle new words not shown in the corpus vocabulary $\mathcal{V}$ either.

Distributed representations are proposed to better capture the syntactic and semantic regularities of languages (Deerwester et al., 1990; Schütze, 1992; Bengio et al., 2003; Collobert et al., 2011a; Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014; Levy and Goldberg, 2014). In contrast to one-hot representations, each word is now associated with a low-dimensional real-valued vector representation. The structure of these representations encodes desired linguistic regularities of languages. For example, synonyms often have high cosine similarity (cf., Table 1.1); syntactic information about the language can also be captured (Andreas and Klein, 2014). In addition, these representations also show interesting linear structural properties: $\mathbf{w}$ ("China") $-$ $\mathbf{w}$ ("Beijing") $+$ $\mathbf{w}$ ("Tokyo") $=$ $\mathbf{w}$ ("Japan") (Mikolov et al., 2013c). In Section 1.2.1, we describe in detail widely adopted training algorithms to obtain these distributed representations.

Example "suit" in Table 1.1 is evidence that different senses of a polysemous word are crammed into a single *static vector*. Though a single vector has sufficient capacity to encode different meanings of a word (Yaghoobzadeh and Schütze, 2016), syntactic and semantic word uses can vary across different contexts (Peters et al., 2018). Thus, we prefer representations sensitive to the contextual information of words. In Section 1.2.2 we introduce state-of-the-art distributed represen-

tation learners for NLP, which largely take the context into consideration when creating and utilizing representations for languages.

1.1.1 Symbolic Unit

Compositionality is an evident property of languages. For example, characters are composed into words which then form sentences. Though we use "word" as the basic symbolic unit in Table 1.1 and in previous descriptions, there is no canonical choice of the basic symbolic unit used for composing and representing natural language in text corpora. Figure 1.1 presents some common choices for English.

Despite its simplicity, using words as the basic symbolic unit has noticeable drawbacks: (i) the instructive character-level information within words is lost, e.g., the character sequence "region" in "Euroregion" and "regional"; (ii) for languages like Chinese and Japanese that do not explicitly mark boundaries of linguistic units, "words" need to be firstly identified by language-specific segmentation algorithms which could be brittle, e.g., in domain mismatch settings. It has been shown that using other linguistic units such as subwords, characters, or bytes also models language well, leading to high quality representations of texts. For instance, Schütze (1992); Wieting et al. (2016); Schütze (2017); Dufter et al. (2018a) learn distributed representations for character- or byte-ngrams. Al-Rfou et al. (2019) build character-level language models with Transformers (Vaswani et al., 2017), outperforming RNN (Werbos, 1990; Hochreiter and Schmidhuber, 1997) counterparts. Xue et al. (2021) create byte-level language models for more than 100 languages for solving multilingual NLP tasks.

With the rise of large-scale PLMs, subwords become the ubiquitously used basic symbolic unit for encoding texts. Devlin et al. (2019); Liu et al. (2020) use wordpiece (Wu et al., 2016) subwords; Raffel et al. (2020); Conneau et al. (2020) leverage sentencepiece (Kudo and Richardson, 2018) subwords. In Chapter 2, we investigate the first subword method applied in NLP: Byte-pair encoding (BPE; Sennrich et al. (2016)). We apply BPE in an language-agnostic way and transfer sentiment information from English to more than one thousand languages.

Learning representations directly for larger symbolic units, such as phrases (Yu and Dredze, 2015), sentences (Conneau et al., 2017; Le and Mikolov, 2014), and documents (Le and Mikolov, 2014) is also widely explored. However, we limit the scope of this dissertation to units smaller or equal to words.

Sentence:	Euroregion Tyrol-South Tyrol-Trentino is formed by regional authorities in Austria and Italy.
Words:	“Euroregion” “Tyrol-South” “Tyrol-Trentino” “is” “formed” “by” “regional” “authorities” …
Subwords:	“Euro” “##re” “##gion” “Ty” “##rol” “-“ “South” “Ty” “##rol” “-“ “Trent” “##ino” “is” …
Characters:	E u r o r e g i o n T y r o l - S o u t h T y r o l - T r e n t i n o i s f o r m e d b y r e g i …
Bytes:	69, 117, 114, 111, 114, 101, 103, 105, 111, 110, 32, 84, 121, 114, 111, 108, 45, 83, 111, 117 …

Figure 1.1: Different basic symbolic units for representing an English sentence. For subword tokenization we use the “bert-large-cased” tokenizer (Devlin et al., 2019). For bytes, we use UTF-8 encoding.

1.2 Pretrained Language Representations

This section introduces several representative methods of learning distributed representations for NLP. Two types of representations are covered: Static and contextualized representations. Static representation learners associate a single vector to each *word-type* in the corpus vocabulary $\mathcal{V}$; contextualized representations are context-sensitive – each *token* in the corpus is associated with a vector representation.

1.2.1 Static Representations

Methods of learning static word[1] representations can be categorized to count-based and predict-based methods. The output of these methods is a real-valued matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where $|\mathcal{V}|$ refers to the vocabulary size and d refers to the embedding dimension size.

Count-based Methods

Motivated by the distributional hypothesis (Harris, 1954), *count-based* methods for learning semantic representations have a long history in computational linguistics. The central idea is to capture the co-occurrences between words in the vocabulary $\mathcal{V}$ because “a word is characterized by the company it keeps” (Firth, 1957). The identification of word co-occurrence is often specified using the context window of a given word w_i, e.g., five words to its left and five words to its right are considered as co-occurring with w_i in the corpus. The co-occurrences are often maintained in a $|\mathcal{V}| \times |\mathcal{V}|$ matrix $\mathbf{C}$. $\mathbf{C}$ can be directly utilized as $\mathbf{E}$, but different transformations of the raw co-occurrence numbers such as point-wise mutual information (PMI) weighting (Church and Hanks, 1990) are introduced

[1] In this section of discussing static representations, we use “word” as the example basic symbolic unit for the simplicity of description.

for better representations. Factorizing $\mathbf{C}$ using singular value decomposition also produces representations encoding linguistic regularities (Schütze, 1992).

Predict-based Methods

Baroni et al. (2014) show that the *predict-based* methods (Bengio et al., 2003; Collobert et al., 2011b; Mikolov et al., 2013a,b) generate representations better capturing linguistic regularities than count-based methods. In contrast to count-based methods, $\mathbf{E}$ in predict-based methods is derived from modeling natural language using neural networks (Bengio et al., 2003) to maximize the likelihood of the training corpus $(w_1, w_2, w_3, ..., w_n)$ with objective:

$$P(w_1, w_2, w_3, ..., w_n) = \prod_{t=1}^{n} P(w_t | w_1, w_2, ..., w_{t-1}), \tag{1.1}$$

where w_t is the tth word in the corpus which contains n words in total. After training, the weights in the embedding layer lookup table of the neural network can be utilized as $\mathbf{E}$.

The language modeling objective is computationally expensive due to the softmax operation over the $|\mathcal{V}|$ possibilities of w_t; Bengio et al. (2003) invest a large amount of efforts improving the training efficiency. Similarly, it takes two months to train the SENNA embeddings (Collobert et al., 2011b). Mikolov et al. (2013a,b) introduced the *word2vec* package enclosing two methods for computing word representations *efficiently*: Continuous bag-of-words (CBOW) and Skip-gram. Instead of running the expensive language modeling objective, word2vec trains a shallow neural network predicting words in context windows (as in count-based methods). Considering a text corpus $(w_1, w_2, ..., w_t, ..., w_n)$, Figure 1.2 visualizes the two methods.

word2vec has stimulated a large wave of work on creating and analyzing representations for NLP. We next introduce in detail the training process of *Skip-gram with negative sampling*, the most well-known method for obtaining a static word representations $\mathbf{E}$.

Skip-gram with Negative Sampling

Recall that $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where $|\mathcal{V}|$ is the corpus vocabulary size and d is the dimentionality of the representation. Skip-gram with negative sampling firstly initializes two weight matrices $\mathbf{O}, \mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$. For each word w_t in the training corpus, its vector representations from $\mathbf{E}$ and $\mathbf{O}$ are denoted as $\mathbf{w}_{t,\mathbf{E}} \in \mathbb{R}^d$ and $\mathbf{w}_{t,\mathbf{O}} \in \mathbb{R}^d$ respectively. The training objective is to maximize the following log likelihood:

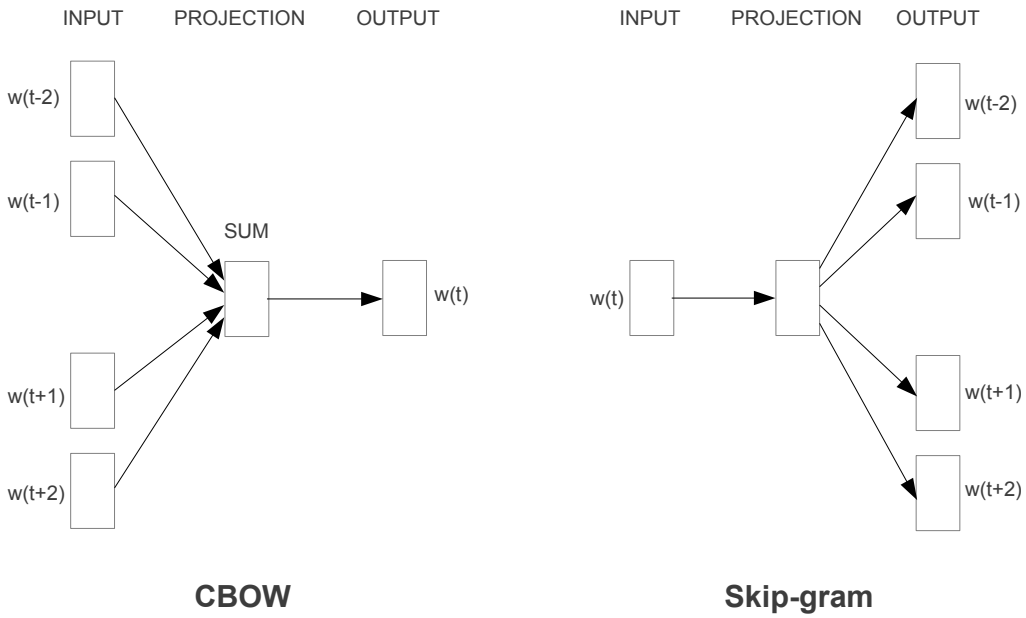


Figure 1.2: The two representation learning methods in word2vec: CBOW and Skip-gram. The figure is taken from (Mikolov et al., 2013a). In CBOW, the shallow neural network predicts the current word w_t based on the context. In Skip-gram, the network predicts context words given the current word w_t.

$$\sum_{w_c \in \mathcal{C}(w_t)} log(\sigma(\mathbf{w}_{t,\mathbf{E}}^{\intercal} \mathbf{w}_{c,\mathbf{O}})) \; - \sum_{w_i \in \mathcal{N}(w_t)} log(\sigma(\mathbf{w}_{t,\mathbf{E}}^{\intercal} \mathbf{w}_{i,\mathbf{O}})), \tag{1.2}$$

where $\mathcal{C}(w_t)$ is the group of context words for w_t, i.e., words within the context window of w_t; $\mathcal{N}(w_t)$ is the group of non-context words of w_t sampled from the whole corpus. $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. After training, $\mathbf{E}$ is utilized as the word representations and $\mathbf{O}$ is discarded. Clearly, the objective in Equation 1.2 is much more efficient than the objective of neural language modeling Equation 1.1: There is no need to predict w_t through the large softmax layer that has large size $|\mathcal{V}| \times d$.

word2vec generates high quality representations efficiently, and it has been widely adopted in NLP research. Several improved methods are also proposed. For example, ordering information among $(..., w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}, ...)$ in the context window is not considered in CBOW and Skip-gram; Ling et al. (2015) strengthen word2vec to be sensitive to word order.

In addition, the character-level information within each word w_t is not lever-

aged; this is sub-optimal for morphologically rich languages like Turkish and Finnish: Low word frequency could lead to inferior representations for rare words that have complex morphological inflections. *fastText* (Bojanowski et al., 2017) enriches word representations with subword information, outperforming word2vec in tasks like word similarity and analogy in different languages.

Prediction as Implicit Matrix Factorization

At first glance, count-based methods and predict-based methods seem to be two completely different types of ways for learning word representations – they are proposed and investigated in the computational linguistic community and the neural network community respectively. However, Levy and Goldberg (2014) attempt to prove that Skip-gram with negative sampling is implicitly factorizing a word-context matrix, connecting these two types of methods. Although the accuracy of the proof has been questioned, this work establishes a clear connection between the two approaches.

Overall, static distributed representations have a long track history in NLP; the invention of word2vec has stimulated a large wave of research learning high quality representations for English (Pennington et al., 2014) and multiple languages (Faruqui and Dyer, 2014; Xing et al., 2015; Artetxe et al., 2016; Ammar et al., 2016; Dufter et al., 2018a,b). However, the drawback of static representations $\mathbf{E}$ is clearly significant: They are not sensitive to the actual contextual information of a word w when using its word representation $\mathbf{w}$. We next introduce contextualized representations that take contextual information into consideration and often achieve state-of-the-art performance in numerous NLP tasks.

1.2.2 Contextualized Representations

The static word representations $\mathbf{E}$ are not context-sensitive. For instance, to represent the polysemous word "suit", the same vector representation $\mathbf{w} \in \mathbb{R}^d$ will be used in the following diverse contexts:

- What time would *suit* you?

- A *suit* of armor.

- His lawyer filed a *suit* against Los Angeles city.

This is clearly sub-optimal because the contexts and usage of an ambiguous word can vary significantly. Thus, it is crucial to take the context of a word occurrence into consideration. An intuitive solution is the bag-of-word method. For example,

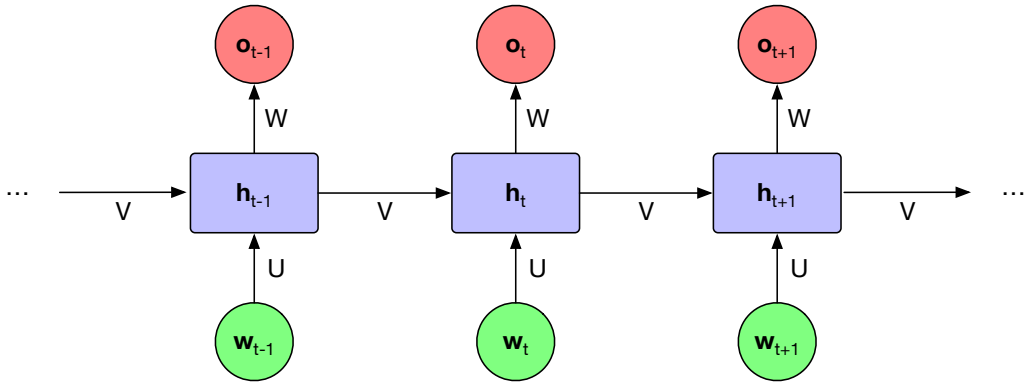


Figure 1.3: Recurrent neural network.

to represent the word "suit" in "A suit of armor.", we can simply average the embedding of all the words in the sentence. However, it has disadvantages: "armor" will have the same embedding as "suit", and averaging too many embeddings results in a vector not specific to a particular word anymore.

Contextualized representation models such as CoVe (McCann et al., 2017), context2vec (Melamud et al., 2016), ELMo (Embeddings from Language Models; Peters et al. (2018)), and BERT (Bidirectional Encoder Representations from Transformers; Devlin et al. (2019)) are proposed to address this issue. Next, we introduce two representative contextualized representation models: ELMo and BERT. We also introduce Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and Transformer which are the backbone neural network architectures of ELMo and BERT.

Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN; (Rumelhart et al., 1986)) widely adopted for processing sequential data such as text and speech. We show in Figure 1.3 an unfolded basic RNN. Given sequential data, e.g., a sentence $(w_1, w_2, ..., w_t, ..., w_n)$, for each word at time step t, we compute

$$\begin{aligned} \mathbf{a}_t &= \mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{w}_t, \\ \mathbf{h}_t &= \tanh(\mathbf{a}_t), \\ \mathbf{o}_t &= \mathbf{c} + \mathbf{V}\mathbf{h}_t, \end{aligned} \tag{1.3}$$

where $\mathbf{b}$, $\mathbf{c}$, $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{U}$ are RNN trainable parameters. $\mathbf{w}_t$ is the embedding vector of w_t. $\tanh$ is the hyperbolic tangent function. Basic RNN in practice suffers from the gradient vanishing problem when modeling long sequences. In such

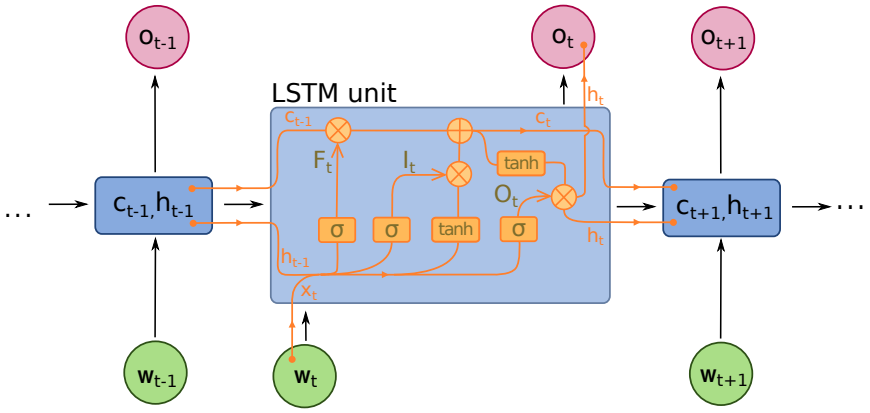


Figure 1.4: LSTM unit. Image from Wikipedia (Wikipedia contributors, 2021).

a case, LSTMs can be leveraged. For the sequence $(w_1, w_2, w_3, ..., w_t, ..., w_n)$, at each time step an LSTM computes

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{w}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{w}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{w}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \\
\tilde{\mathbf{c}}_t &= \sigma_h(\mathbf{W}_c \mathbf{w}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t, \\
\mathbf{h}_t &= \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t),
\end{aligned} \tag{1.4}$$

where $\circ$ refers to the Hadamard product. Initial cell state $\mathbf{c}_0$ and hidden state $\mathbf{h}_0$ are set to $\mathbf{0}$. $\mathbf{W}_x, \mathbf{U}_x, \mathbf{b}_x, x \in \{f, i, o\}$ are the trainable weights and biases for the forget, input, and output gates. $\mathbf{w}_t$ is the word representation of word w_t in the embedding layer. $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = 1/(1+e^{-x})$ and $\sigma_h(\cdot)$ is the hyperbolic tangent function $\sigma_h(x) = (e^x - e^{-x})/(e^x + e^{-x})$. Figure 1.4 visualizes the LSTM unit. Through the usage of gating mechanisms and cell state $\mathbf{c}$ at step t, the LSTM unit decides the information that needs to be taken as input, and the memory that needs to be forgotten. Thus, an LSTM can model the sequential relations in particularly long sequences (Khandelwal et al., 2018).

Deep Contextualized Word Representations (ELMo)

ELMo (Peters et al., 2018) generates embedding vectors that are sensitive to the word use. Its training objective is similar to predict-based representation learning methods in Section 1.2.1: Modeling the languages with an LSTM. After training, the LSTM outputs, or a combination (e.g., a weighted sum) of outputs from

different LSTM layers, are used to represent the text, instead of simply using the embedding layer weights $\mathbf{E}$ as in methods like word2vec. ELMo outperforms its static counterparts like GloVe in several tasks including question answering, textual entailment, and sentiment analysis (Peters et al., 2018).

One limitation of ELMo is that its neural architecture is still very shallow: Only two LSTM layers are utilized. However, it has been shown that neural network depth is of crucial importance for learning high quality representations of data (Simonyan and Zisserman, 2014; Szegedy et al., 2015). The LSTM architecture is the bottleneck in ELMo. As can be observed in Equation 1.4, the amount of forward pass computation at each time step t is significant; paralleling the computation on different GPUs is non-trivial due to the sequential dependency between the time steps. These disadvantages pose difficulties to scaling up depth of neural network architectures for learning high quality representations of texts.

Several RNN and LSTM variants like GRU and SRU are proposed to address the efficiency issue (Cho et al., 2014; Lei et al., 2018), but the autoregressive property when modeling sequences with an LSTM is still present. In contrast, Vaswani et al. (2017) introduce the *Transformer* architecture which drops the inductive bias of sequential order in languages, only relying on a self attention mechanism and position encoding to model the sequential nature. We introduce in detail this architecture because it has been widely adopted in state-of-the-art representation learners for NLP (Devlin et al., 2019; Yang et al., 2019; Raffel et al., 2020; Brown et al., 2020), computer vision (Dosovitskiy et al., 2020), and speech processing (Dong et al., 2018; Baevski et al., 2020).

Transformer

Figure 1.5 left illustrates the Transformer architecture, which was firstly applied in neural machine translation (NMT). We limit our scope to the encoder block since it is the main focus of this dissertation. "Add" and "Norm" refer to the skip-connection (He et al., 2016) and layer normalization (Ba et al., 2016) operation respectively. In practice, the encoder block is repeated N times, e.g., N is set to 6 or 8 in (Vaswani et al., 2017), leading to very deep neural networks.

One prominent property of the Transformer architecture is that the encoder processes the elements in a sentence in parallel, instead of in an autoregressive manner as RNNs do. As a result, the Transformer needs to inject positional information into each element in the sentence. Vaswani et al. (2017) use fixed *position encodings* (Gehring et al., 2017) in the following form:

$$\begin{aligned} \mathbf{PE}_{(pos,2i)} &= \sin(pos/10000^{2i/d}), \\ \mathbf{PE}_{(pos,2i+1)} &= \cos(pos/10000^{2i/d}), \end{aligned} \tag{1.5}$$

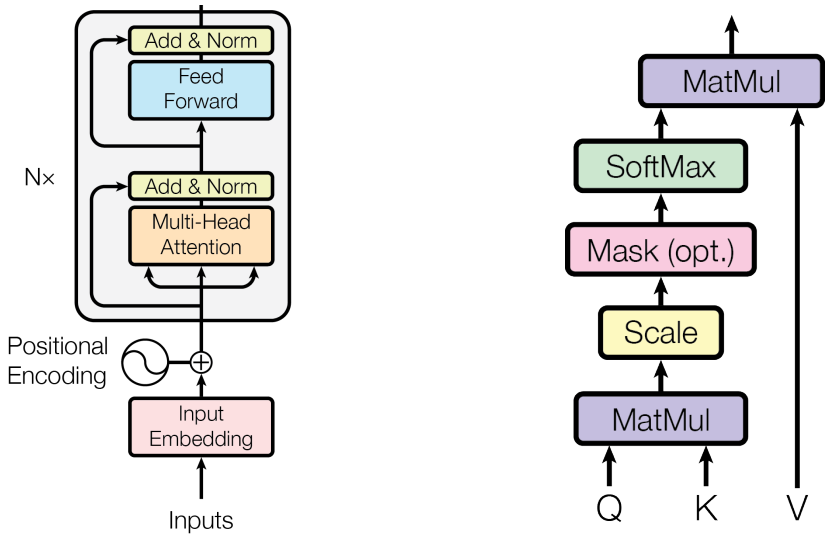


Figure 1.5: The encoder architecture of the Transformer (left) and the self attention operation (right). Figure taken from (Vaswani et al., 2017).

where $\mathbf{PE} \in \mathbb{R}^{T \times d}$, T is the maximum sequence length (512) and d refers to the hidden dimension size. pos refers to the position in the sentence and i is the hidden dimension index. So each dimension of the position encoding corresponds to a sinusoid. It can be observed that Equation 1.5 only introduces absolute positional information; Shaw et al. (2018) propose to enclose relative positional information, further improving task performance of Transformers.

Figure 1.5 right demonstrates in detail the self attention mechanism conducted in every "Multi-Head Attention" cell of each encoder block. Specifically, three linear layers[2] $\mathbf{W}_K$, $\mathbf{W}_Q$, $\mathbf{W}_V$ for computing the self attention among input embeddings of the wordpieces (Wu et al., 2016) are firstly initialized. Next, for an input sentence $\mathbf{X} \in \mathbb{R}^{T \times d}$ where T is the maximum sentence length and d is the hidden dimension size, $\mathbf{W}_K$, $\mathbf{W}_Q$, and $\mathbf{W}_V$ are used to compute transformations of $\mathbf{X}$:

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{V} = \mathbf{X}\mathbf{W}_V, \tag{1.6}$$

and the self attention of $\mathbf{X}$ is then computed as (Figure 1.5 right):

$$\text{Attention}(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d}})\mathbf{V}. \tag{1.7}$$

[2]We omit the bias terms for brevity.

Next, the attention is transformed by another linear layer and then fed forward to the next encoder block.

Equation 1.7 implies that there are no explicit dependencies among the subwords in a sentence $(w_1, w_2, w_3, ..., w_t, ..., w_n)$. For example, $\mathbf{w}_1$ now can directly observe and interact with the future $\mathbf{w}_n$ during training. This is not feasible in recurrent models like LSTMs due to their autoregressive nature. The Transformer architecture has been widely integrated in current NLP systems: It is the top performer in NMT systems (Barrault et al., 2020); most state-of-the-art contextualized representation learners also use it as the backbone (Devlin et al., 2019; Yang et al., 2019; Raffel et al., 2020).

We next introduce in detail one of the most influential contextualized representation learners: BERT (Bidirectional Encoder Representations from Transformers; Devlin et al. (2019)).

Bidirectional Encoder Representations from Transformers

BERT is the first Transformer-based pretrained language model providing high quality contextualized text representations. The BERT encoder consists of 12 to 24 Transformer blocks containing 110 million to 340 million of floating point parameters. The pretraining data consists of ≈3300 million words, and is extracted from the BooksCorpus (Zhu et al., 2015) and English Wikipedia. There are two pretraining objectives: Masked language modeling (MLM) and next sentence prediction (NSP).

MLM. Considering a WordPiece-tokenized corpus $(w_1, w_2, w_3, ..., w_t, ..., w_n)$ in which each w_i is a subword. MLM aims to reconstruct a corrupted version of the corpus, $(w_1, w_2, w_3, ...,$[MASK]$, ..., w_n)$, to the original one. Concretely, the corruption is conducted by randomly replacing 15% of the subwords in the corpus with a special [MASK] token. The encoder then takes as input the corrupted corpus $(w_1, w_2, w_3, ...,$[MASK]$, ..., w_n)$ to predict w_t with cross-entropy loss. The MLM objective is inherently different from standard language modeling object as shown in Equation 1.1, because it leverages the bi-directional contextual information of w_t.

NSP. Many NLP tasks, e.g., natural language inference, require modeling the relationship between sentence pairs. The NSP objective is utilized to capture this information during pretraining BERT. Specifically, BERT takes a pair of sentences and determines whether the two sentences occur in the corpus consecutively (i.e., next to each other) with the standard cross-entropy training objective.

For many NLP tasks, BERT significantly outperforms static text representations like word2vec, as well as contextualized representations derived from other models like ELMo. The invention of BERT has stimulated the development of various alternative models such as RoBERTa (Liu et al., 2020), XLNet (Yang

et al., 2019), and T5 (Raffel et al., 2020).

Large-scale PLMs for multiple languages are also proposed, including mBERT (Devlin et al., 2019), XLM (Conneau and Lample, 2019), and XLM-R (Conneau et al., 2020). In this dissertation, we focus on BERT for English tasks and XLM-R for multilingual tasks. Similar to BERT, XLM-R leverages the Transformer architecture and utilizes MLM as training objective. The training data for XLM-R is a 2.5TB text corpus containing 100 languages. Though there is no explicit supervision aligning different languages when pretraining XLM-R, it performs strongly in crosslingual tasks like crosslingual natural language inference (Conneau et al., 2018b, 2020).

1.2.3 Summary

This section reviews the development of representation learning for NLP: From high-dimensional sparse vectors to low-dimensional dense distributed representations. Representative methods for learning static and contextualized text representations – word2vec and BERT – are also introduced. We put emphasis on contextualized representations because they encode richer syntactic and semantic regularities of languages than their static counterparts, and achieve state-of-the-art performance on NLP tasks. Properly transferring these informative features captured during pretraining to downstream NLP tasks is of vital importance and we focus on *transfer learning* in the next section.

1.3 Transfer Learning

For an NLP task, traditional supervised learning assumes training and testing data are drawn from the same data generation process. For task *A*, a common scenario is to assume that there is sufficient labeled data for training a machine learning model; the model is then evaluated on the test data. When a new task *B* arrives, traditional supervised learning conducts identical procedures.

In contrast, discrepancies and mismatches among the data distributions are allowed in transfer learning (Pan and Yang, 2010). Concretely, transfer learning is defined as the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks (Pan and Yang, 2010). For example, transfer learning could enable a classifier trained to distinguish positive and negative product reviews to classify movie reviews with a reasonable accuracy.

Figure 1.6 compares traditional supervised learning and transfer learning. Traditional supervised learning trains a system for a task from scratch using the task dataset. For tasks that have a sufficient amount of data, i.e., A and B, the systems often have high task performance. However, for tasks that have very limited

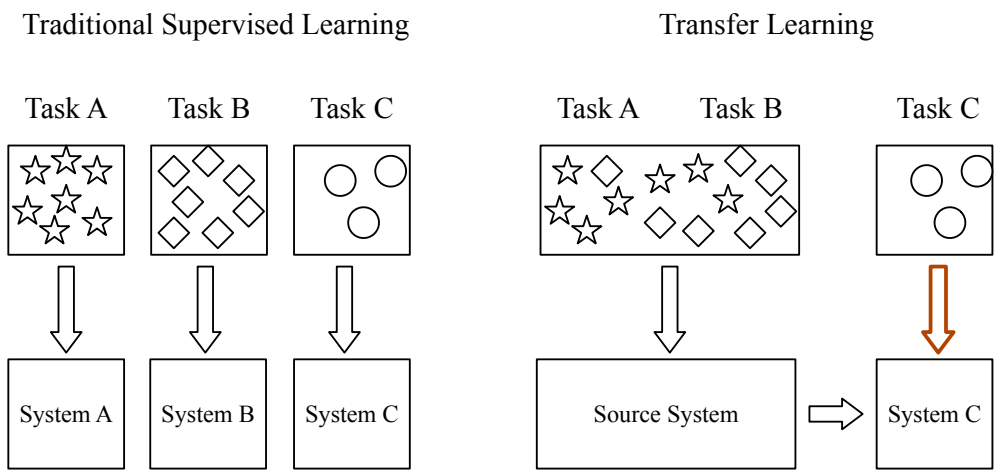


Figure 1.6: Comparing traditional machine learning (left) and transfer learning (right). For each task, traditional supervised learning trains a system with the task dataset only. The system may have inferior performance when the amount of task data is small, i.e., task C. Transfer learning leverages the knowledge from similar tasks, i.e., task A and B, to facilitate the data scarcity problem of task C.

amount of data, i.e., task C, the corresponding system may not perform well due to data scarcity. Transfer learning leverages the knowledge obtained from previous tasks to tackle the data scarcity issue hence better solves a new task, i.e., task C. In this dissertation, we restrict our scope to *sequential transfer learning* (Ruder, 2019). That is, we focus on sequentially transferring the learned representations from source tasks (A and B) to the target task (C).

Making effective use of the pretrained language representations introduced in Section 1.2 is now a central topic in NLP. This problem falls within the range of transfer learning. Concretely, the MLM and NSP objectives of *pretraining* BERT can be considered as Task A and B shown in Figure 1.6, and the target of *adaptation*, i.e., an NLP downstream task can be viewed as task C. The large-scale PLMs have been shown to encode syntactic (Goldberg, 2019), semantic (Tenney et al., 2019), and factual knowledge (Petroni et al., 2019) in the languages; effective and efficient methods of transferring this knowledge to various downstream NLP tasks need to be devised. In addition, developing effective methods of transferring knowledge from high resource languages like English to low resource languages is also a crucial research question. We introduce commonly adopted transfer learning methods in the current NLP literature. Somehow surprisingly, these methods are conceptually simple, evidencing that the informative features about languages encoded in modern large-scale PLMs are highly transferable.

1.3.1 Feature-Based Transfer

Feature-based transfer learning keeps the pretrained weights in language models unchanged. When solving an NLP task, the task data texts are input fed to a PLM; the output representations extracted by the PLM are then fed to task-specific neural architectures for solving the task, e.g., a conditional random field layer for named-entity recognition (Lafferty et al., 2001).

Feature-based transfer learning with PLMs has two major strengths. First, because the PLM weights do not need to be updated, we can extract and save the features of task data by running one single forward pass over the large PLM. This largely reduces computational cost. Second, more flexibility is achieved because a task-specific neural architecture can be integrated, instead of relying on the PLM architecture.

One representative feature-based transfer learning scenario is the utilization of ELMo (Peters et al., 2018) to downstream NLP tasks. Peters et al. (2019) show that a linear combination of features extracted by different ELMo layers performs strongly in tasks like MNLI (Williams et al., 2018) and semantic textual similarity (Marelli et al., 2014).

Another representative usage of feature-based transfer learning is **probing** (Conneau et al., 2018a; Rogers et al., 2020). Probing is a technique that has been widely applied for understanding and interpreting NLP models. In probing, diagnostic classifiers, e.g., linear classifiers, are trained to "probe" the representations for desired properties such as linguistic or structural information. Concretely, simple classifiers, e.g., a logistic regression classifier, taking as input the learned text representations, attempt to answer the question: Do the input representations encode desired properties like syntactic information or not? If the classifier achieves high performance, we could then conclude that rich features about syntax are present in the learned representations. Because the PLM weights remain unchanged, probing can be viewed as a feature-based transfer learning method.

Two important requirements need to be satisfied for successful probing experiments. First, dedicated datasets for training and evaluating the probing classifiers need to be carefully constructed. Second, the probing classifier itself needs to have low capacity, e.g., a linear classifier. Leveraging strong classifiers yields near perfect performance on the probing dataset, but the results confound contributions of the classifier capacity and of the quality of the pretrained representations input to the classifier (Hewitt and Liang, 2019).

In Chapter 3, we leverage probing to understand how BERT progressively contextualizes words, from the non-contextualized embeddings in the lowest embedding layer to the highest self attention layers. For an ambiguous word, we also quantify the number of surrounding contextual words needed to accurately under-

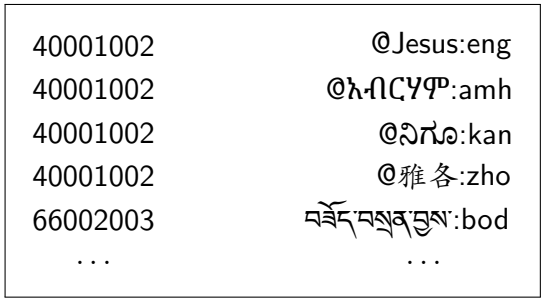


Figure 1.7: Samples of sentenceID embedding training corpus. 40001002 and 66002003 are the IDs of Bible verses. Language abbreviations are: amh: Amharic; kan: Kannada; zho: Chinese; bod: Tibetan.

stand the ambiguous word. In addition, we show that the contextualization ability of a PLM does not change significantly, after adapting the PLM to downstream tasks. This inspires us to devise more efficient transfer learning methods, which we discuss in Section 1.4.

1.3.2 Finetuning

Proposed by Devlin et al. (2019), finetuning is another major method transferring the pretrained representations into downstream NLP tasks. In contrast to the feature-based transfer learning method that preserves PLM parameters, finetuning considers the pretrained PLM parameters as an initialization, and then updates them with gradient descent during the adaptation to downstream tasks. Compared with feature-based transfer learning, finetuning has clear disadvantages. For example, it is more computationally expensive because the large amount of PLM parameters, e.g., 110M in BERT-base, need to be updated. This disadvantage becomes even worse when there are several tasks to be solved – we need to train and save several large model checkpoints respectively. In addition, it is difficult to incorporate extra task-specific neural network architectures, because the number of PLM parameters is already large.

Despite these two disadvantages, finetuning is still by far the most common way of conducting transfer learning with PLMs due to its superior task performance (Howard and Ruder, 2018). For example, Peters et al. (2019) show that finetuning BERT outperforms ELMo with feature-based transfer learning when source and target tasks are similar. In Chapter 5, 6, and 7, we conduct transfer learning with finetuning. In Chapter 4, we introduce *masking*, a more parameter-efficient alternative to plain finetuning for transfer learning.

1.3.3 Crosslingual Transfer

Crosslingual transfer can be viewed as a special case of transfer learning: A, B, and C in Figure 1.6 now refer to different languages when solving the same task like sentiment analysis. C is commonly a low resource language which does not have sufficient data for training a high quality machine learning model. The goal of crosslingual transfer is to improve task performance in language C, by successfully transferring the knowledge and representations learned from A and B, which have sufficient task data.

Crosslingual transfer is critical for NLP because most of the ≈7000 languages in the world are low-resource; the fast development of language technologies only focuses on a few of them like English and Chinese (Joshi et al., 2020). A high quality semantic space accommodating different languages is one of the important building blocks for successful crosslingual transfer (Ruder et al., 2019).

In this section, we introduce sentenceID (Levy et al., 2017) as a method of training static multilingual embeddings. Next, we introduce the process of training contextualized multilingual representation models, which achieve state-of-the-art performance on crosslingual tasks.

- **sentenceID**. When creating *static multilingual embeddings*, sentenceID assumes a sentence-aligned parallel corpus such as Europarl (Koehn, 2005) and the Parallel Bible Corpus (PBC; Mayer and Cysouw (2014)). Next, sentenceID reformulates the sentence-aligned corpus by counting co-occurrences between the ID of a sentence and the sentence's words in all languages. In Figure 1.7 we show samples of the reformulated corpus with PBC. After that, the corpus is fed to word2vec to obtain a multilingual embedding space.

  Despite its simplicity, sentenceID is a strong baseline generating high quality multilingual embeddings, performing strongly in word alignment and dictionary induction (Levy et al., 2017). We leverage sentenceID in Chapter 2 to transfer sentiment information from English to more than one thousand languages.

- **Multilingual BERT**. Somehow surprisingly, creating *contextualized multilingual representation models* that achieve state-of-the-art performance on crosslingual transfer is conceptually simple. For example, the training procedures of multilingual BERT (mBERT) are similar to that of the English BERT (Devlin et al., 2019). Concretely, we train a large Transformer architecture on a large amount of text data using masked language modeling and next sentence prediction objectives (c.f., Section 1.2.2). There are two main differences. First, the training data for mBERT is a concatenation of

Wikipedia dumps in 104 languages. Second, subwords in the vocabulary are shared across different languages. For example, *chat* is used in both English context and French (*cat*) contexts. Other contextualized multilingual representation models like XLM-R follow similar training configurations.

Despite the fact that there is no explicit supervision for aligning different languages, these models are state-of-the-art multilingual representation learners for crosslingual transfer. Understanding rationales behind the effectiveness of contextualized multilingual models is a central topic in multilingual NLP (Dufter and Schütze, 2020; Artetxe et al., 2020; K et al., 2020).

- **Crosslingual Transfer Evaluation**. The next step is to improve task performance of a low-resource target language C, by transferring task knowledge learned from one or more source languages; the multilingual embeddings act as an intermediate supporting the transfer. In this dissertation, we employ a single language – typically English due to its abundant resource status – as our source language, and evaluate task performance of the target language C in two scenarios (see below). Note that in both scenarios we use finetuning as our transfer learning method.

- **Zero-shot transfer**. Proposed by Pires et al. (2019), zero-shot transfer has been widely adopted to evaluate representation learning models for crosslingual transfer learning. In zero-shot transfer, the model is first finetuned with task-specific supervised training data from the source language (English); the task is then directly evaluated with data in the target language (C). Zero-shot crosslingual transfer enables us to observe how the model performs and generalizes across different languages.

  In Chapter 5, we leverage zero-shot transfer to investigate the effectiveness of prompting (c.f., Section 1.4.2) on crosslingual natural language inference.

- **Few-shot transfer** is a recently proposed crosslingual transfer learning paradigm that relaxes the constraint that no training data is available for the target language C as in zero-shot transfer. It assumes the availability of a few shots of data in C, such that practitioners can continue finetune the model obtained during the source-training stage using the small amount of data in C. Lauscher et al. (2020) show that, despite its small size, the few-shot data in C significantly improve performance of C in tasks like dependency parsing and named-entity recognition. Few-shot transfer is more practical and realistic than zero-shot transfer, because annotating a handful of examples is actually not expensive: Garrette and Baldridge (2013) show that it is possible to collect $\approx$100 part-of-speech-tagged sentences in two hours even for

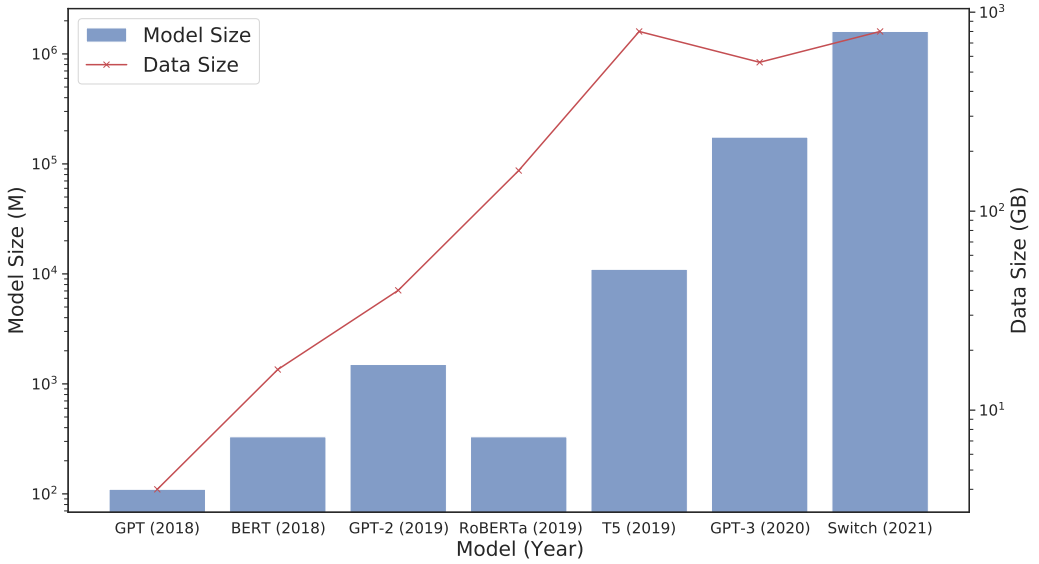


Figure 1.8: Number of parameters (million) of PLMs[3] and their pretraining dataset size (GB). The number of parameters in PLMs keeps growing. Image is from Han et al. (2021).

low-resource languages such as Malagasy.

In Chapter 6 we investigate few-shot crosslingual transfer with extensive experiments, showing that its strength over the zero-shot counterpart largely depends on the choice of the few-shot data in C.

1.3.4 Summary

This section briefly introduces the basic transfer learning paradigm, with an emphasis on sequential transfer learning. We introduce two widely adopted strategies of transferring knowledge from a source task to a target task when using PLMs: Feature-based transfer and finetuning. We also describe zero- and few-shot crosslingual transfer; multilingual representations play a central role in these scenarios. Next, we introduce methods of improving parameter- and label-efficiency of plain transfer learning.

1.4 Efficient Transfer Learning

This section introduces several strategies improving plain transfer learning from two perspectives: Parameter-efficiency and label-efficiency, which are the main focus of this dissertation.

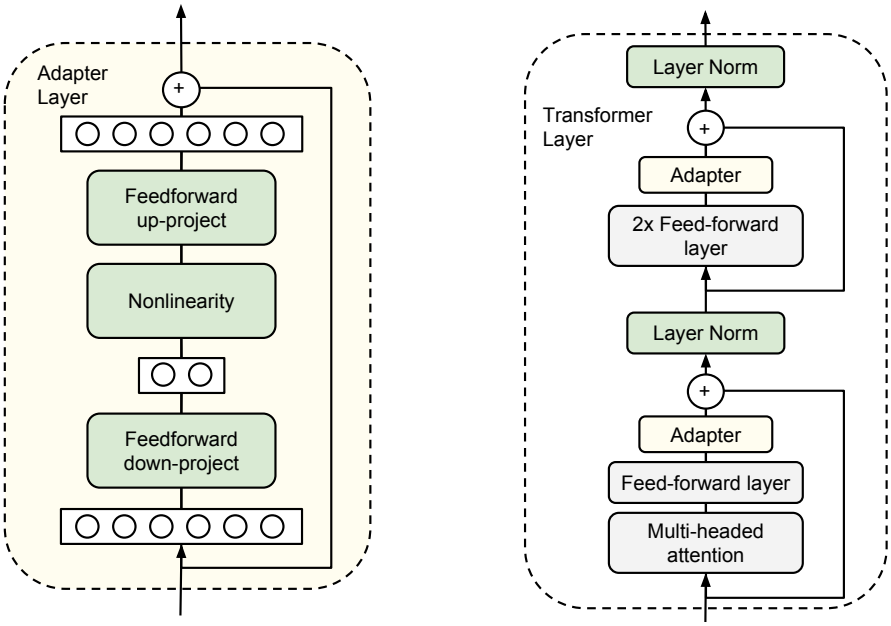


Figure 1.9: An adapter layer consists of a small amount of trainable parameters (left); it is inserted into the Transformer layer. Images are from Houlsby et al. (2019).

1.4.1 Parameter-Efficient Transfer

The scale of PLMs has increased dramatically in the past few years, and the number continues to grow. In Figure 1.8 we visualize the number of parameters and pretraining dataset size of an array of PLMs.

The large amount of parameters poses a challenge when carrying out transfer learning with PLMs. For example, each of the target tasks or languages requires saving a large model checkpoint, e.g., 41.5GB when using T5-XXL (Raffel et al., 2020). This becomes even worse when more than one target task or language need to be processed because the cost increases linearly. To address this issue, several parameter-efficiency methods of conducting transfer learning with PLMs are devised; we briefly review two methods.

Adapter. Pioneered by Rebuffi et al. (2017) and extended by Houlsby et al. (2019) and Stickland and Murray (2019), the adapter has been widely used in NLP. An adapter layer consists of a small amount of trainable parameters as shown in Figure 1.9; it is inserted into a standard Transformer layer. When adapting the PLM into a target task, we only update the parameters in adapter layers. As a result, high parameter-efficiency is achieved because the large PLM is frozen dur-

[3]The PLMs are GPT (Radford et al., 2018), BERT (Devlin et al., 2019), GPT2 (Radford et al., 2019a), RoBERTa (Liu et al., 2020), T5 (Raffel et al., 2020), GPT3 (Brown et al., 2020), and Switch Transformers (Fedus et al., 2021).

ing training such that it can be reused across different tasks; the only parameters that need to be saved are the parameters in the adapter layer. Adapter-based tuning achieves comparable performance to plain finetuning on different natural language understanding and generation tasks (Houlsby et al., 2019; Stickland and Murray, 2019).

Prefix/Prompt tuning. Li and Liang (2021) introduce prefix-tuning, a method that is more parameter-efficient than plain finetuning for transfer learning with large PLMs. Similar to adapter-based tuning, prefix-tuning also freezes PLM parameters. During training, a small amount of task-specific trainable vectors are prepended to the input sequence in all Transformer layers; they are the only trainable parameters. The model is then trained on data of the downstream task. Li and Liang (2021) show that prefix-tuning achieves comparable performance to plain finetuning in tasks like table-to-text generation and summarization. Prompt-tuning (Lester et al., 2021) is a variant of prefix-tuning that only adds the pseudo tokens in the embedding layer; it achieves comparable performance to plain finetuning when scaling up the number of parameters in PLMs.

In Chapter 4, we introduce *masking* which is a parameter-efficient alternative to plain finetuning. Masking learns selective binary masks for the PLM parameters in lieu of modifying them as plain finetuning does. Masking achieves comparable performance to plain finetuning but is much more parameter-efficient because we only need to save a set of binary masks for each of the downstream tasks.

1.4.2 Label-Efficient Transfer

Unlike human learning, current computational models still require a large amount of annotated data to conduct an NLP task (Yin et al., 2020). Methods of converting PLMs into few-shot learners, i.e., models that require only a handful of annotations, are also devised; we introduce priming (Brown et al., 2020) and prompting (Schick and Schütze, 2020; Gao et al., 2020; Liu et al., 2021).

Priming (or in-context learning) is a method of utilizing GPT3 (Brown et al., 2020), a PLM that contains 175 billion parameters. To process an NLP task with GPT-3, priming prepends a few examples (i.e., the few shots) demonstrating the task before the actual data input, and then converts the data into a cloze-style query. Next, the reformulated text is input to the large PLM to get a prediction. Taking sentiment analysis as an example, the sentence

> *Great movie. sentiment: positive. You'll probably love it. sentiment: positive. Allen's funniest and most likeable movie in years. sentiment:* __.

is input to GPT-3 and the model tries to fill the blank with an answer. Note that the parameters in GPT-3 remain unchanged during the whole process.[4] Surprisingly, GPT-3 achieves strong performance with the few shots of training data. For example, it achieves performance comparable to state-of-the-art systems on the COPA (Choice Of Plausible Alternatives) task (Roemmele et al., 2011) with only 32 training examples.

Prompting is another few-shot transfer learning method with PLMs. Similar to priming, prompting also reformulates task data into cloze-style queries. In contrast to priming, which does not change PLM parameters, prompting allows practitioners to update the model parameters with gradient descent using few shots of data (Schick and Schütze, 2020; Gao et al., 2020; Liu et al., 2021). As a result, small PLMs (small with respect to GPT3) also perform strongly in few-shot learning scenarios.

In Chapter 5, we investigate the effectiveness of prompting in crosslingual transfer. We show that prompting is more parameter-efficient than plain finetuning in crosslingual transfer; prompting can also be conducted for non-English.

1.5 Summary and Dissertation Outline

This chapter introduces two basic topics relevant to this dissertation: Representation learning for Natural Language Processing and transfer learning. We describe methods of obtaining static and contextualized, monolingual and multilingual representations for Natural Language Processing. We also introduce transfer learning strategies from pretraining to downstream tasks, and from source languages to target languages. Furthermore, we review state-of-the-art transfer learning methods that are parameter- and label-efficient.

In Chapter 2, we leverage the sentenceID method to transfer sentiments from English to more than one thousand languages. In Chapter 3, we carry out a probing analysis demonstrating how BERT contextualizes words. We also show that finetuning does not significantly change the contextualization of BERT. Inspired by the results in Chapter 3, Chapter 4 proposes masking, an alternative transfer learning method that achieves comparable performance to finetuning, but is much more parameter-efficient. In Chapter 5, we investigate the effectiveness of prompting in crosslingual transfer, and show that it is label-efficient: Prompting outperforms finetuning with the same small number of training examples. Chapter 6 takes a closer look at the recently introduced few-shot crosslingual transfer paradigm; We show that crosslingual transfer results largely depend on the choices of the few shots of data. Lastly in Chapter 7, we explore a crucial question: What

[4]Priming can also be viewed as a parameter-efficient transfer learning method. However, priming gives inferior task performance when the underlying PLM is small (Brown et al., 2020).

is the practical utility of the large-scale pretrained-language-models? We propose *LMTurk*, which considers pretrained-language-model-based few-shot learners as crowdsourcing workers; we show that they are capable of annotating datasets for training small and deployable models that can be used in practical scenarios.

Chapter 2

A Multilingual BPE Embedding Space for Universal Sentiment Lexicon Induction

A Multilingual BPE Embedding Space for Universal Sentiment Lexicon Induction

Mengjie Zhao and **Hinrich Schütze**
CIS, LMU Munich, Germany
mzhao@cis.lmu.de

Abstract

We present a new method for sentiment lexicon induction that is designed to be applicable to the entire range of typological diversity of the world's languages. We evaluate our method on Parallel Bible Corpus+ (PBC+), a parallel corpus of 1593 languages. The key idea is to use Byte Pair Encodings (BPEs) as basic units for multilingual embeddings. Through zero-shot transfer from English sentiment, we learn a seed lexicon for each language in the domain of PBC+. Through domain adaptation, we then generalize the domain-specific lexicon to a general one. We show – across typologically diverse languages in PBC+ – good quality of seed and general-domain sentiment lexicons by intrinsic and extrinsic and by automatic and human evaluation. We make freely available our code, seed sentiment lexicons for all 1593 languages and induced general-domain sentiment lexicons for 200 languages.[1]

1 Introduction

Lexicons play an important role in sentiment analysis. Sentiment lexicons are available for high-resource languages like English (Pang et al., 2008; Baccianella et al., 2010; Mohammad and Turney, 2013), but not for many low-resource languages. Researchers are trying to fill this gap by inducing lexicons monolingually (Badaro et al., 2014; Eskander and Rambow, 2015; Rouces et al., 2018) as well as multilingually (Chen and Skiena, 2014), often by transfer from high-resource to low-resource languages.

The world's languages are heterogeneous – of particular relevance for us is heterogeneity with respect to morphology and with respect to marking token boundaries. This heterogeneity poses difficulties when designing a universal approach to lexicon induction that works for all languages – implementing a high quality tokenizer and morphological analyzer for each language is not feasible short-term. Given the small number of native speakers in low-resource languages (Goldhahn et al., 2016), crowdsourcing cannot easily be carried out either.

To overcome this heterogeneity and provide sentiment resources for low-resource languages, we present a new approach to sentiment lexicon induction that is universal – that is, it is applicable to the full range of typologically different languages – and apply it to 1593 languages. Our method first takes a parallel corpus as input and applies BPE (Gage, 1994) segmentation to it. We then create a multilingual BPE embedding space, from which a ZS (zero-shot) lexicon for each language $\mathcal{L}$ is extracted by zero-shot transfer from English sentiment to $\mathcal{L}$. We use PBC+, an expansion of the Parallel Bible Corpus (Mayer and Cysouw, 2014), as our parallel corpus. The ZS lexicons show high quality, but are specific to the domain of PBC+ (the Bible). We then adapt them to the general domain. For brevity, we also use *generic* to refer to *general-domain*.

Our method is universal and language-agnostic – it does not require language-dependent preprocessing. We carry out intrinsic and extrinsic, automatic and human evaluations on 95 languages. Intrinsic evaluation shows that our approach produces word ratings that strongly correlate with gold standard lexicons and human judgments. Extrinsic evaluation on Twitter sentiment classification demonstrates that our lexicons perform comparably or better than existing lexicons derived in multilingual settings.

We chose an **approach to sentiment analysis based on lexicons** in this paper because it is transparent and meets high standards of explainability. A classification decision can easily be traced

[1] cistern.cis.lmu.de

back to the lexicon entries in the document that are responsible. Many more complex methods, e.g., many deep learning approaches, do not meet this standard. Transparency is of particular importance for low-resource languages because error analysis and verification are paramount when working with small and noisy resources that are typical of low-resource languages.

Our **contributions**: (i) We propose a new method for inducing sentiment lexicons for a broad range of typologically diverse languages. We use BPEs as basic units and show that they work well across languages. (ii) We carry out extensive evaluation to confirm correctness and high quality of the created lexicons. (iii) We make our code, the 1593 ZS seed sentiment lexicons and 200 generic sentiment lexicons freely available to the community. This is the up-to-now largest sentiment resource in terms of language coverage that has been published.

2 Related Work

Monolingual Lexicon Induction. Sentiment lexicons for many languages have been induced. Eskander and Rambow (2015), Wang and Ku (2016), and Rouces et al. (2018) create Arabic, Chinese, and Swedish sentiment lexicons, respectively. Monolingually induced sentiment lexicons for specific domains like Twitter and finance are also devised (Mohammad et al., 2013; Hamilton et al., 2016). These methods are specialized such that applying them to other languages is non-trivial. For example, Eskander and Rambow (2015) link AraMorph (Buckwalter, 2004) with SentiWordNet by additionally considering part-of-speech information, which may not be available in lexical resources in other languages. Inducing Chinese sentiment lexicons (Wang and Ku, 2016) needs properly tokenized corpora, which is not a hard requirement in Swedish. In contrast, we aim to design a method applicable to typologically diverse languages and we apply it to 1500+ languages.

Bi/Multi-Lingual Lexicon Induction. Gao et al. (2015) propose a graph based method for learning sentiment lexicons in target language by leveraging English sentiment lexicons. They rely on a high-quality word alignment, which is difficult to produce if languages are typologically diverse and the size of the parallel corpus is small. Chen and Skiena (2014) devise a knowledge graph based method to build sentiment lexicons for 136 major languages. Several linguistic resources such as Google Translate and Wiktionary are used to link words across languages. In contrast, our approach uses *BPE embeddings* to extract alignment signals from the parallel corpus, an approach that is better applicable across diverse languages. We do not require resources like Wiktionary. We cover more languages than Chen and Skiena (2014) and more words (e.g., 300K for Amharic).

eng	The book of the history of Jesus Christ , son of David , son of Abraham :
fra	Le livre de l'histoire de Jésus Christ , fils de David , fils d'Abraham :
jpn	アブラハムの子，ダビデの子，イエス･キリストについての歴史の書 :

Table 1: PBC+ verse 40001001 in three languages

Language-Agnostic NLP. Language-agnostic NLP has demonstrated strong performance in areas such as *neural machine translation* (NMT) and universal *representation learning*. A particular difficulty is languages that do not mark token boundaries by whitespace such as Japanese. We refer to them as *non-segmented languages*. Sennrich et al. (2016) show the strength of BPE in translating rare words. Kudo (2018) introduces subword regularization that utilizes multiple subword sequences to improve the robustness of NMT models. Sennrich et al. (2016)'s `subword-nmt`[2] requires preprocessing (specifically, tokenization) for non-segmented languages, however, `sentencepiece`[3] (Kudo and Richardson, 2018) used by Kudo (2018) requires no preprocessing even for non-segmented languages. This research indicates the potential of language-agnostic *NMT*.

Effective representations of *words* (Schütze, 1993), e.g., word embeddings (Mikolov et al., 2013; Pennington et al., 2014), have been extended to be bilingual (Ruder, 2017; Artetxe et al., 2017) or multilingual (Dufter et al., 2018), with (Artetxe et al., 2018) and without (Conneau et al., 2017) supervision. Artetxe and Schwenk (2018) train a language-agnostic BiLSTM encoder creating universal *sentence* representations of 93 languages, and performing strongly in crosslingual tasks. Lample and Conneau (2019) show that pretraining the encoders with a crosslingual language model objective helps in achieving state-

[2] `github.com/rsennrich/subword-nmt`
[3] `github.com/google/sentencepiece`

of-the-art results in crosslingual classification and NMT. This research demonstrates the strength of language-agnostic methods for *representation learning* in NLP. Language-agnostic NLP models can generalize across languages without requiring language-dependent preprocessing. These advantages motivate us to design a universal approach for sentiment lexicon induction for 1500+ languages.

3 Method

Figure 1 shows the four steps of our method: (i) BPE segmentation. (ii) Multilingual embedding space creation. (iii) ZS lexicon induction. (iv) Domain adaptation to the general domain. We work with the parallel corpus PBC+. PBC+ extends the Parallel Bible Corpus by adding[4] 500 translations of the New Testament in 334 languages, resulting in a sentence-aligned parallel corpus containing New Testament verses in 2164 translations of 1593 languages. Many languages have several translations of the New Testament in PBC+. We use the term "edition" to refer to a single translation. Table 1 shows a verse in three languages. As shown, the Japanese (jpn) verse is not tokenized.

3.1 BPE Segmentation

Given the linguistic heterogeneity of the world's languages, it is crucial to first decide which type of linguistic unit to use to represent a language $\mathcal{L}$ in the multilingual space. The word, the linguistic unit typically generated from whitespace tokenization, is not ideal for universal approaches because non-segmented languages require carefully designed tokenizers. Character (or byte) n-gram is an alternative unit (Wieting et al., 2016; Gillick et al., 2016; Schütze, 2017; Dufter et al., 2018), but the optimum length n varies across languages, e.g., $n = 2$ may be suitable for Chinese (Foo and Li, 2004), but clearly not for English.

In our desire to design a universal approach, we use `sentencepiece` to segment PBC+ editions in all 1593 languages into sequences of *BPE segments*. We will show that this segmentation works across languages.

The widely used BPE segmentation algorithm `subword-nmt` only considers BPE segments within words (Sennrich et al., 2016) and some frequent BPEs are essentially valid *words*. `sentencepiece` adopts this setting for segmented languages like English (Kudo, 2018). But for non-segmented languages, `sentencepiece` does not require any language-dependent preprocessing – it learns a data-driven "tokenizer" on-the-fly from raw text. Hence, `sentencepiece` BPE segments can be larger linguistic units than say, English words, e.g., phrases. Examples for Japanese BPE segments in PBC+ are: "愛のうちに" (in love) and "何と言えばよいでしょうか" (what should I say).

We will use the term "BPE" to refer to all BPE segments produced by `sentencepiece`, including subwords, words and cross-token units like phrases. Figure 1 (a) shows some sample units. As shown, the English segments can be words or subwords (underlined). Dominant contexts of shown subwords – insp: inspiration, inspired; crim: crime, criminals; blasphe: blasphemy, blasphemed; hest: highest, richest.

3.2 Multilingual Space Creation

We next create the multilingual space hosting BPEs in 1593 languages of PBC+. We use the Sentence ID (S-ID) method (Levy et al. (2017), cf. also Le and Mikolov (2014)), a strong baseline in multilingual embedding learning.

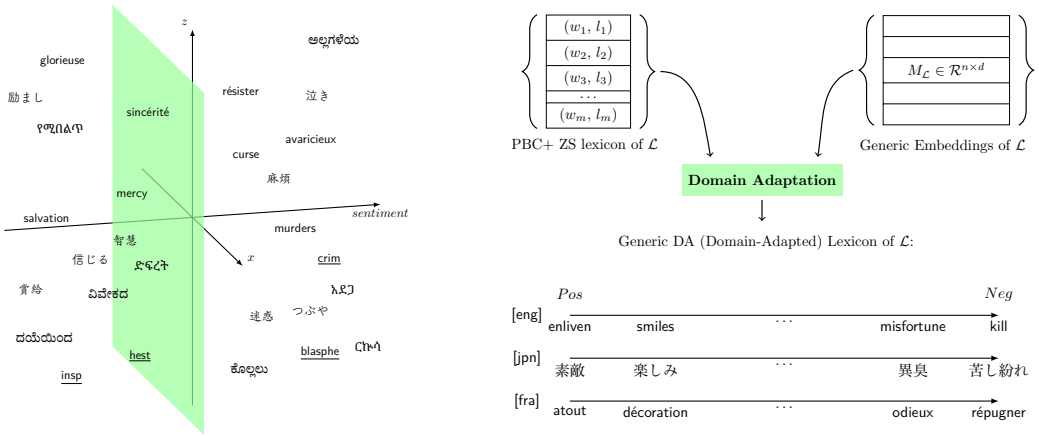
Given a sentence-aligned parallel corpus, the S-ID method first creates an embedding training corpus by recording co-occurrences between the sentence ID and the sentence's words (the New Testament verse ID and BPEs in our case) in all languages. Figure 2 shows examples from the training corpus; each BPE is associated with a 3-digit ISO 639-3 language code. After that, an embedding learner is applied to the created corpus to learn the multilingual space. We use *word2vec-skipgram* (Mikolov et al., 2013) as our embedding learner.

3.3 Zero-Shot Transfer of English Sentiment

Embeddings encode sentiment information (Pennington et al., 2014; Tang et al., 2014; Amir et al., 2015; Rothe et al., 2016). We exploit this for zero-shot transfer of English sentiment to the other 1592 languages. We train two linear SVMs to classify sentiment of English BPE embeddings as positive vs. non-positive (POS) and as negative vs. non-negative (NEG).

We use this setup – as opposed to binary classification positive vs. negative – to address the fact that some long BPE segments in non-segmented

[4] We use `github.com/ehsanasgari/1000Langs`



(a) **PBC+ ZS (zero-shot) lexicons**: Created by zero-shot crosslingual transfer

(b) **Generic DA (domain-adapted) lexicons**: Created by PBC-to-general-domain adaptation

Figure 1: Universal sentiment lexicon induction. (a): S-ID multilingual space of BPEs and sentiment classification hyperplanes (only the positive vs. non-positive plane is shown) learned from English. Underlined units are English BPEs with strong sentiment. (b): Creating generic DA lexicons using PBC+ ZS lexicons and generic embeddings.

languages may encode both sentiments. Using two SVMs allows us to identify then filter out segments with compositional sentiments during zero-shot transfer. This setup also enables direct comparison with Dufter et al. (2018) in Table 2.

The two SVMs are then applied to all embedding vectors in the multilingual space to yield a ZS lexicon for each of the 1593 languages.

3.4 PBC+ to General Domain Adaptation

Our ZS lexicons show high quality (see §5.2), but are specific to the PBC+ domain, i.e., the Bible. We adapt them to the general domain by obtaining generic embeddings and using ZS lexicon BPEs as labels to predict the sentiment of each generic embedding.

We assume that we have access to generic embeddings or, alternatively, that we can learn them from a generic corpus. We now describe how we predict the sentiment of generic embeddings. Given the PBC+ ZS lexicon $\mathcal{B}$ and the generic embedding matrix $M_\mathcal{L} \in \mathbb{R}^{n \times d}$ of language $\mathcal{L}$, we train a matrix $Q_\mathcal{L} \in \mathbb{R}^{d \times d}$ such that BPE pairs with same sentiment ($G_s \subset \mathcal{B} \times \mathcal{B}$) have small $l2$ distance while BPE pairs with different sentiment ($G_d \subset \mathcal{B} \times \mathcal{B}$) have large $l2$ distance, i.e., $\forall w, v \in \mathcal{B}, w \neq v$:

$$\arg\min_{Q_\mathcal{L}} \sum_{(w,v) \in G_d} -\alpha \|PQ_\mathcal{L}(e_w - e_v)\|_2 \quad + \sum_{(w,v) \in G_s} (1-\alpha)\|PQ_\mathcal{L}(e_w - e_v)\|_2 + \frac{\lambda}{2}\|PQ_\mathcal{L}\|_F^2$$

where $e_w, e_v \in \mathbb{R}^d$ are embeddings of BPEs w, v. d is embedding dimension. n is vocabulary size. $\alpha \in [0, 1]$ is the hyperparameter balancing the two sub-objectives. λ is a regularization weight. $P \in \mathbb{R}^{d \times d}$ is an identity matrix in the first dimension, i.e., a selector. This objective concentrates sentiment information in an embedding vector to a 1-dimensional ultradense sentiment space, resulting in a real-valued generic sentiment score. We minimize the objective using stochastic gradient descent (SGD).

After training, the generic sentiment score of BPE w in language $\mathcal{L}$ is computed as $s_w = PQ_\mathcal{L}e_w$. We refer to this method as *REG* and we call a lexicon computed by REG a *generic DA (domain-adapted) lexicon* since we always adapt from the Bible to the general domain in this paper.

REG is inspired by Densifier (Rothe et al., 2016), which is state of the art on SemEval2015 10E (Rosenthal et al., 2015) – determining

40001002	@Jesus:eng
40001002	@አብርሃም:amh
40001002	@ನಿಸ್ಸೂ:kan
40001002	@雅各:zho
66002003	བཟོད་བསྟན་བྱས:bod
...	...

Figure 2: Samples of S-ID embedding training corpus. 40001002 and 66002003: S-ID, i.e., IDs of New Testament verses. amh=Amharic, kan=Kannada, zho=Chinese, bod=Tibetan.

strength of association of Twitter terms with sentiment. Rothe et al. (2016) show that Densifier induces high quality and coverage sentiment lexicons in a *domain adaptation* setup. Densifier forces $Q_{\mathcal{L}}$ to be orthogonal to preserve the structure of the embedding space. As we are only interested in accurate sentiment prediction, we replace the orthogonality with $l2$ regularization: $\frac{\lambda}{2}\|PQ_{\mathcal{L}}\|_F^2$. The orthogonal constraint in Densifier – computing an SVD after each batch update – is expensive ($\mathcal{O}(d^3)$) and requires non-trivial training regime (Rothe et al., 2016). We will show that our formalization delivers comparable results.

In our experiments, we can use the generic *word embeddings* provided by Bojanowski et al. (2017) for 157 languages. Additionally, Heinzerling and Strube (2018) create generic *BPE embeddings* for 257 languages by segmenting Wikipedia articles using `sentencepiece` then running GloVe on the segmented corpora. As discussed above (§3.1), some BPEs in the PBC+ ZS lexicons are words, some are subwords – so we can utilize both sets.

4 Experiments

4.1 Datasets and Settings

We use the 7958 New Testament verses in PBC+ that were also used by Dufter et al. (2018) to create the multilingual BPE embedding space. To cover as many BPEs as we can, we segment each PBC+ edition three times with vocabulary sizes 2000, 4000 and 8000 using `sentencepiece`. S-ID generates a 31GB embedding training corpus including 7,414,810 BPEs in 1593 languages.

English training set. We employ VADER, a simple but widely used rule-based model for general sentiment analysis (Hutto and Gilbert, 2014), to create sentiment labels for English BPEs. We consider BPEs with sentiment score $\geqslant$ +0.1 (resp. $\leqslant$ -0.1) as positive (resp. negative). BPEs with score 0 are treated as neutral. As a result, we have 851 positive, 906 negative and 13,861 neutral training BPEs in English. We uniformly sample 878 = floor$((851 + 906)/2)$ neutral BPEs to speed up training.

Zero-shot transfer. The two SVMs for POS and NEG (§3.3) are trained on English training set (see above), then applied to all vectors in the multilingual BPE embedding space to create ZS lexicons for 1593 languages. We only keep high-confidence BPEs – those with a predicted probability for either POS or NEG of ≥ 0.7 (Platt et al., 1999) – to ensure ZS lexicons encode clear sentiment signals. The PBC+ ZS lexicon of language $\mathcal{L}$ is then the set of all high-confidence sentiment-bearing BPEs from $\mathcal{L}$.

Evaluation. Following Abdaoui et al. (2017), Bar-Haim et al. (2017), Rouces et al. (2018), we evaluate the quality of *PBC+ ZS lexicons* based on gold sentiment lexicons in Japanese (JA) (concatenation of Kobayashi et al. (2005); Higashiyama et al. (2008)), Czech (CZ) (Veselovská and Bojar, 2013), German (DE) (Waltinger, 2010), Spanish (ES) (Perez-Rosas et al., 2012), French (FR) (Abdaoui et al., 2017) and English (EN) (*WHM* lexicon, the concatenation of Wilson et al. (2005), Hu and Liu (2004) and Mohammad and Turney (2013), created by Rothe et al. (2016)). F1 is evaluation metric. We always compute F1 on the intersection of our and gold lexicon. Gold lexicons are also used in intrinsic evaluation of generic DA lexicons (Table 6). Additionally, the English WHM lexicon is also used in the evaluation of the universality of our approach (Table 8).

For *intrinsic evaluation* of *generic DA lexicons*, we compare our results with Densifier. Rothe et al. (2016) provide embeddings and train/validation splits of gold standard lexicons in CZ, DE, ES, FR and EN – we also use them in our experiments. We show (i) using GEN (the same training words as Densifier), REG (§3.4) induces generic lexicons in comparable quality; (ii) using PBC+ ZS lexicons, the induced generic DA lexicons are also in high quality. Kendall's τ (Kendall, 1938) is evaluation metric. As Densifier is implemented in *MATLAB*, we implement our model in *NumPy* (Oliphant, 2006) which is more accessible to the community.

For *extrinsic evaluation* of *generic DA lexicons*, we carry out Twitter sentiment classification in 13 languages. For each language, we retrieve $\approx$12,000 tweets from the human annotated dataset devised by Mozetič et al. (2016), and sample balanced number of positive and negative tweets (for clearer comparisons and descriptions) which are then randomly split 80/20 into train/test. We compare our lexicons with Chen and Skiena (2014)'s work. Two classification models are used (§5.3) – COUNT (count-based, Chen and Skiena (2014)) and ML (machine-learning-based, Eskander and Rambow (2015)). Accuracy is evaluation metric.

4.2 Hyperparameter Tuning

We train the multilingual BPE embedding space using *word2vec-skipgram* with default parameters except: 25 negative samples, 10^{-4} occurrence threshold, 200 dimensions and 10 iterations.

We tune the two linear SVMs for POS and NEG by 5-fold cross validation on English training set.

Following Rothe et al. (2016), when inducing generic DA lexicons, we run a grid search on their train/validation sets to find α and λ. With the same settings, we additionally conduct an experiment on Japanese (JA Wiki), a non-segmented language, to show the universality of our approach. For EN Twitter (SemEval2015 10E), we tune our model on the trial (dev) set and report results on the test set. In all experiments, we search $\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$, $\lambda \in \{0.01, 0.1, 1\}$. Learning rate is 0.1, batch size 100, and the maximum number of updating steps 30,000.

Following Eskander and Rambow (2015), in machine-learning-based Twitter sentiment classification for each of the 13 languages, we find the optimum SVM (positive vs. negative tweet) hyperparameters (C and kernel) by running 5-fold cross validation on the training set.

5 Results and Discussion

5.1 Multilingual BPE Space Evaluation

We first evaluate the multilingual BPE space by carrying out the crosslingual *verse sentiment* classification experiment in Dufter et al. (2018). Two linear SVMs are trained on 2147 English training verses to classify the verse sentiment (positive vs. non-positive, i.e., POS, and negative vs. non-negative, i.e., NEG). A verse is represented as the TF-IDF weighted sum of the embeddings of its BPEs. We then conduct the crosslingual verse sentiment analysis – using the SVMs to classify 476 test verses of Dufter et al. (2018)'s 1664 editions in 1259 languages. Table 2 gives results averaged over 1664 editions. *Word* and *Char* are two multilingual spaces created by Dufter et al. (2018). For *Word*, whitespace tokenization is used to segment all editions. For *Char*, all editions are segmented to sequences of overlapping byte-ngrams (length *n* varies across languages, see Dufter et al. (2018)). Next, the S-ID method is utilized to create the two multilingual spaces.

The S-ID BPE space outperforms both S-ID Word and S-ID Char spaces. This observation meets our expectation – the data-driven BPE segmentation is superior to splitting on whitespace (Word) or overlapping byte-ngram segmentation (Char), for non-segmented languages like Japanese whose PBC+ editions are not tokenized.

	Word		Char		BPE	
	POS	NEG	POS	NEG	POS	NEG
S-ID	.79	.88	.65	.86	**.81**	**.89**

Table 2: F1 for verse sentiment classification. Bold: our results. Word/Char are from Dufter et al. (2018).

ISO	B	W	Δ	ISO	B	W	Δ
lzh1	.82	.04	+.78	eng1	.88	.84	+.04
jpn1	.86	.19	+.67	fra1	.85	.85	-.00
khm2	.87	.21	+.66	deu1	.84	.83	+.01
khm3	.86	.25	+.61	spa1	.85	.85	+.00
ksw0	.86	.32	+.54	por1	.84	.87	-.03

Table 3: The most improved (left) editions when using S-ID BPE (*B*) compared with S-ID Word (*W*). *B* and *W* perform similarly on segmented languages (right) like English (eng), French (fra), German (deu), Spanish (spa) and Portuguese (por). Numbers are in F1.

For the more challenging subtask POS, we find the biggest improvement of S-ID BPE over Word is for non-segmented languages like Classical Chinese (lzh), Japanese (jpn), Khmer (khm) and S'gaw Karen (ksw) as shown in Table 3 (left). For segmented languages, S-ID BPE delivers similar performance as S-ID Word as shown in Table 3 (right). This observation also meets our expectation – lots of BPEs in segmented languages are essentially valid words.

These observations show the universality of our approach. The sentiment information derived from English is successfully transferred to heterogeneous languages without language-dependent preprocessing – even for non-segmented languages.

5.2 PBC+ ZS (Zero-Shot) Lexicon Evaluation

Sample entries in the English ZS lexicon are shown in Table 4 (left) as a **qualitative** evaluation. Table 5 shows the high consistency between the PBC+ ZS lexicons and gold lexicons in six languages. These results indicate that the

positive	negative	positive	negative
magnificent	fought	#blessedbeyondbelief	shats
privilege	blamed	alhamduillah	#worstpain
enjoyed	debauchery	#365daysofgratitude	theiving
salvation	adulter	#excellence	#stuffynose
rejoices	gloomy	co-create	sorethroat

Table 4: Sample entries in English ZS lexicon (left) and DA lexicon with Twitter embeddings (right).

two SVMs trained on English BPE embeddings perform strongly in a zero-shot crosslingual setting, and the resulting PBC+ ZS lexicons in difficult (morphologically rich, e.g., Czech; non-segmented, e.g., Japanese) languages encode clear sentiment information.

5.3 Generic DA (Domain-Adapted) Lexicon Evaluation

Table 4 (right) **qualitatively** shows the most sentiment-bearing words of the DA lexicon induced with English ZS lexicon and Twitter embeddings (EN Twitter). Lots of top ranked words are strong sentiment-bearing hashtags that never occur in the ZS lexicon domain, illustrating that our approach functions well in the *domain adaptation* setup. This observation is consistent with Densifier (Rothe et al., 2016).

Intrinsic evaluation: ranking correlation. We compute ranking correlation between our generic DA lexicons and gold standard lexicons. There are overlapping words between our PBC+ ZS lexicon BPEs and the validation/test sets used by Rothe et al. (2016) – we discard these training words for a clean comparison.

Columns (i) and (ii) of Table 6 show that REG (§3.4) delivers results comparable to Densifier (ORTH) when using the same set of generic training words (GEN) in lexicon induction. However, our method is more efficient – no need to compute the expensive SVD after every batch update.

Comparing columns (ii) and (iii), we see a marginal decrease of τ between .020 and .057 when GEN is replaced by PBC+ ZS lexicons. Note that PBC+ ZS lexicons have much fewer training BPEs than GEN (e.g., 343 vs. 4298 in JA Wiki) – this may contribute to the decrease. These comparable results also reflect the correctness of PBC+ ZS lexicons.

We also use $\alpha = 0.4$ and $\lambda = 0.01$, the optimal hyperparameter values found on the trial set of EN Twitter, to induce generic DA lexicons for the other languages. This is the common setting in real applications – other languages most likely do not have validation sets available. Results are shown in column (iv). Compared with tuned results (PBC+/T), performance slightly drops as the hyperparameters are not tuned (PBC+/NT) for languages other than EN Twitter.

Overall, the performance differences between GEN (based on generic gold standard lexicons) and PBC+ (based on PBC+ ZS lexicons) are small and τ correlations are high. The high quality of generic DA lexicons in these six diverse (morphologically rich and non-segmented) languages shows the universality of our approach again – no language-dependent preprocessing is needed.

Extrinsic evaluation: Twitter sentiment classification. Based on the subset of frequent words only,[5] we use the top 10% most positive and most negative words for this evaluation. We compare with the closest work – lexicons from Chen and Skiena (2014).

Two classification models are used – word-count-based model COUNT (Chen and Skiena,

	JA	CZ	DE	ES	FR	EN
F1	.883	.914	.903	.963	.916	.939
∩ size	120	141	788	63	407	1145
|PBC+|	728	1793	2827	1766	2193	2563

Table 5: High consistency between PBC+ ZS lexicons and generic gold lexicons in JA and five languages used in Rothe et al. (2016). ∩ size: intersection size. |PBC+|: ZS lexicon size.

	(i)	(ii)	(iii)	(iv)
	ORTH	REG		
	GEN	GEN	PBC+/T	PBC+/NT
CZ web	.580	.576	.529	.524
DE web	.654	.654	.634	.634
ES web	.563	.568	.524	.514
FR web	.544	.540	.514	.474
EN Tw.	.654	.629	.583	.583
EN Ne.	.622	.582	.562	.557
JA Wiki	n/a	.628	.571	.558

Table 6: Correlation (τ) of generic DA lexicons with gold standard lexicons. ORTH results are from Rothe et al. (2016). The other columns use REG (§3.4). Training words for lexicon induction are from Rothe et al. (2016) (GEN) and from PBC+ ZS lexicons.

Algorithm 1 Creating tweet representation

```
procedure RepTweet(String: Tweet, Dict: Lexicon)
    words = Tweet.split(" ")
    vec = [0.0, 0.0]
    for w ∈ words do
        val = Lexicon.get(w)
        if val > 0 then
            vec[0] = vec[0] + val
        else if val < 0 then
            vec[1] = vec[1] + val
        else
            continue
    return vec
```

Figure 3: Creating the representation of a tweet in Twitter sentiment classification using ML.

[5] In all discussions, we consider words that are top 50% frequent in the embedding vocabulary as "frequent" words.

		sqi	bul	hrv	deu	hun	pol	por	rus	srp	slk	slv	spa	swe	$\bar{x}$
COUNT	C&S	.55	.57	.57	.61	.61	.55	.57	.54	.51	.55	.64	.54	.57	.57
	Ours	.50	.60	.60	.56	.64	.62	.53	.65	.50	.61	.57	.55	.63	**.58**
ML	C&S	**.58**	.59	.60	.62	.64	.56	.54	.56	.51	.57	**.66**	.53	.59	.58
	Ours	.54	**.65**	**.65**	**.64**	**.66**	**.66**	.54	**.67**	.51	**.64**	.59	**.57**	**.64**	**.61**

Table 7: Accuracy of Twitter sentiment classification in Albanian (sqi), Bulgarian (bul), Croatian (hrv), German (deu), Hungarian (hun), Polish (pol), Portuguese (por), Russian (rus), Serbian (srp), Slovak (slk), Slovenian (slv), Spanish (spa) and Swedish (swe). Baseline of all experiments: 0.5.

2014), and machine-learning-based model ML (Eskander and Rambow, 2015). COUNT labels a tweet with the sentiment that has more word occurrences in the tweet (positive in case of ties). COUNT does not require training and the results are from all tweets for each language. In ML, the vector representation of a tweet is created according to Figure 3. Our generic DA lexicons support computing real-valued vectors in this way. Chen and Skiena (2014)'s lexicons are discrete (1/-1); we use these discrete values when applying ML to their lexicons. Finally, for each language, an SVM is trained on the 2-dimensional vectors.

Table 7 shows results. The baseline accuracy is 0.5 for all experiments as our dataset is balanced. Rows *Ours* and *C&S* show results using our and Chen and Skiena (2014)'s lexicons respectively. As shown, the two sets of lexicons give comparable results in COUNT. But ML generally performs better than COUNT, and our lexicons give better classification results – our real-valued representation of tweets is superior to the discrete one computed with Chen and Skiena (2014)'s lexicons.

Overall, intrinsic and extrinsic evaluations on diverse languages demonstrate the high quality of our generic DA lexicons.

5.4 Evaluation of Universality

We further conduct automatic and human evaluations on 95 diverse languages to show the universality of our approach. We focus on intrinsic evaluation – verifying the correctness of PBC+ ZS lexicons with F1, and assessing the quality of generic DA lexicons using τ. The extrinsic evaluation, i.e., Twitter sentiment classification, is not feasible here due to missing human annotated Twitter datasets in low-resource languages.

Automatic evaluation. Similar to Chen and Skiena (2014); Abdaoui et al. (2017), we use Google Translate (GT) for automatic evaluation – given a non-English language $\mathcal{L}$, we translate its PBC+ ZS lexicon and generic DA lexicon into English. Translated English lexicons are then evaluated against the gold English lexicon WHM.

GT supports 102 non-English languages. We omit ten languages that (i) are not covered by PBC+ (Corsican, Galician, Pashto, Yiddish); (ii) are covered in PBC+, but not in the alphabet used by GT (Malayalam); (iii) do not have public pretrained embeddings (Filipino, Hmong, Kyrgyz, Sesotho); or (iv) are very close to another language (we keep Croatian, but do not include Bosnian). We conduct separate experiments for Bokmål and Nynorsk, which are not distinguished by GT. Thus, we evaluate on 93 languages. When translating words to English, we discard entries where GT fails (i.e., output is identical to input). As GT requires the uploaded file to be small ($\leqslant$ 1MB), we do the evaluation on uniformly sampled 600 top 1% positive and negative words that are frequent. For ten languages (Chichewa, Hausa, Hawaiian, Igbo, Lao, Maori, Samoan, Shona, Xhosa, Zulu) that have very small embedding training corpora (<5MB Wikipedia pages and articles) and vocabulary sizes (e.g., 5000 for Hausa), we sample 200 words at 10%.

Table 8 shows results. We see that PBC+ ZS lexicons show high consistency with gold labels across all 93 languages (F1 columns), including morphologically rich languages like Czech and Turkish, and non-segmented languages like Japanese and Khmer. The generic DA lexicons show high correlation with gold labels (τ columns) – with two exceptions. First, some languages have low-quality embeddings due to small embedding training corpora (e.g., Hawaiian: 998 KB; Igbo: 1014 KB) or because the training corpora apparently have low quality – e.g., the Luxembourgish embedding vocabulary contains a large amount of French and German words, suggesting that it was trained on mixed text and that the genuine Luxembourgish part is small. Second, GT does not perform well for some of the languages, again this is the case for Luxembourgish and also for Frisian. To give an example from Lux-

Language	F1	τ	Language	F1	τ	Language	F1	τ	Language	F1	τ	Language	F1	τ
Afrikaans	.909	.508	Esperanto	.933	.361	Italian	.924	.591	Mongolian	.840	.222	Sundanese	.912	.409
Albanian	.916	.570	Estonian	.889	.606	Japanese	.901	.411	Myanmar	.916	.534	Shona	.885	.223
Amharic	.870	.418	Finnish	.932	.584	Javanese	.904	.398	Nepali	.862	.491	Swedish	.936	.621
Arabic	.905	.509	French	.919	.600	Kannada	.921	.447	Nynorsk	.853	.434	Sinhala	.880	.540
Armenian	.848	.524	Frisian	.885	.065	Kazakh	.893	.421	Punjabi	.927	.506	Tajik	.876	.436
Azerbaijani	.768	.401	Georgian	.908	.540	Khmer	.906	.474	Persian	.903	.390	Tamil	.911	.513
Basque	.898	.477	German	.898	.548	Korean	.897	.481	Polish	.923	.530	Telugu	.934	.297
Belarusian	.915	.597	Greek	.912	.570	Kurdish	.925	.258	Portuguese	.913	.574	Thai	.867	.357
Bengali	.910	.389	Gujarati	.896	.479	Latin	.927	.336	Romanian	.917	.644	Turkish	.897	.607
Bokmål	.927	.625	Haitian	.891	.238	Lao	.834	.222	Russian	.910	.596	Ukrainian	.909	.612
Bulgarian	.911	.511	Hausa	.905	.184	Latvian	.919	.538	Scots	.848	.385	Urdu	.825	.258
Catalan	.937	.453	Hawaiian	.951	.078	Lithuanian	.922	.491	Serbian	.957	.559	Uzbek	.900	.361
Cebuano	.917	.390	Hebrew	.833	.522	Luxemb'gish	.834	.031	Sindhi	.845	.169	Vietnamese	.840	.403
Chichewa	.872	.061	Hindi	.878	.447	Macedonian	.918	.425	Slovak	.942	.515	Welsh	.879	.560
Chinese	.889	.486	Hungarian	.910	.502	Malagasy	.923	.417	Samoan	.857	.116	Xhosa	.892	.057
Croatian	.926	.519	Igbo	.791	.088	Malay	.892	.494	Swahili	.842	.403	Yoruba	.873	.188
Czech	.915	.545	Icelandic	.947	.417	Maori	.836	.015	Slovenian	.957	.483	Zulu	.889	.226
Danish	.936	.359	Indonesian	.898	.498	Maltese	.938	.488	Somali	.954	.335			
Dutch	.906	.553	Irish	.902	.476	Marathi	.942	.479	Spanish	.943	.428			

Table 8: Intrinsic evaluation of our PBC+ ZS and generic DA lexicons in 93 languages. We see high consistency (F1) between PBC+ ZS lexicons and gold labels across languages. The generic DA lexicons are strongly correlated (τ) with gold labels in most languages.

	Hiligaynon		Tibetan	
	τ	size	τ	size
2-way	.474	103	.542	64
3-way	.357	188	.361	148

Table 9: Human evaluation of generic DA lexicons in Hiligaynon and Tibetan. 2-way: positive, negative. 3-way: positive, neutral, negative.

embourgish for both problems: "vergloust" and its first nearest neighbor "verglousten" are translated by GT as "glowed" and "forget about it". We recommend to use the higher quality PBC+ ZS lexicon for these languages.

Apart from above exceptions, both F1 and τ are reasonably high, evidencing that our universal approach is applicable to a broad range of typologically diverse languages.

We do **human evaluation** for Hiligaynon and Tibetan, languages not supported by GT.

There are no public pretrained embeddings for Hiligaynon. We train embeddings on a concatenation of texts from project *Palito* (Dita et al., 2009) and *Jehovah's Witnesses* e-books (www.jw.org). From the generic DA Hiligaynon and Tibetan lexicons, we uniformly sample 199 from the top 10% positive and negative frequent BPEs.

Two Tibetan scholars and three Hiligaynon speakers annotated these BPEs as positive, negative, neutral, unclear where the last category refers to cases where the intended word is not apparent from the BPE. We omit entries labeled as unclear and compute τ. Table 9 shows τ averaged over annotators. We see that our lexicons have consistent positive correlation with the human annotation in both languages.

6 Conclusion

We proposed a universal approach for sentiment lexicon induction. By creating a multilingual BPE embedding space for 1500+ languages, we successfully transfer sentiment to each language without language-dependent preprocessing. We created 1593 ZS (zero-shot) sentiment lexicons and showed for a subset that they are highly consistent with gold lexicons. To address the fact that the small-size ZS lexicons are specific to PBC+'s domain, we conduct domain adaptation and induce large-size generic DA (domain-adapted) lexicons for 200 languages. Extensive intrinsic and extrinsic, automatic and human evaluations on 95 languages confirm the correctness and good quality of our lexicons. We make our code and lexicons freely available to the community.

To induce generic lexicons, our approach requires generic embeddings, which are not always available for low-resource languages. Solving this problem is non-trivial as many low-resource languages have a limited amount of written text in electronic form (and in any form). In such cases, the PBC+ ZS lexicons can be utilized because they also have high quality.

Acknowledgements. We thank Philipp Dufter and the anonymous reviewers for comments and suggestions; and Mary Ann C. Tan, Samyo Rode and Nikolai Solmsdorf for sentiment judgments for Hiligaynon and Tibetan. This work was funded by the European Research Council (ERC #740516).

References

Amine Abdaoui, Jérôme Azé, Sandra Bringay, and Pascal Poncelet. 2017. Feel: a french expanded emotion lexicon. *Language Resources and Evaluation*, 51(3):833–855.

Silvio Amir, Ramón Astudillo, Wang Ling, Bruno Martins, Mario J. Silva, and Isabel Trancoso. 2015. Inesc-id: A regression model for large scale twitter sentiment lexicon induction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 613–618. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5012–5019.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Gilbert Badaro, Ramy Baly, Hazem Hajj, Nizar Habash, and Wassim El-Hajj. 2014. A large scale arabic sentiment lexicon for arabic opinion mining. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 165–173.

Roy Bar-Haim, Lilach Edelstein, Charles Jochim, and Noam Slonim. 2017. Improving claim stance classification with lexical knowledge expansion and context utilization. In *Proceedings of the 4th Workshop on Argument Mining*, pages 32–38, Copenhagen, Denmark. Association for Computational Linguistics.

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.

T Buckwalter. 2004. Buckwalter arabic morphological analyzer (bama) version 2.0. linguistic data consortium (ldc) catalogue number ldc2004l02. Technical report, ISBN1-58563-324-0.

Yanqing Chen and Steven Skiena. 2014. Building sentiment lexicons for all major languages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 383–389. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Shirley N. Dita, Rachel Edita O. Roxas, and Paul Inventado. 2009. Building online corpora of philippine languages. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2*.

Philipp Dufter, Mengjie Zhao, Martin Schmitt, Alexander Fraser, and Hinrich Schütze. 2018. Embedding learning through multilingual concept induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1520–1530. Association for Computational Linguistics.

Ramy Eskander and Owen Rambow. 2015. Slsa: A sentiment lexicon for standard arabic. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2545–2550. Association for Computational Linguistics.

Schubert Foo and Hui Li. 2004. Chinese word segmentation and its effect on information retrieval. *Information processing & management*, 40(1):161–190.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Dehong Gao, Furu Wei, Wenjie Li, Xiaohua Liu, and Ming Zhou. 2015. Cross-lingual sentiment lexicon learning with bilingual word graph label propagation. *Computational Linguistics*, 41(1):21–40.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306. Association for Computational Linguistics.

Dirk Goldhahn, Maciej Sumalvico, and Uwe Quasthoff. 2016. Corpus collection for under-resourced languages with more than one million speakers. *CCURL 2016 Collaboration and Computing for Under-Resourced Languages: Towards an Alliance for Digital Language Diversity*, page 67.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605. Association for Computational Linguistics.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Masahiko Higashiyama, Kentaro Inui, and Yuji Matsumoto. 2008. Learning sentiment of nouns from selectional preferences of verbs and adjectives. In *Proceedings of the 14th Annual Meeting of the Association for Natural Language Processing*, pages 584–587.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Clayton J. Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014.*

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Nozomi Kobayashi, Kentaro Inui, Yuji Matsumoto, Kenji Tateishi, and Toshikazu Fukushima. 2005. Collecting evaluative expressions for opinion extraction. In *Proceedings of the First International Joint Conference on Natural Language Processing*, IJCNLP'04, pages 596–605, Berlin, Heidelberg. Springer-Verlag.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 765–774. Association for Computational Linguistics.

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel bible corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327. Association for Computational Linguistics.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Igor Mozetič, Miha Grčar, and Jasmina Smailović. 2016. Multilingual twitter sentiment classification: The role of human annotators. *PloS one*, 11(5):e0155036.

Travis E Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International*

Workshop on Semantic Evaluation (SemEval 2015), pages 451–463. Association for Computational Linguistics.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777. Association for Computational Linguistics.

Jacobo Rouces, Nina Tahmasebi, Lars Borin, and Stian Rødven Eide. 2018. SenSALDO: Creating a Sentiment Lexicon for Swedish. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Sebastian Ruder. 2017. A survey of cross-lingual embedding models. *CoRR*, abs/1706.04902.

Hinrich Schütze. 1993. Word space. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan-Kaufmann.

Hinrich Schütze. 2017. Nonsymbolic text representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 785–796. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565. Association for Computational Linguistics.

Kateřina Veselovská and Ondřej Bojar. 2013. Czech SubLex 1.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Ulli Waltinger. 2010. Germanpolarityclues: A lexical resource for german sentiment analysis. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).

Shih-Ming Wang and Lun-Wei Ku. 2016. Antusd: A large chinese sentiment dictionary. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Chapter 3

Quantifying the Contextualization of Word Representations with Semantic Class Probing

Quantifying the Contextualization of Word Representations with Semantic Class Probing

Mengjie Zhao[†], **Philipp Dufter**[†], **Yadollah Yaghoobzadeh**[‡], **Hinrich Schütze**[†]
[†] CIS, LMU Munich, Germany [‡] Microsoft Turing, Montréal, Canada
mzhao@cis.lmu.de

Abstract

Pretrained language models achieve state-of-the-art results on many NLP tasks, but there are still many open questions about how and why they work so well. We investigate the contextualization of words in BERT. We quantify the amount of contextualization, i.e., how well words are interpreted in context, by studying the extent to which *semantic classes* of a word can be inferred from its contextualized embedding. Quantifying contextualization helps in understanding and utilizing pretrained language models. We show that the top layer representations support highly accurate inference of semantic classes; that the strongest contextualization effects occur in the lower layers; that local context is mostly sufficient for contextualizing words; and that top layer representations are more task-specific after finetuning while lower layer representations are more transferable. Finetuning uncovers task-related features, but pretrained knowledge about contextualization is still well preserved.

1 Introduction

Pretrained language models like ELMo (Peters et al., 2018a), BERT (Devlin et al., 2019), and XL-Net (Yang et al., 2019) are top performers in NLP because they learn contextualized representations, i.e., representations that reflect the interpretation of a word in context as opposed to its general meaning, which is less helpful in solving NLP tasks. As stated, pretrained language models contextualize words, is clear *qualitatively*; there has been little work on investigating contextualization, i.e., to which extent a word can be interpreted in context, *quantitatively*.

We use BERT (Devlin et al., 2019) as our pretrained language model and quantify contextualization by investigating how well BERT infers **semantic classes** (s-classes) of a word in context, e.g., the s-class *organization* for “Apple" in “Apple stock rises" vs. the s-class *food* in “Apple juice is healthy". We use s-class inference as a proxy for contextualization since accurate s-class inference reflects a successful contextualization of a word: an effective interpretation of the word in context.

We adopt the methodology of probing (Adi et al., 2016; Shi et al., 2016; Belinkov et al., 2017; Liu et al., 2019; Tenney et al., 2019b; Belinkov and Glass, 2019; Hewitt and Liang, 2019; Yaghoobzadeh et al., 2019): diagnostic classifiers are applied to pretrained language model embeddings to determine whether they encode desired syntactic or semantic features.

By probing for s-classes we quantify directly where and how contextualization happens in BERT. E.g., we find that the strongest contextual interpretation effects occur in the lower layers and that the top two layers contribute little to contextualization. We also investigate how the amount of context available affects contextualization.

In addition, since pretrained language models in practice need to be finetuned on downstream tasks (Devlin et al., 2019; Peters et al., 2019), we further investigate the interactions between finetuning and contextualization. We show that the pretrained knowledge about contextualization is well preserved in finetuned models.

We make the following **contributions**: (i) We investigate how accurately BERT interprets words in context. We find that BERT’s performance is high (almost 85% F_1), but that there is still room for improvement. (ii) We quantify how much each additional layer in BERT contributes to contextualization. We find that the strongest contextual interpretation effects occur in the lower layers. The top two layers seem to be optimized only for the pretraining objective of predicting masked words (Devlin et al., 2019) and only add small increments to contextualization. (iii) We investigate the amount of context BERT needs to exploit for interpreting a

GloVe	BERT
suits	suits
lawsuit	suited
filed	lawsuit
lawsuits	##suit
sued	lawsuits
complaint	slacks
jacket	47th

Table 1: Nearest neighbors of "suit" in GloVe and in BERT (BERT-base-uncased) wordpiece embeddings

word and find that BERT effectively integrates local context up to five words to the left and to the right (a 10-word context window). (iv) We investigate the dynamics of BERT's representations in finetuning. We find that finetuning has little effect on lower layers, suggesting that they are more easily transferable across tasks. Higher layers are strongly changed for word-level tasks like part-of-speech tagging, but less noticeably for sentence-level tasks like paraphrase classification. Finetuning uncovers task-related features, but the knowledge captured in pretraining is well preserved. We quantify these effects by s-class inference performance.

2 Motivation and Methodology

The key benefit of pretrained language models (McCann et al., 2017; Peters et al., 2018a; Radford et al., 2019; Devlin et al., 2019) is that they produce contextualized embeddings that are useful in NLP. The top layer contextualized word representations from pretrained language models are widely utilized; however, the fact that pretrained language models implement *a process of* contextualization – starting with a completely uncontextualized layer of wordpieces at the bottom – is not well studied. Table 1 gives an example: BERT's wordpiece embedding of "suit" is not contextualized: it contains several meanings of the word, including "to suit" ("be convenient"), lawsuit, and garment ("slacks"). Thus, there is no difference in this respect between BERT's wordpiece embeddings and uncontextualized word embeddings like GloVe (Pennington et al., 2014). Pretrained language models start out with an uncontextualized representation at the lowest layer, then gradually contextualize it. This is the process we analyze in this paper.

For investigating the contextualization process, one possibility is to use word senses and to tap resources like the WordNet (WN) (Fellbaum, 1998) based word sense disambiguation benchmarks of the Senseval series (Edmonds and Cotton, 2001; Snyder and Palmer, 2004; Raganato et al., 2017). However, the abstraction level in WN sense inventories has been criticized as too fine-grained (Izquierdo et al., 2009), providing limited information to applications requiring higher level abstraction. Various levels of granularity of abstraction have been explored such as WN domains (Magnini and Cavaglià, 2000), supersenses (Ciaramita and Johnson, 2003; Levine et al., 2019) and basic level concepts (Beviá et al., 2007). In this paper, we use semantic classes (s-classes) (Yarowsky, 1992; Resnik, 1993; Kohomban and Lee, 2005; Yaghoobzadeh et al., 2019) as the proxy for the meaning contents of words to study the contextualization capability of BERT. Specifically, we use the Wikipedia-based resource for Probing Semantics in Word Embeddings (Wiki-PSE) (Yaghoobzadeh et al., 2019) which is detailed in §3.1.

	words	comb's	contexts
train	35,399	62,184	2,178,895
dev	8,850	15,437	542,938
test	44,250	77,706	2,722,893

Table 2: Number of words, word-s-class combinations, and contexts per split in our probing dataset. Appendix §A.6 shows the 34 s-classes and statistics per class.

3 Probing Dataset and Task

3.1 Probing dataset

For s-class probing, we use the s-class labeled corpus Wiki-PSE (Yaghoobzadeh et al., 2019). It consists of a set of 34 s-classes, an inventory of word→s-class mappings and an English Wikipedia text corpus in which words in context are labeled with the 34 s-classes. For example, contexts of "Apple" that refer to the company are labeled with "organization". We refer to a word labeled with an s-class as a word-s-class combination, e.g., "@apple@-organization".[1]

The Wiki-PSE text corpus contains >550 million tokens, >17 million of which are annotated with an s-class. Working on the entire Wiki-PSE with BERT is not feasible, e.g., the word-s-class combination "@france@-location" has 98,582 contexts. Processing all these contexts by BERT consumes significant amounts of energy (Strubell et al., 2019; Schwartz et al., 2019) and time. Hence for each word-s-class combination, we sample a maximum of 100 contexts to speed up our experiments.

[1] In Wiki-PSE, s-class-labeled occurrences are enclosed with "@", e.g., "@apple@".

Algorithm 1 Train a classifier with type-level embeddings

```
procedure TypeSclsTrainer(Dict: word2vec, Dict: word2sclass, sclass: S, List: TrainWords):
    PosVecs, NegVecs = [], []
    for word ∈ TrainWords do
        vector = word2vec.get(word)
        sclasses = word2sclass.get(word)
        if S ∈ sclasses then
            PosVecs.append(vector)
        else
            NegVecs.append(vector)
    classifier = Classifier()
    classifier.train(PosVecs, NegVecs)
    return classifier
```

Figure 1: Training a diagnostic classifier with uncontextualized word representations for an s-class $\mathcal{S}$.

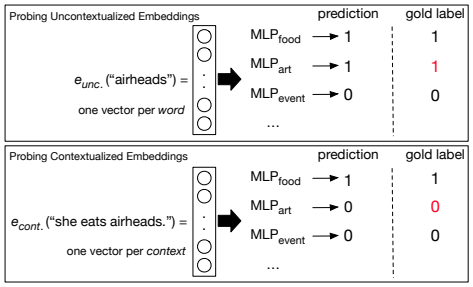


Figure 2: Setups for probing uncontextualized and contextualized embeddings. For BERT, we input a context sentence to extract the contextualized embedding of a word, e.g., "airheads"; "food" is the correct s-class label for this context.

Wiki-PSE provides a balanced train/test split; we use 20% of the training set as our development set. Table 2 gives statistics of our dataset.

3.2 Probing for semantic classes

For each of the 34 s-classes in Wiki-PSE, we train a binary classifier to diagnose if an input embedding encodes information for inferring the s-class.

3.2.1 Probing uncontextualized embeddings

We make a distinction in this paper between two different factors that contribute to BERT's performance: (i) a powerful learning architecture that gives rise to high-quality representations and (ii) contextualization in applications, i.e., words are represented as contextualized embeddings for solving NLP tasks. Here, we adopt Schuster et al. (2019)'s method of computing uncontextualized BERT embeddings (AVG-BERT-ℓ, see §4.2.1) and show that (i) alone already has a strong positive effect on performance when compared to other uncontextualized embeddings. So BERT's representation learning yields high performance, even when used in a completely uncontextualized setting.

We adopt the setup in Yaghoobzadeh et al. (2019) to probe uncontextualized embeddings – for each of the 34 s-classes, we train a binary classifier as shown in Figure 1. Table 2, column *words* shows the sizes of train/dev/test. The evaluation measure is micro F_1 over all decisions of the 34 binary classifiers.

3.2.2 Probing contextualized embeddings

We probe BERT with the same setup: a binary classifier is trained for each of the 34 s-classes; each BERT layer is probed individually.

For uncontextualized embeddings, a word has a single vector, which is either a positive or negative example for an s-class. For contextualized embeddings, the contexts of a word will typically be mixed; for example, "food" contexts (a candy) of "@airheads@" are positive but "art" contexts (a film) of "@airheads@" are negative examples for the classifier of "food". Table 2, column *contexts* shows the sizes of train/dev/test when probing BERT. Figure 2 compares our two probing setups.

In evaluation, we weight frequent word-s-class combinations (those having 100 contexts in our dataset) and the much larger number of less frequent word-s-class combinations equally. To this end, we aggregate the decisions for the contexts of a word-s-class combination. We stipulate that at least half of the contexts must be correctly classified. For example, "@airheads@-art" occurs 47 times, so we evaluate the "art" classifier as accurate for "@airheads@-art" if it classifies at least 24 contexts correctly. The final evaluation measure is micro F_1 over all 15,437 (for dev) and 77,706 (for test) decisions (see Table 2) of the 34 classifiers for the word-s-class combinations.

4 Experiments and Results

4.1 Data preprocessing

BERT uses wordpieces (Wu et al., 2016) to represent text and infrequent words are tokenized to several wordpieces. For example, "infrequent" is tokenized to "in", "##fr", "##e", and "##quent". Following He and Choi (2020), we average wordpiece embeddings to get a single vector representation of a word.[2]

[2]Some "words" in Wiki-PSE are in reality multiword phrases. Again, we average in these cases to get a single vector representation.

We limit the maximum sequence length of the context sentence input to BERT to 128. Consistent with the probing literature, we use a simple probing classifier: a 1-layer multilayer perceptron (MLP) with 1024 hidden dimensions and ReLU.

4.2 Quantifying contextualization

4.2.1 Representation learners

Six **uncontextualized embedding spaces** are evaluated: (i) PSE. A 300-dimensional embedding space computed by running skipgram with negative sampling (Mikolov et al., 2013) on the Wiki-PSE text corpus. Yaghoobzadeh et al. (2019) show that PSE outperforms other embedding spaces. (ii) Rand. An embedding space with the same vocabulary and dimension size as PSE. Vectors are drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{300})$. Rand is used to confirm that word representations indeed encode valid meaning contents that can be identified by diagnostic MLPs rather than random weights. (iii) The 300-dimensional fastText (Bojanowski et al., 2017) embeddings. (iv) GloVe. The 300-dimensional space trained on 6 billion tokens (Pennington et al., 2014). Out-of-vocabulary (OOV) words are associated with vectors drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{300})$. (v) BERTw. The 768-dimensional wordpiece embeddings in BERT. We tokenize a word with the BERT tokenizer then average its wordpiece embeddings. (vi) AVG-BERT-ℓ.[3] For an annotated word in Wiki-PSE, we average all of its contextualized embeddings from BERT layer ℓ in the Wiki-PSE text corpus. Comparing AVG-BERT-ℓ with others brings a new insight: to which extent does this "uncontextualized" variant of BERT outperform others in encoding different s-classes of a word?

Four **contextualized embedding models** are considered: (i) BERT. We use the PyTorch (Paszke et al., 2019; Wolf et al., 2019) implementation of the 12-layer BERT-base-uncased model (Wiki-PSE is uncased). (ii) P-BERT. A bag-of-word model that "contextualizes" the wordpiece embedding of an annotated word by averaging the embeddings of wordpieces of the sentence it occurs in. Comparing BERT with P-BERT reveals to which extent the self attention mechanism outperforms an average pooling practice when contextualizing words. (iii) P-fastText. Similar to P-BERT, but we use fastText word embeddings. Comparing BERT with P-fastText indicates to which extent BERT outperforms uncontextualized embedding spaces when they also have access to contextual information. (iv) P-Rand. Similar to P-BERT, but we draw word embeddings from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{300})$. Wieting and Kiela (2019) show that a random baseline has good performance in tasks like sentence classification.

[3] BERTw and AVG-BERT-ℓ have more dimensions. But Yaghoobzadeh et al. (2019) showed that different dimensionalities have a negligible impact on relative performance when probing for s-classes using MLPs as diagnostic classifiers.

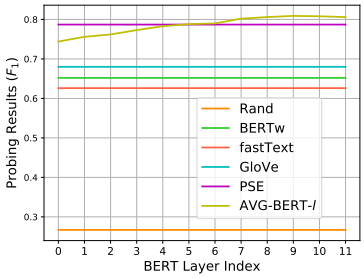


Figure 3: S-class probing results for **uncontextualized** embeddings. Results are micro F_1 on Wiki-PSE test set. Numerical values are in Table 5 in Appendix.

4.2.2 S-class inference results

Figure 3 shows **uncontextualized embedding** probing results. Comparing with random weights, all embedding spaces encode informative features helping s-class inference. BERTw delivers results similar to GloVe and fastText, demonstrating our earlier point (cf. the qualitative example in Table 1) that the lowest embedding layer of BERT is uncontextualized; several meanings of a word are conflated into a single vector.

PSE performs strongly, consistent with observations in Yaghoobzadeh et al. (2019). AVG-BERT-10 performs best among all spaces. Thus for a given word, averaging its contextualized embeddings from BERT yields a high quality type-level embedding vector, similar to "anchor words" in cross-lingual alignment (Schuster et al., 2019).

As expected, the top AVG-BERT layers outperform lower layers, given the deep architecture of BERT. Additionally, AVG-BERT-0 significantly outperforms BERTw, evidencing the importance of position embeddings and the self attention mechanism (Vaswani et al., 2017) when composing the wordpieces of a word.

Figure 4 shows **contextualized embedding** probing results. Comparing BERT layers, a clear trend can be identified: s-class inference performance increases monotonically with higher layers. This increase levels off in the top layers. Thus, the features from deeper layers improve word

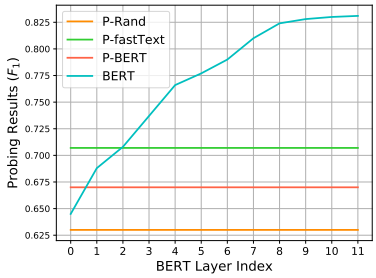


Figure 4: S-class probing results for **contextualized** embedding models. Results are micro F_1 on Wiki-PSE test set. Numerical values are in Table 6 in Appendix.

contextualization, advancing s-class inference. It also verifies previous findings: semantic tasks are mainly solved at higher layers (Liu et al., 2019; Tenney et al., 2019a). We can also observe that the strongest contextualization occurs early at lower layers – going up to layer 1 from layer 0 brings a 4% (absolute) improvement.

The very limited contextualization improvement brought by the top two layers may explain why representations from the top layers of BERT can deliver suboptimal performance on NLP tasks (Liu et al., 2019): the top layers are optimized for the pretraining objective, i.e., predicting masked words (Voita et al., 2019), not for the contextualization of words that is helpful for NLP tasks.

BERT layer 0 performs slightly worse than P-BERT, which may be due to the fact that some attention heads in lower layers of BERT attend broadly in the sentence, producing "bag-of-vector-like" representations (Clark et al., 2019), which is in fact close to the setup of P-BERT. However, starting from layer 1, BERT gradually improves and surpasses P-BERT, achieving a maximum gain of 0.16 in F_1 in layer 11. Thus, BERT knows how to better interpret the word in context, i.e., contextualize the word, when progressively going to deeper (higher) layers.

P-Rand performs strongly, but is noticeably worse than P-fastText and P-BERT. P-fastText outperforms P-BERT and BERT layers 0 and 1. We hypothesize that this may be due to the fact that fastText learns embeddings directly for words; P-BERT and BERT have to compose subwords to understand the meaning of a word, which is more challenging. Starting from layer 2, BERT outperforms P-fastText and P-BERT, illustrating the effectiveness of self attention in better integrating the information from the context into contextualized word embeddings than the average pooling practice in bag-of-word models.

Figure 3 and Figure 4 jointly illustrate the high quality of word representations computed by BERT. The BERT-derived uncontextualized AVG-BERT-ℓ representations – modeled as Schuster et al. (2019)'s anchor words – show superior capability in inferring s-classes of a word, performing best among all uncontextualized embeddings. This suggests that BERT's powerful learning architecture may be the main reason for BERT's high performance, not contextualization proper, i.e., the representation of words as contextualized embeddings on the highest layer when BERT is applied to NLP tasks. This offers intriguing possibility for creating (or distilling) strongly performing uncontextualized BERT-derived models that are more compact and more efficiently deployable.

4.2.3 Qualitative analysis

§4.2.2 quantitatively shows that BERT performs strongly in contextualizing words, thanks to its deep integration of information from the entire input sentence in each contextualized embedding. But there are scenarios where BERT fails. We identify two such cases in which the contextual information does not help s-class inference.

(i) **Tokenization**. In some domains, the annotated word and/or its context words are tokenized into several wordpieces due to their low frequency in the pretraining corpora. As a result, BERT may not be able to derive the correct composed meaning. Then the MLPs cannot identify the correct s-class from the noisy input. Consider the tokenized results of "@glutamate@-biology" and one of its contexts:

"three ne ##uro ##tra ##ns ##mit ##ters that play important roles in adolescent brain development are ***g ##lu ##tama ##te*** *. . . "*

Though "brain development" hints at a context related to "biology", this signal could be swamped by the noise in embeddings of other – especially short – wordpieces. Schick and Schütze (2020) propose a mimicking approach (Pinter et al., 2017) to help BERT understand rare words.

(ii) **Uninformative contexts**. Some contexts do not provide sufficient information related to the s-class. For example, according to probing results on BERTw, the wordpiece embedding of "goodfellas" does not encode the meaning of s-class "art" (i.e., movies); the context "Chase also said he wanted Imperioli because he had been in Goodfellas" of

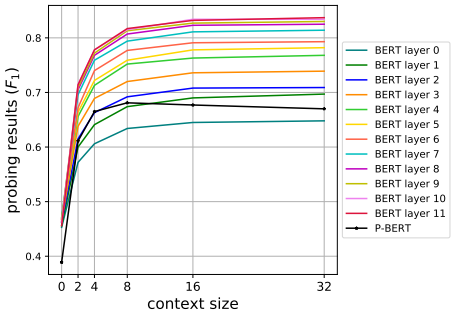


Figure 5: Probing results on the dev set with different context sizes. For BERT, performance increases with context size. Large context sizes like 16 and 32 slightly hurt performance of P-BERT.

word-s-class combination "@goodfellas@-art" is not informative enough for inferring an "art" context, yielding incorrect predictions in higher layers.

4.3 Context size

We now quantify the amount of context required by BERT for properly contextualizing words to produce accurate s-class inference results.

When probing for the s-class of word w, we define *context size* as the number of words surrounding w (left and right) in a sentence before wordpiece tokenization. For example, a context size of 5 means 5 words left, 5 words right. The context size seems to be picked heuristically in other work. Yarowsky (1992) and Gale et al. (1992) use 50 while Black (1988) uses 3–6. We experiment with a range of context sizes then compare s-class inference results. We also enclose P-BERT for comparison. Note that this experiment is different from edge probing (Tenney et al., 2019b), which takes the full sentence as input. We only make input words within the context window available to BERT and P-BERT.

4.3.1 Probing results

We report micro F_1 on Wiki-PSE dev, with context size $\in \{0, 2, 4, 8, 16, 32\}$. Context size 0 means that the input consists only of the wordpiece embeddings of the input word. Figure 5 shows results.

Comparing context sizes. Larger context sizes have higher performance for all BERT layers. Improvements are most prominent for small context sizes, e.g., 2 and 4, meaning that often local features are sufficient to contextualize words and infer s-classes, supporting Black (1988)'s design choice of 3–6. Further increasing the context size improves contextualization only marginally.

A qualitative example showing informative local features is "The Azande speak Zande, which they call Pa-Zande." In this context, the gold s-class of "Zande" is "language" (instead of "people-ethnicity", i.e., the Zande people). The MLPs for BERTw and for context size 0 for BERT fail to identify s-class "language". But the BERT MLP for context size 2 predicts "language" correctly since it includes the strong signal "speak". This context is a case of selectional restrictions (Resnik, 1993; Jurafsky and Martin, 2009), in this case possible objects of "speak".

As small context sizes already contain noticeable information contextualizing the words, we hypothesize that it may not be necessary to exploit the full context in cases where the quadratic complexity of full-sentence self attention is problematic, e.g., on edge devices. Initial results on part-of-speech tagging with the Penn Treebank (Marcus et al., 1993) in Appendix §C confirm our hypothesis. We leave more experiments to future work.

P-BERT shows a similar pattern when varying the context sizes. However, large context sizes such as 16 and 32 hurt contextualization, meaning that averaging too many embeddings results in a bag of words not specific to a particular token.

Comparing BERT layers. Higher layers of BERT yield better contextualized word embeddings. This phenomenon is more noticeable for large context sizes such as 8, 16 and 32. However for small context sizes, e.g., 0, embeddings from all layers perform similarly and badly. This means that without context information, simply passing the wordpiece embedding of a word through BERT layers does not help, suggesting that contextualization is the key ability of BERT yielding impressive performance across NLP tasks.

Again, P-BERT only outperforms layer 0 of BERT with most context sizes, suggesting that BERT layers, especially the top layers, contextualize words with abstract and informative representations, instead of naively aggregating all information within the context sentence.

4.4 Probing finetuned embeddings

We have done "classical" probing: extracting features from pretrained BERT and feeding them to diagnostic classifiers. However, pretrained BERT needs to be adapted, i.e., finetuned, for good performance on tasks (Devlin et al., 2019; Peters et al.,

	POS	SST2	MRPC	NER
Ours	.977	.928	.853	.946
Devlin et al. (2019)	n/a	.927	.867	.964

Table 3: Dev set performance of finetuning BERT (bert-base-uncased). For NER, we report micro F_1. For other tasks, we report accuracy.

2019). Thus, it is necessary to investigate how finetuning BERT affects the contextualization of words and analyze how the pretrained knowledge and probed features change.

4.4.1 Finetuning tasks

We finetune BERT on four tasks: part-of-speech (POS) tagging on the Penn Treebank (Marcus et al., 1993), named-entity recognition (NER) on the CoNLL-2003 Shared Task (Tjong Kim Sang and De Meulder, 2003), binary sentiment classification on the Stanford Sentiment Treebank (SST2) (Socher et al., 2013) and paraphrase detection on the Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005). For SST2 and MRPC, we use the GLUE train and dev sets (Wang et al., 2018). For POS, sections 0-18 of WSJ are train and sections 19-21 are dev (Collins, 2002). For NER, we use the official data splits.

Following Devlin et al. (2019), we put a linear layer on top of the pretrained BERT, then finetune all parameters. We use Adam (Kingma and Ba, 2014) with learning rate 5e-5 for 5 epochs. We save the model from the step that performs best on dev (of MRPC/SST2/POS/NER), extract representations from Wiki-PSE using this model and then report results on Wiki-PSE dev.

Table 3 reports the finetuning results. Our finetuned models perform comparably to Devlin et al. (2019) on SST2 and MRPC. Our NER result is slightly worse, this may due to the fact that Devlin et al. (2019) use "maximal document context" while we use sentence-level context of 128 max sequence length. More finetuning details are available in Appendix §B.

4.4.2 Probing results

We now quantify the contextualization of word representations from finetuned BERT models. Two setups are considered: (a) directly apply the MLPs in §4.2 (trained with pretrained embeddings) to finetuned BERT embeddings; (b) train and evaluate a new set of MLPs on the finetuned BERT embeddings.

Comparing (a) with probing results on pretrained BERT (§4.2) gives us an intuition about how many changes occurred to the knowledge captured during pretraining. Comparing (b) with §4.2 reveals whether or not the pretrained knowledge about contextualization is still preserved in finetuned models.

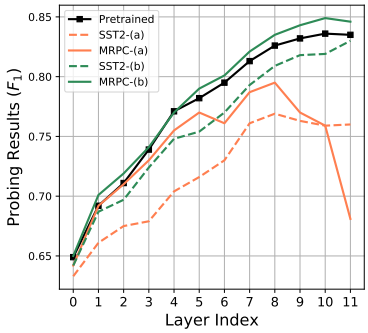
Figure 6 shows s-class probing results of finetuned BERT with setup (a) and (b). For example in (ii), layer 11 s-class inference performance of the POS-finetuned BERT decreases by 0.763 (0.835 $\rightarrow$ 0.072, from "Pretrained" to "POS-(a)") when using the MLPs from §4.2.

Comparing setup (a) and "Pretrained", we see that finetuning brings significant changes to the word representations. Finetuning on POS and NER introduces more obvious probing accuracy drops than finetuning on SST2 and MRPC. This may be due to the fact that the training objective of SST2 and MRPC takes as input only the `[CLS]` token while all words in a sentence are involved in the training objective of POS and NER.

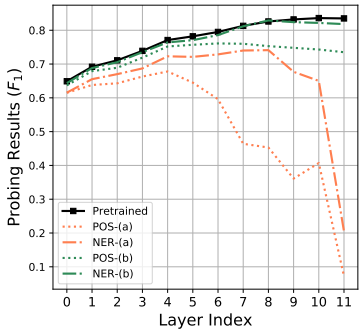
Comparing setup (b) and "Pretrained". Finetuning BERT on MRPC introduces small but consistent improvements on s-class inference. For SST2 and NER, very small s-class inference accuracy drops are observed. Finetuning on POS brings more noticeable changes. Solving POS requires more syntactic information than the other tasks, inducing BERT to "propagate" the syntactic information that is represented in lower layers to the upper layers; due to their limited capacity, the fixed-size vectors from the upper layers may lose some semantic information, yielding a more noticeable performance drop on s-class inference.

Comparing (a) and (b), we see that the knowledge about contextualizing words captured during pretraining is still well preserved after finetuning. For example, the MLPs trained with layer 11 embeddings computed by the POS-finetuned BERT still achieve a reasonably good score of 0.735 (a 0.100 drop compared with "Pretrained" – compare black and green dotted lines in Figure 6 (ii)). Thus, the semantic information needed for inferring s-classes is still present to a large extent.

Finetuning may introduce large changes (setup (a)) to the representations – similar to the projection utilized to uncover divergent information in uncontextualized word embeddings (Artetxe et al., 2018) – but relatively little information about contextualization is lost as the good performance of the newly trained MLPs shows (setup (b)). Similarly,

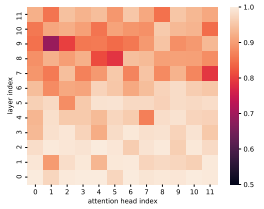


(i) MRPC and SST2

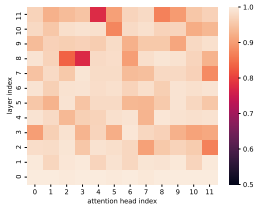


(ii) POS and NER

Figure 6: Comparing s-class inference results of pretrained BERT and BERT finetuned on MRPC, SST2, POS, and NER. "Pretrained": probing results on weight-frozen pretrained BERT in §4.2. For (a), we directly apply the MLPs in §4.2 (trained with pretrained embeddings) to finetuned BERT embeddings; for (b), we train and evaluate a new set of MLPs on the finetuned BERT embeddings.



(i) pretrained vs. POS



(ii) pretrained vs. MRPC

Figure 7: Cosine similarity of flattened self attention weights. X-axis: index of the 12 self attention heads; y-axis: layer index. Darker colors: smaller similarities, i.e., larger changes brought by finetuning.

Merchant et al. (2020) show that finetuned BERT still well preserves the probed "linguistic features" in pretrained BERT.

Comparing BERT layers. Contextualized embeddings from BERT's top layers are strongly affected by finetuning, especially for setup (a). In contrast, lower layers are more invariant and show s-class inference results similar to the pretrained model. Hao et al. (2019), Lee et al. (2019), Kovaleva et al. (2019) make similar observations: lower layer representations are more transferable across different tasks and top layer representations are more task-specific after finetuning.

Figure 7 shows the cosine similarity of the flattened self attention weights computed by pretrained, POS-, and MRPC-finetuned BERT using the dev set examples. We see that top layers are more sensitive to finetuning (darker color) while lower layers are barely changed (lighter color). Top layers have more changes for POS than for MRPC, in line with probing results in Figure 6.

5 Related Work

Interpreting deep networks. Pretrained language models (McCann et al., 2017; Peters et al., 2018a; Radford et al., 2019; Devlin et al., 2019) advance NLP by contextualized representations of words. A key goal of current research is to understand how these models work and what they represent on different layers.

Probing is a recent strand of work that investigates – via diagnostic classifiers – desired syntactic and semantic features encoded in pretrained language model representations. Shi et al. (2016) show that string-based RNNs encode syntactic information. Belinkov et al. (2017) investigate word representations at different layers in NMT. Linzen et al. (2016) assess the syntactic ability of LSTM (Hochreiter and Schmidhuber, 1997) encoders and Goldberg (2019) of BERT. Tenney et al. (2019a) find that information on POS tagging, parsing, NER, semantic roles, and coreference is represented on increasingly higher layers of BERT. Yaghoobzadeh et al. (2019) assess the disambiguation properties of type-level word representations. Liu et al. (2019) and Lin et al. (2019) investigate the linguistic knowledge encoded in BERT. Adi et al. (2016), Conneau et al. (2018), and Wieting and Kiela (2019) study sentence embedding properties via probing. Peters et al. (2018b) probe how the network architecture affects the learned vectors.

In all of these studies, probing serves to analyze representations and reveal their properties. We employ probing to investigate the contextualization of words in pretrained language models quantitatively. In addition, we exploit how finetuning affects word contextualization.

Ethayarajh (2019) quantitatively investigates contextualized embeddings, using unsupervised cosine-similarity-based evaluation. Inferring s-classes, we address a complementary set of questions because we can quantify contextualization with a uniform set of semantic classes. Brunner et al. (2020) employ token identifiability to compute the deviation of a contextualized embedding from the uncontextualized embedding. Voita et al. (2019) address this from the mutual information perspective, e.g., low mutual information between an uncontextualized embedding and its contextualized embedding can be viewed as a reflection of more contextualization. Similar observations are made: higher layer embeddings are more contextualized while lower layer embeddings are less contextualized. In contrast, we draw the observations from the perspective of s-class inference. The higher layer embeddings perform better when evaluating the semantic classes – they are better contextualized and have higher fitness to the context than the lower layer embeddings.

Two-stage NLP paradigm. Recent work (Dai and Le, 2015; Howard and Ruder, 2018; Devlin et al., 2019) introduces a "two-stage paradigm" in NLP: pretrain a language encoder on a large amount of unlabeled data via self-supervised learning, then finetune the encoder on task-specific benchmarks like GLUE (Wang et al., 2018, 2019). This transfer-learning pipeline yields good and robust results compared to models trained from scratch (Hao et al., 2019).

In this work, we shed light on how BERT's pretrained knowledge about contextualization changes during finetuning by comparing s-class inference ability of pretrained and finetuned models. Merchant et al. (2020) analyze BERT models finetuned on different downstream tasks with the edge probing suite (Tenney et al., 2019b) and make similar observations as us. They focus on "linguistic features" while we focus on the contextualization of words.

6 Conclusion

We presented a quantitative study of the contextualization of words in BERT by investigating BERT's semantic class inference capabilities. We focused on two key factors for successful contextualization by BERT: layer index and context size. By comparing pretrained and finetuned models, we showed that word-level tasks like part-of-speech tagging bring more noticeable changes than sentence-level tasks like paraphrase classification; and top layers of BERT are more sensitive to the finetuning objective than lower layers. We also found that BERT's pretrained knowledge about contextualizing words is still well retained after finetuning.

We showed that exploiting the full context may be unnecessary in applications where the quadratic complexity of full-sentence attention is problematic. Future work may evaluate this phenomenon on more datasets and downstream tasks.

Acknowledgements. We thank the anonymous reviewers for the insightful comments and suggestions. This work was funded by the European Research Council (ERC #740516) and a Zentrum Digitalisierung.Bayern fellowship award.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Mikel Artetxe, Gorka Labaka, Iñigo Lopez-Gazpio, and Eneko Agirre. 2018. Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 282–291, Brussels, Belgium. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Rubén Izquierdo Beviá, Armando Suárez Cueto, and Germán Rigau Claramunt. 2007. Exploring the automatic selection of basic level concepts.

Ezra Black. 1988. An experiment in computational discrimination of english word senses. *IBM Journal of research and development*, 32(2):185–194.

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.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers. In *International Conference on Learning Representations*.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 168–175. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning.

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 Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

William A Gale, Kenneth W Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. 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 4134–4143, Hong Kong, China. Association for Computational Linguistics.

Han He and Jinho D. Choi. 2020. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'20. Best Paper Candidate.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. 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 2733–2743, Hong Kong, China. Association for Computational Linguistics.

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. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Rubén Izquierdo, Armando Suárez, and German Rigau. 2009. An empirical study on class-based word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 389–397. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Upali Sathyajith Kohomban and Wee Sun Lee. 2005. Learning semantic classes for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 34–41, Ann Arbor, Michigan. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. 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 4356–4365, Hong Kong, China. Association for Computational Linguistics.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning.

Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2019. Sensebert: Driving some sense into bert. *arXiv preprint arXiv:1908.05646*.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside bert's linguistic knowledge. *arXiv preprint arXiv:1906.01698*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Bernardo Magnini and Gabriela Cavaglià. 2000. Integrating subject field codes into WordNet. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to bert embeddings during fine-tuning?

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. *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. 2018a. 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.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

Philip Resnik. 1993. Semantic classes and syntactic ambiguity. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8766–8774. AAAI Press.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. 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 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green ai.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain. Association for Computational Linguistics.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. 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 4387–4397, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: 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 R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. *arXiv preprint arXiv:1901.10444*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Transformers: State-of-the-art natural language processing.

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*.

Yadollah Yaghoobzadeh, Katharina Kann, T. J. Hazen, Eneko Agirre, and Hinrich Schütze. 2019. Probing for semantic classes: Diagnosing the meaning content of word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5740–5753, Florence, Italy. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *ArXiv*, abs/1906.08237.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

A Reproducibility Checklist

A.1 Computing infrastructure

All experiments are conducted on GeForce GTX 1080 Ti and GeForce GTX 1080.

A.2 Number of parameters

We use a set of 34 binary MLPs to conduct our probing task. Each MLP has input dimension 768, hidden dimension 1024 and output dimension 2. As a result, the total number of parameters is 26,843,204. For finetuning, we use the BERT-base-uncased model containing about 110 million parameters (https://github.com/google-research/bert).

A.3 Validation performance

Following Table 5 and Table 6 report the validation performance of probing uncontextualized and contextualized embeddings.

A.4 Evaluation metric

Our evaluation is the micro F_1 over all decisions of the 34 probing classifiers. More details are available in §3.2 of the main paper.

A.5 Hyperparameter search

For probing tasks, we do not conduct hyperparameter search since our goal is to analyze the contextualization. The probing classifiers are trained with learning rate 1e-3 and 400 epochs. For finetuning BERT, we do not search hyperparameters but directly adopt the setup in Devlin et al. (2019) as shown in Table 4.

A.6 Datasets

List of the 34 semantic classes (s-classes), number of word-s-class combinations and contexts per s-class in the sampled Wiki-PSE (Yaghoobzadeh et al., 2019) are listed in Table 8. Some annotated contexts in Wiki-PSE are also displayed in Table 9. The Wiki-PSE developed by Yaghoobzadeh et al. (2019) is publicly available at https://github.com/yyaghoobzadeh/WIKI-PSE.

When finetuning BERT, we use the GLUE (Wang et al., 2018) splits of MRPC and SST2 from https://gluebenchmark.com/. Our POS dataset is from the linguistic data consortium (LDC). For NER (Tjong Kim Sang and De Meulder, 2003), we use the official shared task dataset: https://www.clips.uantwerpen.be/conll2003/ner/.

	POS	SST2	MRPC	NER
batch size	150	200	350	32
learning rate	5e-5	5e-5	5e-5	5e-5
max epoch	5	5	5	5
max sequence length	128	128	128	128

Table 4: Hyperparameters for finetuning.

B Finetuning Details

Hyperparameters in Table 4 are used when we finetune BERT on POS, NER, SST2, and MRPC. For SST2 and MRPC, we use the embedding of `[CLS]` as the representation of the sentence (pair). For POS and NER, we use the embedding of the last wordpiece of the word as Liu et al. (2019).

A plain Adam (Kingma and Ba, 2014) optimizer is used and we did not use strategies like learning rate warmup and layer-wise learning rate (Howard and Ruder, 2018) during finetuning to avoid potential side effects to ensure a clear comparison of different BERT layers.

C Context Sizes in POS

We investigate how the findings from §4.3 in the main paper transfer to downstream tasks. To this end we perform standard finetuning of BERT for different tasks, but we prune the attention matrix to a context size of length k. That is we apply a mask on the attention matrix such that each word can only attend to k left and k right words. This has great benefits as it reduces the memory and computation requirements from $\mathcal{O}(n^2)$ to $\mathcal{O}(nk)$ where n is the sequence length. We only consider part-of-speech tagging as for sentence pair classification tasks such as SST2 and MRPC this is not a sensible approach.

Table 7 confirms that small context windows are sufficient to achieve full performance for POS-tagging. This indicates that the finding from the main paper (i.e., local context is sufficient for BERT to achieve a high degree of contextualization) is to some degree applicable to a downstream tasks, as well. Note that the median sentence length in the Penn Treebank dataset is 25 words (the number of wordpieces even higher). Thus masking the context to the next 4 or 8 words does indeed reduce the available context words. In future work we plan to investigate this effect not only during finetuning but also during pretraining.

	Standard Embeddings					AVG-BERT-ℓ											
	Rand	BERTw	fastText	GloVe	PSE	0	1	2	3	4	5	6	7	8	9	10	11
dev	.269	.653	.625	.681	.790	.746	.759	.764	.775	.786	.791	.794	.805	.811	.812	.813	.809
test	.267	.652	.626	.680	.787	.744	.756	.762	.773	.783	.788	.790	.802	.806	.809	.808	.806

Table 5: S-class probing results for **uncontextualized** embeddings. Numbers are micro F_1 on Wiki-PSE. Our result (0.787 on PSE-test) is consistent with Yaghoobzadeh et al. (2019). Additionally, for the top 6 layers {6, 7, 8, 9, 10, 11} of AVG-BERT, we repeat the experiments 5 times with random seed in {1, 2, 3, 4, 5}. Mean and standard deviation on test per layer are: {.791±.001, .801±.001, .807±.001, .808±.001, .808±.001, .805±.001}.

	Bag-of-word context			BERT Layer											
	P-Rand	P-fastText	P-BERT	0	1	2	3	4	5	6	7	8	9	10	11
dev	.637	.707	.672	.649	.692	.711	.739	.771	.782	.795	.813	.826	.832	.836	.835
test	.630	.707	.670	.645	.688	.708	.737	.766	.777	.790	.810	.824	.828	.830	.831

Table 6: S-class probing results for **contextualized** embedding models. Numbers are micro F_1 on Wiki-PSE.

Context size	POS
0	.886
2	.973
4	.975
8	.976
16	.977
32	.977
All	.977

Table 7: POS accuracy on dev for different context sizes.

	train		dev		test	
semantic classes	comb's	contexts	comb's	contexts	comb's	contexts
location	13,474	618,932	3,408	152,470	16,859	776,848
person	15,423	617,270	3,744	151,005	19,212	765,655
organization	9,556	332,063	2,496	88,682	11,915	411,716
art	7,428	201,529	1,854	52,295	9,192	247,481
event	3,515	87,735	900	21,566	4,404	108,963
broadcast-program	2,287	67,261	530	15,062	2,828	84,343
title	1,429	43,041	311	9,646	1,792	56,333
product	3,121	49,076	766	13,438	3,808	61,585
living-thing	1,302	35,595	320	9,035	1,702	46,040
people-ethnicity	754	27,573	181	6,699	951	35,332
language	671	14,842	145	3,147	824	20,308
broadcast-network	325	12,392	80	3,036	362	13,006
time	157	7,765	39	1,997	192	9,984
religion-religion	192	6,461	45	1,760	265	9,719
award	251	7,589	61	1,776	301	8,877
internet-website	88	2,466	21	645	141	3,851
god	246	7,306	52	1,998	340	11,810
education-educational-degree	97	3,282	24	901	142	4,833
food	381	7,805	112	2,003	480	9,514
computer-programming-language	105	2,739	29	402	123	2,677
metropolitan-transit-transit-line	285	5,603	76	1,259	382	6,948
transit	135	3,781	26	628	186	4,305
finance-currency	127	3,107	30	548	166	3,388
disease	163	2,619	33	381	260	4,385
chemistry	170	3,350	43	1,254	195	3,858
body-part	135	1,901	31	415	156	2,591
finance-stock-exchange	27	617	3	5	51	795
law	23	474	6	54	27	535
medicine-medical-treatment	77	886	7	124	106	1,803
medicine-drug	50	1,023	7	54	72	1,157
broadcast-tv-channel	45	564	14	210	74	1,264
medicine-symptom	55	752	15	97	72	1,172
biology	49	485	15	118	63	911
visual-art-color	41	1,011	13	228	63	906
total	62,184	2,178,895	15,437	542,938	77,706	2,722,893

Table 8: Number of word-s-class combinations and contexts for each of the 34 semantic classes in Wiki-PSE.

word	word-s-class combination	contexts
roberta	@roberta@-art	this recording is also available on cd paired with @roberta@-art . ... to star as huckleberry haines in the jerome kern / dorothy fields musical @roberta@-art .
	@roberta@-location	there are also learning centers in eatonton , forsyth , gray , jeffersonville , and @roberta@-location . ... the concurrency curves to a nearly due north routing and enters @roberta@-location .
	@roberta@-person	ken williams : along with wife @roberta@-person , founded on-line systems after working at ibm mystery house is an adventure game released in 7 by @roberta@-person and ken williams for the apple ii .
larch	@larch@-comp-prog-lang	wing has been a leading member of the formal methods community , especially in the area of @larch@-comp-prog-lang . a major contribution was his involvement with the @larch@-comp-prog-lang approach to formal specification with ...
	@larch@-living-thing	the more recent plantings include @larch@-living-thing and pine . these consist mainly of oak , alder , @larch@-living-thing and corsican pine .

Table 9: Example contexts of the annotated word "roberta" and "larch".

Chapter 4

Masking as an Efficient Alternative to Finetuning for Pretrained Language Models

Masking as an Efficient Alternative to Finetuning for Pretrained Language Models

Mengjie Zhao[†*], **Tao Lin**[‡*], **Fei Mi**[‡], **Martin Jaggi**[‡], **Hinrich Schütze**[†]
[†] LMU Munich, Germany [‡] EPFL, Switzerland
mzhao@cis.lmu.de, {tao.lin, fei.mi, martin.jaggi}@epfl.ch

Abstract

We present an efficient method of utilizing pretrained language models, where we learn selective binary masks for pretrained weights in lieu of modifying them through finetuning. Extensive evaluations of masking BERT, RoBERTa, and DistilBERT on eleven diverse NLP tasks show that our masking scheme yields performance comparable to finetuning, yet has a much smaller memory footprint when several tasks need to be inferred. Intrinsic evaluations show that representations computed by our binary masked language models encode information necessary for solving downstream tasks. Analyzing the loss landscape, we show that masking and finetuning produce models that reside in minima that can be connected by a line segment with nearly constant test accuracy. This confirms that masking can be utilized as an efficient alternative to finetuning.

1 Introduction

Finetuning a large pretrained language model like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019) often yields competitive or even state-of-the-art results on NLP benchmarks (Wang et al., 2018, 2019). Given an NLP task, standard finetuning stacks a linear layer on top of the pretrained language model and then updates all parameters using mini-batch SGD. Various aspects like brittleness (Dodge et al., 2020) and adaptiveness (Peters et al., 2019) of this two-stage transfer learning NLP paradigm (Dai and Le, 2015; Howard and Ruder, 2018) have been studied.

Despite the simplicity and impressive performance of finetuning, the prohibitively large number of parameters to be finetuned, e.g., 340 million in BERT-large, is a major obstacle to wider deployment of these models. The large memory footprint of finetuned models becomes more prominent when multiple tasks need to be solved – several copies of the millions of finetuned parameters have to be saved for inference.

Recent work (Gaier and Ha, 2019; Zhou et al., 2019) points out the potential of searching neural architectures within a fixed model, as an alternative to optimizing the model weights for downstream tasks. Inspired by these results, we present *masking*, a simple yet efficient scheme for utilizing pretrained language models. Instead of directly updating the pretrained parameters, we propose to *select* weights important to downstream NLP tasks while *discarding* irrelevant ones. The selection mechanism consists of a set of binary masks, one learned per downstream task through end-to-end training.

We show that masking, when being applied to pretrained language models like BERT, RoBERTa, and DistilBERT (Sanh et al., 2019), achieves performance comparable to finetuning in tasks like part-of-speech tagging, named-entity recognition, sequence classification, and reading comprehension. This is surprising in that a simple subselection mechanism that does not change any weights is competitive with a training regime – finetuning – that can change the value of every single weight. We conduct detailed analyses revealing important factors and possible reasons for the desirable performance of masking.

Masking is parameter-efficient: only a set of 1-bit binary masks needs to be saved per task after training, instead of all 32-bit float parameters in finetuning. This small memory footprint enables deploying pretrained language models for solving multiple tasks on edge devices. The compactness of masking also naturally allows parameter-efficient ensembles of pretrained language models.

Our **contributions**: (i) We introduce *masking*, a new scheme for utilizing pretrained language models by learning selective masks for pretrained weights, as an efficient alternative to finetuning.

* Equal contribution.

We show that masking is applicable to models like BERT/RoBERTa/DistilBERT, and produces performance on par with finetuning. (ii) We carry out extensive empirical analysis of masking, shedding light on factors critical for achieving good performance on eleven diverse NLP tasks. (iii) We study the binary masked language models' loss landscape and language representations, revealing potential reasons why masking has task performance comparable to finetuning.

2 Related Work

Two-stage NLP paradigm. Pretrained language models (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019b; Yang et al., 2019; Radford et al., 2019) advance NLP with contextualized representation of words. Finetuning a pretrained language model (Dai and Le, 2015; Howard and Ruder, 2018) often delivers competitive performance partly because pretraining leads to a better initialization across various downstream tasks than training from scratch (Hao et al., 2019). However, finetuning on individual NLP tasks is not parameter-efficient. Each finetuned model, typically consisting of hundreds of millions of floating point parameters, needs to be saved individually. Stickland and Murray (2019) use projected attention layers with multi-task learning to improve efficiency of finetuning BERT. Houlsby et al. (2019) insert adapter modules to BERT to improve memory efficiency. The inserted modules alter the forward pass of BERT, hence need to be carefully initialized to be close to identity.

We propose to directly pick parameters appropriate to a downstream task, by learning selective binary masks via end-to-end training. Keeping the pretrained parameters untouched, we solve several downstream NLP tasks with minimal overhead.

Binary networks and network pruning. Binary masks can be trained using the "straight-through estimator" (Bengio et al., 2013; Hinton, 2012). Hubara et al. (2016), Rastegari et al. (2016), Hubara et al. (2017), *inter alia*, apply this technique to train efficient binarized neural networks. We use this estimator to train selective masks for pretrained language model parameters.

Investigating the lottery ticket hypothesis (Frankle and Carbin, 2018) of network pruning (Han et al., 2015a; He et al., 2018; Liu et al., 2019c; Lee et al., 2019; Lin et al., 2020), Zhou et al. (2019) find that applying binary masks to a neural network is a form of training the network. Gaier and Ha (2019) propose to search neural architectures for reinforcement learning and image classification tasks, without any explicit weight training. This work inspires our masking scheme (which can be interpreted as implicit neural architecture search (Liu et al., 2019c)): applying the masks to a pretrained language model is similar to finetuning, yet is much more parameter-efficient.

Perhaps the closest work, Mallya et al. (2018) apply binary masks to CNNs and achieve good performance in computer vision. We learn selective binary masks for pretrained language models in NLP and shed light on factors important for obtaining good performance. Mallya et al. (2018) explicitly update weights in a task-specific classifier layer. In contrast, we show that end-to-end learning of selective masks, consistently for both the pretrained language model and a randomly initialized classifier layer, achieves good performance. Radiya-Dixit and Wang (2020) investigate finetuning of BERT by employing a number of techniques, including what they call sparsification, a method similar to masking. Their focus is analysis of finetuning BERT whereas our goal is to provide an efficient alternative to finetuning.

3 Method

3.1 Background on Transformer and finetuning

The encoder of the Transformer architecture (Vaswani et al., 2017) is ubiquitously used when pretraining large language models. We briefly review its architecture and then present our masking scheme. Taking BERT-base as an example, each one of the 12 transformer blocks consists of (i) four linear layers[1] $\mathbf{W}_K$, $\mathbf{W}_Q$, $\mathbf{W}_V$, and $\mathbf{W}_{AO}$ for computing and outputting the self attention among input wordpieces (Wu et al., 2016). (ii) two linear layers $\mathbf{W}_I$ and $\mathbf{W}_O$ feeding forward the word representations to the next transformer block.

More concretely, consider an input sentence $\mathbf{X} \in \mathbb{R}^{N \times d}$ where N is the maximum sentence length and d is the hidden dimension size. $\mathbf{W}_K$, $\mathbf{W}_Q$, and $\mathbf{W}_V$ are used to compute transformations of $\mathbf{X}$:

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$

[1] We omit the bias terms for brevity.

and the self attention of $\mathbf{X}$ is computed as:

$$\text{Attention}(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}.$$

The attention is then transformed by $\mathbf{W}_{AO}$, and subsequently fed forward by $\mathbf{W}_I$ and $\mathbf{W}_O$ to the next transformer block.

When finetuning on a downstream task like sequence classification, a linear classifier layer $\mathbf{W}_T$, projecting from the hidden dimension to the output dimension, is randomly initialized. Next, $\mathbf{W}_T$ is stacked on top of a pretrained linear layer $\mathbf{W}_P$ (the *pooler layer*). All parameters are then updated to minimize the task loss such as cross-entropy.

3.2 Learning the mask

Given a pretrained language model, we do not finetune, i.e., we do not update the pretrained parameters. Instead, we *select* a subset of the pretrained parameters that is critical to a downstream task while *discarding* irrelevant ones with binary masks. We associate each linear layer $\mathbf{W}^l \in \{\mathbf{W}^l_K, \mathbf{W}^l_Q, \mathbf{W}^l_V, \mathbf{W}^l_{AO}, \mathbf{W}^l_I, \mathbf{W}^l_O\}$ of the l-th transformer block with a real-valued matrix $\mathbf{M}^l$ that is randomly initialized from a uniform distribution and has the same size as $\mathbf{W}^l$. We then pass $\mathbf{M}^l$ through an element-wise thresholding function (Hubara et al., 2016; Mallya et al., 2018), i.e., a binarizer, to obtain a binary mask $\mathbf{M}^l_{\text{bin}}$ for $\mathbf{W}^l$:

$$(m^l_{\text{bin}})_{i,j} = \begin{cases} 1 & \text{if } m^l_{i,j} \geq \tau \\ 0 & \text{otherwise} \end{cases}, \qquad (1)$$

where $m^l_{i,j} \in \mathbf{M}^l$, i, j indicate the coordinates of the 2-D linear layer and τ is a global thresholding hyperparameter.

In each forward pass of training, the binary mask $\mathbf{M}^l_{\text{bin}}$ (derived from $\mathbf{M}^l$ via Eq. 1) selects weights in a pretrained linear layer $\mathbf{W}^l$ by Hadamard product:

$$\hat{\mathbf{W}}^l := \mathbf{W}^l \odot \mathbf{M}^l_{\text{bin}}.$$

In the corresponding backward pass of training, with the associated loss function $\mathcal{L}$, we cannot backpropagate through the binarizer, since Eq. 1 is a hard thresholding operation and the gradient with respect to $\mathbf{M}^l$ is zero almost everywhere. Similar to the treatment[2] in Bengio et al. (2013); Hubara et al. (2016); Lin et al. (2020), we use $\frac{\partial \mathcal{L}(\hat{\mathbf{W}}^l)}{\partial \mathbf{M}^l_{\text{bin}}}$ as a noisy estimator of $\frac{\partial \mathcal{L}(\hat{\mathbf{W}}^l)}{\partial \mathbf{M}^l}$ to update $\mathbf{M}^l$, i.e.:

$$\mathbf{M}^l \leftarrow \mathbf{M}^l - \eta \frac{\partial \mathcal{L}(\hat{\mathbf{W}}^l)}{\partial \mathbf{M}^l_{\text{bin}}}, \qquad (2)$$

where η refers to the step size. Hence, the whole structure can be trained end-to-end.

We learn a set of binary masks for an NLP task as follows. Recall that each linear layer $\mathbf{W}^l$ is associated with a $\mathbf{M}^l$ to obtain a masked linear layer $\hat{\mathbf{W}}^l$ through Eq. 1. We randomly initialize an additional linear layer with an associated $\mathbf{M}^l$ and stack it on top of the pretrained language model. We then update each $\mathbf{M}^l$ through Eq. 2 with the task objective during training.

After training, we pass each $\mathbf{M}^l$ through the binarizer to obtain $\mathbf{M}^l_{\text{bin}}$, which is then saved for future inference. Since $\mathbf{M}^l_{\text{bin}}$ is binary, it takes only $\approx 3\%$ of the memory compared to saving the 32-bit float parameters in a finetuned model. Also, we will show that many layers – in particular the embedding layer – do not have to be masked. This further reduces memory consumption of masking.

3.3 Configuration of masking

Our masking scheme is motivated by the observation: the pretrained weights form a good initialization (Hao et al., 2019), yet a few steps of adaptation are still needed to produce competitive performance for a specific task. However, not every pretrained parameter is necessary for achieving reasonable performance, as suggested by the field of neural network pruning (LeCun et al., 1990; Hassibi and Stork, 1993; Han et al., 2015b). We now investigate two configuration choices that affect how many parameters are "eligible" for masking.

Initial sparsity of $\mathbf{M}^l_{\text{bin}}$. As we randomly initialize our masks from uniform distributions, the sparsity of the binary mask $\mathbf{M}^l_{\text{bin}}$ in the mask initialization phase controls how many pretrained parameters in a layer $\mathbf{W}^l$ are assumed to be irrelevant to the downstream task. Different initial sparsity rates entail different optimization behaviors.

It is crucial to better understand how the initial sparsity of a mask impacts the training dynamics and final model performance, so as to generalize our masking scheme to broader domains and tasks. In §5.1, we investigate this aspect in detail. In practice, we fix τ in Eq. 1 while adjusting the uniform distribution to achieve a target initial sparsity.

[2] Bengio et al. (2013); Hubara et al. (2016) describe it as the "straight-through estimator", and Lin et al. (2020) provide convergence guarantee with error feedback interpretation.

Which layers to mask. Different layers of pretrained language models capture distinct aspects of a language during pretraining, e.g., Tenney et al. (2019) find that information on part-of-speech tagging, parsing, named-entity recognition, semantic roles, and coreference is encoded on progressively higher layers of BERT. It is hard to know a priori which types of NLP tasks have to be addressed in the future, making it non-trivial to decide layers to mask. We study this factor in §5.2.

We do not learn a mask for the lowest embedding layer, i.e., the uncontextualized wordpiece embeddings are completely "selected", for all tasks. The motivation is two-fold. (i) The embedding layer weights take up a large part, e.g., almost 21% (23m/109m) in BERT-base-uncased, of the total number of parameters. Not having to learn a selective mask for this layer reduces memory consumption. (ii) Pretraining has effectively encoded context-independent general meanings of words in the embedding layer (Zhao et al., 2020). Hence, learning a selective mask for this layer is unnecessary. Also, we do not learn masks for biases and layer normalization parameters as we did not observe a positive effect on performance.

4 Datasets and Setup

Datasets. We present results for masking BERT, RoBERTa, and DistilBERT in part-of-speech tagging, named-entity recognition, sequence classification, and reading comprehension.

We experiment with **part-of-speech tagging** (POS) on Penn Treebank (Marcus et al., 1993), using Collins (2002)'s train/dev/test split. For **named-entity recognition** (NER), we conduct experiments on the CoNLL-2003 NER shared task (Tjong Kim Sang and De Meulder, 2003).

For **sequence classification**, the following GLUE tasks (Wang et al., 2018) are evaluated: Stanford Sentiment Treebank (SST2) (Socher et al., 2013), Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019), Recognizing Textual Entailment (RTE) (Dagan et al., 2005), and Question Natural Language Inference (QNLI) (Rajpurkar et al., 2016).

In addition, we experiment on sequence classification datasets that have publicly available test sets: the 6-class question classification dataset TREC (Voorhees and Tice, 2000), the 4-class news classification dataset AG News (AG) (Zhang et al., 2015), and the binary Twitter sentiment classification task SemEval-2016 4B (SEM) (Nakov et al., 2016).

We experiment with **reading comprehension** on SWAG (Zellers et al., 2018) using the official data splits. We report Matthew's correlation coefficient (MCC) for CoLA, micro-F1 for NER, and accuracy for the other tasks.

Setup. Due to resource limitations and in the spirit of environmental responsibility (Strubell et al., 2019; Schwartz et al., 2019), we conduct our experiments on the base models: BERT-base-uncased, RoBERTa-base, and DistilBERT-base-uncased. Thus, the BERT/RoBERTa models we use have 12 transformer blocks (0–11 indexed) producing 768-dimension vectors; the DistilBERT model we use has the same dimension but contains 6 transformer blocks (0–5 indexed). We implement our models in PyTorch (Paszke et al., 2019) with the HuggingFace framework (Wolf et al., 2019).

Throughout all experiments, we limit the maximum length of a sentence (pair) to be 128 after wordpiece tokenization. Following Devlin et al. (2019), we use the Adam (Kingma and Ba, 2014) optimizer of which the learning rate is a hyperparameter while the other parameters remain default. We carefully tune the learning rate for each setup: the tuning procedure ensures that the best learning rate does not lie on the border of our search grid, otherwise we extend the grid accordingly. The initial grid is {1e-5, 3e-5, 5e-5, 7e-5, 9e-5}.

For sequence classification and reading comprehension, we use `[CLS]` as the representation of the sentence (pair). Following Devlin et al. (2019), we formulate NER as a tagging task and use a linear output layer, instead of a conditional random field layer. For POS and NER experiments, the representation of a tokenized word is its last wordpiece (Liu et al., 2019a; He and Choi, 2020). Note that a 128 maximum length of a sentence for POS and NER means that some word-tag annotations need to be excluded. Appendix §A shows our reproducibility checklist containing more implementation and preprocessing details.

5 Experiments

5.1 Initial sparsity of binary masks

We first investigate how initial sparsity percentage (i.e., fraction of zeros) of the binary mask $\mathbf{M}^l_{\text{bin}}$ influences performance of a binary masked language model on downstream tasks. We experiment on four tasks, with initial sparsities in {1%, 3%, 5%,

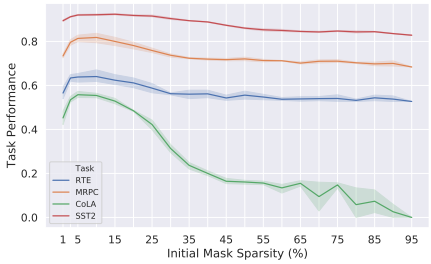


Figure 1: Dev set performance of masking BERT when selecting different amounts of pretrained parameters.

10%, 15%, 20%, . . . , 95%}. All other hyperparameters are controlled: learning rate is fixed to 5e-5; batch size is 32 for relatively small datasets (RTE, MRPC, and CoLA) and 128 for SST2. Each experiment is repeated four times with different random seeds {1, 2, 3, 4}. In this experiment, all transformer blocks, the pooler layer, and the classifier layer are masked.

Figure 1 shows that masking achieves decent performance without hyperparameter search. Specifically, (i) a large initial sparsity removing most pretrained parameters, e.g., 95%, leads to bad performance for the four tasks. This is due to the fact that the pretrained knowledge is largely discarded. (ii) Gradually decreasing the initial sparsity improves task performance. Generally, an initial sparsity in 3% $\sim$ 10% yields reasonable results across tasks. Large datasets like SST2 are less sensitive than small datasets like RTE. (iii) Selecting almost all pretrained parameters, e.g., 1% sparsity, hurts task performance. Recall that a pretrained model needs to be adapted to a downstream task; masking achieves adaptation by learning selective masks – preserving too many pretrained parameters in initialization impedes the optimization.

5.2 Layer-wise behaviors

Neural network layers present heterogeneous characteristics (Zhang et al., 2019) when being applied to tasks. For example, syntactic information is better represented at lower layers while semantic information is captured at higher layers in ELMo (Peters et al., 2018). As a result, simply masking all transformer blocks (as in §5.1) may not be ideal.

We investigate the task performance when applying the masks to different BERT layers. Figure 2 presents the optimal task performance when masking only a subset of BERT's transformer blocks on MRPC, CoLA, and RTE. Different amounts and indices of transformer blocks are masked: "bottom-up" and "top-down" indicate to mask the targeted amount of transformer blocks, either from bottom or top of BERT.

We can observe that (i) in most cases, top-down masking outperforms bottom-up masking when initial sparsity and the number of masked layers are fixed. Thus, it is reasonable to select all pretrained weights in lower layers, since they capture general information helpful and transferable to various tasks (Liu et al., 2019a; Howard and Ruder, 2018). (ii) For bottom-up masking, increasing the number of masked layers gradually improves performance. This observation illustrates dependencies between BERT layers and the learning dynamics of masking: provided with selected pretrained weights in lower layers, higher layers need to be given flexibility to select pretrained weights accordingly to achieve good task performance. (iii) In top-down masking, CoLA performance increases when masking a growing number of layers while MRPC and RTE are not sensitive. Recall that CoLA tests linguistic acceptability that typically requires both syntactic and semantic information[3]. All of BERT layers are involved in representing this information, hence allowing more layers to change should improve performance.

5.3 Comparing finetuning and masking

We have investigated two factors – initial sparsity (§5.1) and layer-wise behaviors (§5.2) – that are important in masking pretrained language models. Here, we compare the performance and memory consumption of masking and finetuning.

Based on observations in §5.1 and §5.2, we use 5% initial sparsity when applying masking to BERT, RoBERTa, and DistilBERT. We mask the transformer blocks 2–11 in BERT/RoBERTa and 2–5 in DistilBERT. $\mathbf{W}_P$ and $\mathbf{W}_T$ are always masked. Note that this global setup is surely suboptimal for some model-task combinations, but our goal is to illustrate the effectiveness and the generalization ability of masking. Hence, conducting extensive hyperparameter search is unnecessary.

For AG and QNLI, we use batch size 128. For the other tasks we use batch size 32. We search the optimal learning rate per task as described in §4,

[3]For example, to distinguish acceptable caused-motion constructions (e.g., "the professor talked us into a stupor") from inacceptable ones (e.g., "water talked it into red"), both syntactic and semantic information need to be considered (Goldberg, 1995).

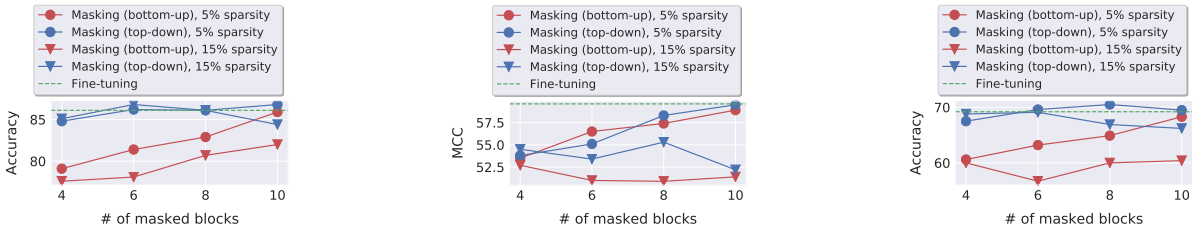


Figure 2: The impact of masking different transformer blocks of BERT for MRPC (left), CoLA (middle), and RTE (right). The number of masked blocks is shown on the x-axis; that number is either masked "bottom-up" or "top-down". More precisely, a bottom-up setup (red) masking 4 blocks means we mask the transformer blocks $\{0, 1, 2, 3\}$; a top-down setup (blue) masking 4 blocks means we mask the transformer blocks $\{8, 9, 10, 11\}$. $\mathbf{W}_P$ and $\mathbf{W}_T$ are always masked.

		MRPC 3.5k	SST2 67k	CoLA 8.5k	RTE 2.5k	QNLI 108k	SEM 4.3k	TREC 4.9k	AG 96k	POS 38k	NER 15k	SWAG 113k
BERT	Finetuning	86.1 ± 0.8	93.3 ± 0.2	59.6 ± 0.8	69.2 ± 2.7	91.0 ± 0.6	86.6 ± 0.3	96.4 ± 0.2	94.4 ± 0.1	97.7 ± 0.0	94.6 ± 0.2	80.9 ± 1.7
	Masking	86.8 ± 1.1	93.2 ± 0.5	59.5 ± 0.1	69.5 ± 3.0	91.3 ± 0.4	85.9 ± 0.5	96.0 ± 0.4	94.2 ± 0.0	97.7 ± 0.0	94.5 ± 0.1	80.3 ± 0.1
RoBERTa	Finetuning	89.8 ± 0.5	95.0 ± 0.3	62.1 ± 1.7	78.2 ± 1.1	92.9 ± 0.2	90.2 ± 0.5	96.2 ± 0.4	94.7 ± 0.0	98.1 ± 0.0	94.9 ± 0.1	83.4 ± 0.8
	Masking	88.5 ± 1.1	94.5 ± 0.3	60.3 ± 1.3	69.2 ± 2.1	92.4 ± 0.1	90.1 ± 0.1	95.9 ± 0.5	94.5 ± 0.1	98.0 ± 0.0	93.9 ± 0.1	82.1 ± 0.2
DistilBERT	Finetuning	85.4 ± 0.5	91.6 ± 0.4	55.1 ± 0.3	62.2 ± 3.0	89.0 ± 0.8	85.9 ± 0.2	95.7 ± 0.6	94.2 ± 0.1	97.6 ± 0.0	94.1 ± 0.1	72.5 ± 0.2
	Masking	86.0 ± 0.3	91.3 ± 0.3	53.1 ± 0.7	61.6 ± 1.5	89.2 ± 0.2	86.6 ± 0.6	95.9 ± 0.6	94.2 ± 0.1	97.6 ± 0.0	94.1 ± 0.2	71.0 ± 0.0

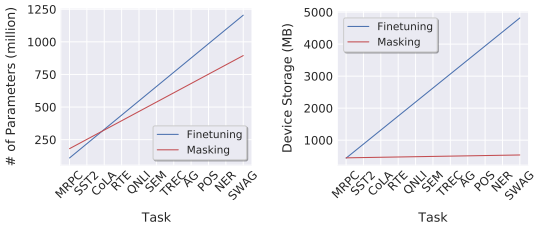
Table 1: Dev set task performances (%) of masking and finetuning. Each experiment is repeated four times with different random seeds and we report mean and standard deviation. Numbers below dataset name (second row) are the size of training set. For POS and NER, we report the number of sentences.

and they are shown in Appendix §A.4.

Performance comparison. Table 1 reports performance of masking and finetuning on the dev set for the eleven NLP tasks. We observe that applying masking to BERT/RoBERTa/DistilBERT yields performance comparable to finetuning. We observe a performance drop[4] on RoBERTa-RTE. RTE has the smallest dataset size (train: 2.5k; dev: 0.3k) among all tasks – this may contribute to the imperfect results and large variances.

Our BERT-NER results are slightly worse than Devlin et al. (2019). This may be due to the fact that "maximal document context" is used by Devlin et al. (2019) while we use sentence-level context of 128 maximum sequence length[5].

Rows "Single" in Table 2 compare performance of masking and finetuning BERT on the test set of SEM, TREC, AG, POS, and NER. The same setup and hyperparameter searching as Table 1 are used, the best hyperparameters are picked on the dev set. Results from Sun et al. (2019); Palogiannidi et al. (2016) are included as a reference. Sun et al. (2019) employ optimizations like layer-wise learning rate, producing slightly better performance than ours. Palogiannidi et al. (2016) is the best performing system on task SEM (Nakov et al., 2016). Again, masking yields results comparable to finetuning.



(a) Number of parameters. (b) Memory consumption.

Figure 3: The accumulated number of parameters and memory required by finetuning and masking to solve an increasing number of tasks.

Memory comparison. Having shown that task performance of masking and finetuning is comparable, we next demonstrate one key strength of masking: memory efficiency. We take BERT-base-uncased as our example. Figure 3 shows the accumulated number of parameters in million and memory in megabytes (MB) required when an increasing number of downstream tasks need to be solved using finetuning and masking. Masking re-

[4]Similar observations were made: DistilBERT has a 10% accuracy drop on RTE compared to BERT-base (Sanh et al., 2019); Sajjad et al. (2020) report unstableness on MRPC and RTE when applying their model reduction strategies.

[5]Similar observations were made: https://github.com/huggingface/transformers/issues/64

		SEM	TREC	AG	POS	NER	Memory (MB)
Masking	Single	12.03	3.30	5.62	2.34	9.85	447
	Ensem.	11.52	3.20	5.28	2.12	9.19	474
Finetun.	Single	11.87	3.80	5.66	2.34	9.85	438
	Ensem.	11.73	2.80	5.17	2.29	9.23	1752
Sun et al. (2019)		n/a	2.80	5.25	n/a	n/a	n/a
Palogiannidi et al. (2016)		13.80	n/a	n/a	n/a	n/a	n/a

Table 2: Error rate (%) on test set and model size comparison. Single: the averaged performance of four models with different random seeds. Ensem.: ensemble of the four models.

quires a small overhead when solving a single task but is much more efficient than finetuning when several tasks need to be inferred. Masking saves a single copy of a pretrained language model containing 32-bit float parameters for all the eleven tasks and a set of 1-bit binary masks for each task. In contrast, finetuning saves every finetuned model so the memory consumption grows linearly.

Masking naturally allows light ensembles of models. Rows "Ensem." in Table 2 compare ensembled results and model size. We consider the ensemble of predicted (i) labels; (ii) logits; (iii) probabilities. The best ensemble method is picked on dev and then evaluated on test. Masking only consumes 474MB of memory – much smaller than 1752MB required by finetuning – and achieves comparable performance. Thus, masking is also much more memory-efficient than finetuning in an ensemble setting.

6 Discussion

6.1 Intrinsic evaluations

§5 demonstrates that masking is an efficient alternative to finetuning. Now we analyze properties of the representations computed by binary masked language models with intrinsic evaluation.

One intriguing property of finetuning, i.e., stacking a classifier layer on top of a pretrained language model then update all parameters, is that a linear classifier layer suffices to conduct reasonably accurate classification. This observation implies that the configuration of data points, e.g., sentences with positive or negative sentiment in SST2, should be close to linearly separable in the hidden space. Like finetuning, masking also uses a linear classifier layer. Hence, we hypothesize that upper layers in binary masked language models, even without explicit weight updating, also create a hidden space in which data points are close to linearly separable.

Figure 4 uses t-SNE (Maaten and Hinton, 2008)

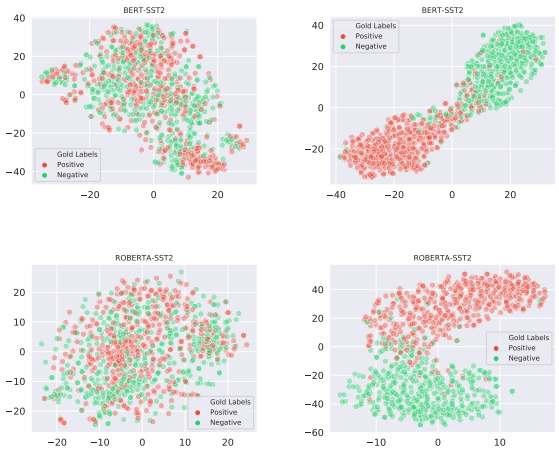


Figure 4: t-SNE visualization of the representation of `[CLS]` computed by the topmost transformer block in pretrained (left), finetuned (top right), and masked (bottom right) BERT/RoBERTa. We use `scikit-learn` (Pedregosa et al., 2011) and default t-SNE parameters.

	SST2	SEM
SST2	41.8	-13.4
SEM	20.0	11.5

(a) Masking

	SST2	SEM
SST2	41.8	-10.1
SEM	18.9	12.2

(b) Finetuning

Table 3: Generalization on dev (%) of binary masked and finetuned BERT. Row: training dataset; Column: evaluating dataset. Numbers are improvements against the majority-vote baseline: 50.9 for SST2 and 74.4 for SEM. Results are averaged across four random seeds.

to visualize the representation of `[CLS]` computed by the topmost transformer block in pretrained, finetuned, and masked BERT/RoBERTa, using the dev set examples of SST2. The pretrained models' representations (left) are clearly not separable since the model needs to be adapted to downstream tasks. The sentence representations computed by the finetuned (top right) and the binary masked (bottom right) encoder are almost linearly separable and consistent with the gold labels. Thus, a linear classifier is expected to yield reasonably good classification accuracy. This intrinsic evaluation illustrates that binary masked models extract good representations from the data for the downstream NLP task.

6.2 Properties of the binary masked models

Do binary masked models generalize? Figure 4 shows that a binary masked language model produces proper representations for the classifier layer and hence performs as well as a finetuned model. Here, we are interested in verifying that

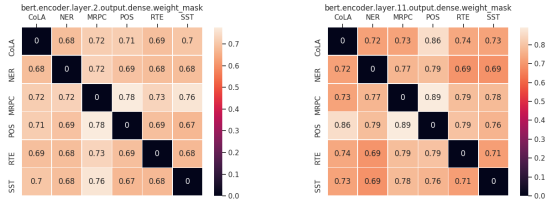


Figure 5: Scores s of two sets of masks, trained with two different tasks, of layer $\mathbf{W}_O$ in transformer blocks 2 (left) and 11 (right) in BERT. A large s means that the two masks are dissimilar.

the binary masked model does indeed solve downstream tasks by learning meaningful representations – instead of exploiting spurious correlations that generalize poorly (Niven and Kao, 2019; McCoy et al., 2019). To this end, we test if the binary masked mode is generalizable to other datasets of the same type of downstream task. We use the two sentiment classification datasets: SST2 and SEM. We simply evaluate the model masked or finetuned on SST2 against the dev set of SEM and vice versa. Table 3 reports the results against the majority-vote baseline. The finetuned and binary masked models of SEM generalize well on SST2, showing $\approx 20\%$ improvement against the majority-vote baseline.

On the other hand, we observe that the knowledge learned on SST2 does not generalize to SEM, for both finetuning and masking. We hypothesize that this is because the Twitter domain (SEM) is much more specific than movie reviews (SST2). For example, some Emojis or symbols like ":)" reflecting strong sentiment do not occur in SST2, resulting in unsuccessful generalization. To test our hypothesis, we take another movie review dataset IMDB (Maas et al., 2011), and directly apply the SST2-finetuned- and SST2-binary-masked- models on it. Masking and finetuning achieve accuracy 84.79% and 85.25%, which are comparable and both outperform the baseline 50%, demonstrating successful knowledge transfer.

Thus, finetuning and masking yield models with similar generalization ability. The binary masked models indeed create representations that contain valid information for downstream tasks.

Analyzing masks. We study the dissimilarity between masks learned by different BERT layers and downstream tasks. For the initial and trained binary masks $\mathbf{M}_{\text{bin}}^{t,init}$ and $\mathbf{M}_{\text{bin}}^{t,trained}$ of a layer trained on task $t \in \{t1, t2\}$. We compute:

$$s = \frac{\left\|\mathbf{M}_{\text{bin}}^{t1,trained} - \mathbf{M}_{\text{bin}}^{t2,trained}\right\|_1}{\left\|\mathbf{M}_{\text{bin}}^{t1,trained} - \mathbf{M}_{\text{bin}}^{t1,init}\right\|_1 + \left\|\mathbf{M}_{\text{bin}}^{t2,trained} - \mathbf{M}_{\text{bin}}^{t2,init}\right\|_1},$$

where $\|\mathbf{W}\|_1 = \sum_{i=1}^{m}\sum_{j=1}^{n}|w_{i,j}|$. Note that for the same random seed, $\mathbf{M}_{\text{bin}}^{t1,init}$ and $\mathbf{M}_{\text{bin}}^{t2,init}$ are the same. The dissimilarity s measures the difference between two masks as a fraction of all changes brought about by training. Figure 5 shows that, after training, the dissimilarities of masks of higher BERT layers are larger than those of lower BERT layers. Similar observations are made for finetuning: top layer weights in finetuned BERT are more task-specific (Kovaleva et al., 2019). The figure also shows that the learned masks for downstream tasks tend to be dissimilar to each other, even for similar tasks. For a given task, there exist different sets of masks (initialized with different random seeds) yielding similar performance. This observation is similar to the results of evaluating the lottery ticket hypothesis on BERT (Prasanna et al., 2020; Chen et al., 2020): a number of subnetworks exist in BERT achieving similar task performance.

6.3 Loss landscape

Training complex neural networks can be viewed as searching for good minima in the highly non-convex landscape defined by the loss function (Li et al., 2018). Good minima are typically depicted as points at the bottom of different locally convex valleys (Keskar et al., 2016; Draxler et al., 2018), achieving similar performance. In this section, we study the relationship between the two minima obtained by masking and finetuning.

Recent work analyzing the loss landscape suggests that the local minima in the loss landscape reached by standard training algorithms can be connected by a simple path (Garipov et al., 2018; Gotmare et al., 2018), e.g., a Bézier curve, with low task loss (or high task accuracy) along the path. We are interested in testing if the two minima found by finetuning and masking can be easily connected in the loss landscape. To start with, we verify the task performance of an interpolated model $\mathbf{W}(\gamma)$ on the line segment between a finetuned model $\mathbf{W}_0$ and a binary masked model $\mathbf{W}_1$:

$$\mathbf{W}(\gamma) = \mathbf{W}_0 + \gamma(\mathbf{W}_1 - \mathbf{W}_0), 0 \leq \gamma \leq 1\,.$$

We conduct experiments on MRPC and SST2 with the best-performing BERT and RoBERTa

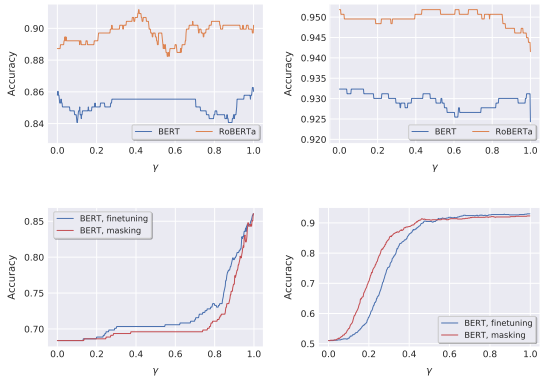


Figure 6: Mode connectivity results on MRPC (left) and SST2 (right). Top images: dev set accuracy of an interpolated model between the two minima found by finetuning ($\gamma = 0$) and masking ($\gamma = 1$). Bottom images: accuracy of an interpolated model between pretrained ($\gamma = 0$) and finetuned/masked ($\gamma = 1$) BERT.

models obtained in Table 1 (same seed and training epochs); Figure 6 (top) shows the results of mode connectivity, i.e., the evolution of the task accuracy along a line connecting the two candidate minima.

Surprisingly, the interpolated models on the line segment connecting a finetuned and a binary masked model form a high accuracy path, indicating the extremely well-connected loss landscape. Thus, masking finds minima on the same connected low-loss manifold as finetuning, confirming the effectiveness of our method. Also, we show in Figure 6 (bottom) for the line segment between the pretrained BERT and a finetuned/masked BERT, that mode connectivity is not solely due to an overparameterized pretrained language model. Bézier curves experiments show similar results, cf. Appendix §B.

7 Conclusion

We have presented masking, an efficient alternative to finetuning for utilizing pretrained language models like BERT/RoBERTa/DistilBERT. Instead of updating the pretrained parameters, we only train one set of binary masks per task to select critical parameters. Extensive experiments show that masking yields performance comparable to finetuning on a series of NLP tasks. Leaving the pretrained parameters unchanged, masking is much more memory efficient when several tasks need to be solved. Intrinsic evaluations show that binary masked models extract valid and generalizable representations for downstream tasks. Moreover, we demonstrate that the minima obtained by finetuning and masking can be easily connected by a line segment, confirming the effectiveness of applying masking to pretrained language models. Our code is available at: `https://github.com/ptlmasking/maskbert`.

Future work may explore the possibility of applying masking to the pretrained multilingual encoders like mBERT (Devlin et al., 2019) and XLM (Conneau and Lample, 2019). Also, the binary masks learned by our method have low sparsity such that inference speed is not improved. Developing methods improving both memory and inference efficiency without sacrificing task performance can open the possibility of widely deploying the powerful pretrained language models to more NLP applications.

Acknowledgments

We thank the anonymous reviewers for the insightful comments and suggestions. This work was funded by the European Research Council (ERC #740516), SNSF grant 200021_175796, as well as a Google Focused Research Award.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors,

Advances in Neural Information Processing Systems 28, pages 3079–3087. Curran Associates, Inc.

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.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. 2018. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks.

Adam Gaier and David Ha. 2019. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, pages 5365–5379.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798.

Adele E Goldberg. 1995. *Construction grammar*. Wiley.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015a. Learning both weights and connections for efficient neural network. In *NeurIPS - Advances in Neural Information Processing Systems*, pages 1135–1143.

Song Han, Jeff Pool, John Tran, and William Dally. 2015b. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. 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 4143–4152, Hong Kong, China. Association for Computational Linguistics.

Babak Hassibi and David G. Stork. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 164–171. Morgan-Kaufmann.

Han He and Jinho D. Choi. 2020. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'20. Best Paper Candidate.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2234–2240.

Geoffrey Hinton. 2012. Neural networks for machine learning.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4107–4115. Curran Associates, Inc.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. 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 4356–4365, Hong Kong, China. Association for Computational Linguistics.

Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *ICLR - International Conference on Learning Representations*.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399.

Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. 2020. Dynamic model pruning with feedback. In *International Conference on Learning Representations*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019c. Rethinking the value of network pruning. In *ICLR - International Conference on Learning Representations*.

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*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *The European Conference on Computer Vision (ECCV)*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif, Nikolaos Malandrakis, Haris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos. 2016. Tweester at SemEval-2016 task 4: Sentiment analysis in twitter using semantic-affective model adaptation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 155–163, San Diego, California. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

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.

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.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT plays the lottery, all tickets are winning.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Evani Radiya-Dixit and Xin Wang. 2020. How fine can fine-tuning be? learning efficient language models. volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443, Online. PMLR.

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.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*.

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*.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green ai.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995, Long Beach, California, USA. PMLR.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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 Voorhees and Dawn Tice. 2000. The trec-8 question answering track evaluation. *Proceedings of the 8th Text Retrieval Conference*.

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 3261–3275.

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. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

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*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2019. Are all layers created equal? *arXiv preprint arXiv:1902.01996*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Mengjie Zhao, Philipp Dufter, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2020. Quantifying the contextualization of word representations with semantic class probing. In *Findings of EMNLP*.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pages 3592–3602.

A Reproducibility Checklist

A.1 Computing infrastructure

All experiments are conducted on following GPU models: Tesla V100, GeForce GTX 1080 Ti, and GeForce GTX 1080. We use per-GPU batch size 32. Thus, experiments comparing masking and finetuning on QNLI and AG take 4 GPUs and all the other tasks use a single GPU.

A.2 Number of parameters

In §5.3 we thoroughly compare the number of parameters and memory consumption of finetuning and masking. Numerical values are in Table 8.

A.3 Validation performance

The dev set performance of Table 2 is covered in Table 1. We report Matthew's correlation coefficient (MCC) for CoLA, micro-F1 for NER, and accuracy for the other tasks. We use the evaluation functions in `scikit-learn` (Pedregosa et al., 2011) and `seqeval` (https://github.com/chakki-works/seqeval).

A.4 Hyperparameter search

The only hyperparameter we searched is learning rate, for both masking and finetuning, according to the setup discussion in §4. The optimal values are in Table 4.

A.5 Datasets

For GLUE tasks, we use the official datasets from the benchmark https://gluebenchmark.com/. For TREC and AG, we download the datasets developed by Zhang et al. (2015), which are available at here. Note that this link is provided by Zhang et al. (2015) and also used by Sun et al. (2019). For SEM, we obtain the dataset from the official SemEval website: http://alt.qcri.org/semeval2016/task4/. For NER, we use the official dataset: https://www.clips.uantwerpen.be/conll2003/ner/. We obtain our POS dataset from the linguistic data consortium (LDC). We use the official dataset of SWAG (Zellers et al., 2018): https://github.com/rowanz/swagaf/tree/master/data.

For POS, sections 0-18 of WSJ are train, sections 19-21 are dev, and sections 22-24 are test (Collins, 2002). We use the official train/dev/test splits of all the other datasets.

To preprocess the datasets, we use the tokenizers provided by the `Transformers` package (Wolf et al., 2019) to convert the raw dataset to the formats required by BERT/RoBERTa/DistilBERT. Since wordpiece tokenization is used, there is no out-of-vocabulary words.

Since we use a maximum sequence length of 128, our preprocessing steps exclude some word-tag annotations in POS and NER. For POS, after wordpiece tokenization, we see 1 sentence in dev and 2 sentences in test have more than 126 (the `[CLS]` and `[SEP]` need to be considered) wordpieces. As a result, we exclude 5 annotated words in dev and 87 annotated words in test. Similarly, for NER (which is also formulated as a tagging task following Devlin et al. (2019)), we see 3 sentences in dev and 1 sentence in test have more than 126 wordpieces. As a result, we exclude 27 annotated words in dev and 8 annotated words in test.

The number of examples in dev and test per task is shown in following Table 5.

B More on Mode Connectivity

Following the mode connectivity framework proposed in Garipov et al. (2018), we parameterize the path joining two minima using a Bézier curve. Let $\mathbf{w}_0$ and $\mathbf{w}_{n+1}$ be the parameters of the models trained from finetuning and masking. Then, an n-bend Bézier curve connecting $\mathbf{w}_0$ and $\mathbf{w}_{n+1}$, with n trainable intermediate models $\theta = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$, can be represented by $\phi_\theta(t)$, such that $\phi_\theta(0) = \mathbf{w}_0$ and $\phi_\theta(1) = \mathbf{w}_{n+1}$, and

$$\phi_\theta(t) = \sum_{i=0}^{n+1} \binom{n+1}{i} (1-t)^{n+1-i} t^i \mathbf{w}_i \,.$$

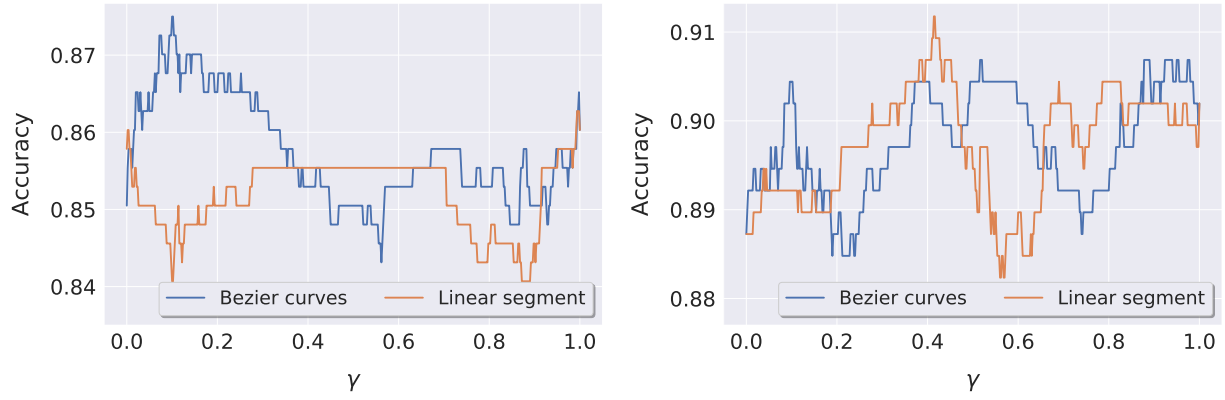
We train a 3-bend Bézier curve by minimizing the loss $\mathbb{E}_{t\sim U[0,1]} \mathcal{L}\left(\phi_\theta(t)\right)$, where $U[0,1]$ is the uniform distribution in the interval $[0,1]$. Monte Carlo method is used to estimate the gradient of this expectation-based function and gradient-based optimization is used for the minimization. The results are illustrated in Figure 7. Masking implicitly performs gradient descent, analogy to the weights update achieved by finetuning; the observations complement our arguments in the main text.

C More Empirical Results

Ensemble results of RoBERTa and DistilBERT. Following Table 6 shows the single and ensemble results of RoBERTa and DistilBERT on the test set of SEM, TREC, AG, POS, and NER.

		MRPC	SST2	CoLA	RTE	QNLI	POS	NER	SWAG	SEM	TREC	AG
BERT	Finetuning	5e-5	1e-5	3e-5	5e-5	3e-5	3e-5	3e-5	7e-5	1e-5	3e-5	3e-5
	Masking	1e-3	5e-4	9e-4	1e-3	7e-4	5e-4	7e-4	1e-4	7e-5	1e-4	5e-4
RoBERTa	Finetuning	3e-5	1e-5	1e-5	7e-6	1e-5	9e-6	3e-5	1e-5	7e-6	9e-6	3e-5
	Masking	3e-4	9e-5	3e-4	3e-4	1e-4	3e-4	3e-4	1e-4	3e-4	5e-4	5e-4
DistilBERT	Finetuning	3e-5	7e-5	3e-5	3e-5	3e-5	3e-5	1e-5	7e-6	1e-5	3e-5	3e-5
	Masking	9e-4	7e-4	9e-4	9e-4	1e-3	7e-4	7e-4	3e-4	3e-4	9e-4	1e-3

Table 4: The optimal learning rate on different tasks for BERT/RoBERTa/DistilBERT. We perform finetuning/masking on all tasks for 10 epochs with early stopping of 2 epochs.



(a) BERT (b) RoBERTa

Figure 7: The accuracy on MRPC dev set, as a function of the point on the curves $\phi_\theta(\gamma)$, connecting the two minima found by finetuning (left, $\gamma=0$) and masking (right, $\gamma=1$).

	Dev	Test
MRPC	408	n/a
SST2	872	n/a
CoLA	1,042	n/a
RTE	277	n/a
QNLI	5,732	n/a
SEM	1,325	10,551
TREC	548	500
AG	24,000	7,600
POS	135,105	133,082
NER	51,341	46,425
SWAG	20,006	n/a

Table 5: Number of examples in dev and test per task. For POS and NER, we report the number of words.

			SEM	TREC	AG	POS	NER
RoBERTa	Masking	Single	11.12	3.15	5.06	2.11	11.03
		Ensem.	10.54	2.40	4.55	2.11	10.57
	Finetun.	Single	10.74	3.00	5.10	2.00	10.43
		Ensem.	10.74	2.60	4.50	1.96	9.54
DistilBERT	Masking	Single	11.89	3.70	5.71	2.39	10.40
		Ensem.	11.60	3.00	5.29	2.54	9.86
	Finetun.	Single	11.94	3.30	5.42	2.39	10.18
		Ensem.	11.48	3.00	4.84	2.29	9.74

Table 6: Error rate (%) on test set of tasks by RoBERTa and DistilBERT. Single: the averaged performance of four models with different random seeds. Ensem.: ensemble of the four models.

D Numerical Values of Plots

D.1 Layer-wise behaviors

Table 7 details the numerical values of Figure 2.

D.2 Memory consumption

Table 8 details the numerical values of Figure 3.

	MRPC	RTE	CoLA
Finetuning (BERT + classifier)	0.861 ± 0.008	0.692 ± 0.027	0.596 ± 0.015
Masking (BERT 00-11 + classifier, initial sparsity 5%)	0.862 ± 0.015	0.673 ± 0.036	0.592 ± 0.004
Masking (BERT 00-11 + classifier, initial sparsity 15%)	0.825 ± 0.039	0.626 ± 0.040	0.522 ± 0.027
Masking (BERT 02-11 + classifier, initial sparsity 5%)	0.868 ± 0.011	0.695 ± 0.030	0.595 ± 0.010
Masking (BERT 02-11 + classifier, initial sparsity 15%)	0.844 ± 0.024	0.662 ± 0.021	0.556 ± 0.012
Masking (BERT 04-11 + classifier, initial sparsity 5%)	0.861 ± 0.004	0.705 ± 0.037	0.583 ± 0.005
Masking (BERT 04-11 + classifier, initial sparsity 15%)	0.861 ± 0.009	0.669 ± 0.014	0.553 ± 0.014
Masking (BERT 06-11 + classifier, initial sparsity 5%)	0.862 ± 0.004	0.696 ± 0.027	0.551 ± 0.006
Masking (BERT 06-11 + classifier, initial sparsity 15%)	0.868 ± 0.008	0.691 ± 0.033	0.534 ± 0.016
Masking (BERT 08-11 + classifier, initial sparsity 5%)	0.848 ± 0.016	0.675 ± 0.034	0.538 ± 0.014
Masking (BERT 08-11 + classifier, initial sparsity 15%)	0.851 ± 0.009	0.688 ± 0.022	0.545 ± 0.005
Masking (BERT 00-09 + classifier, initial sparsity 5%)	0.859 ± 0.012	0.683 ± 0.031	0.589 ± 0.011
Masking (BERT 00-09 + classifier, initial sparsity 15%)	0.820 ± 0.052	0.604 ± 0.021	0.514 ± 0.016
Masking (BERT 00-07 + classifier, initial sparsity 5%)	0.829 ± 0.032	0.649 ± 0.053	0.574 ± 0.012
Masking (BERT 00-07 + classifier, initial sparsity 15%)	0.807 ± 0.042	0.600 ± 0.027	0.509 ± 0.004
Masking (BERT 00-05 + classifier, initial sparsity 5%)	0.814 ± 0.033	0.632 ± 0.058	0.565 ± 0.027
Masking (BERT 00-05 + classifier, initial sparsity 15%)	0.781 ± 0.032	0.567 ± 0.030	0.510 ± 0.025
Masking (BERT 00-03 + classifier, initial sparsity 5%)	0.791 ± 0.026	0.606 ± 0.027	0.535 ± 0.034
Masking (BERT 00-03 + classifier, initial sparsity 15%)	0.776 ± 0.035	0.600 ± 0.019	0.527 ± 0.014

Table 7: Numerical value of the layer-wise behavior experiment. We train for 10 epochs with mini-batch size 32. The learning rate is finetuned using the mean results on four different random seeds.

	Number of Parameters		Memory Usage (Kilobytes)	
	Finetuning	Masking	Finetuning	Masking
Pretrained	109,482,240		437,928.96	
MRPC	+ 1,536	+ 1,536 + 71,368,704 + 1,536	+ 6.144	+ 6.144 + 8,921.088 + 0.192
SST2	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
CoLA	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
RTE	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
QNLI	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
SEM	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
TREC	+ 4,608 + 109,482,240	+ 4,608 + 71,368,704 + 4,608	+ 18.432 + 437,928.96	+ 18.432 + 8,921.088 + 0.576
AG	+ 3,072 + 109,482,240	+ 3,072 + 71,368,704 + 3,072	+ 12.288 + 437,928.96	+ 12.288 + 8,921.088 + 0.384
POS	+ 37,632 + 109,482,240	+ 37,632 + 71,368,704 + 37,632	+ 150.528 + 437,928.96	+ 150.528 + 8,921.088 + 4.704
NER	+ 6,912 + 109,482,240	+ 6,912 + 71,368,704 + 6,912	+ 27.648 + 437,928.96	+ 27.648 + 8,921.088 + 0.864
SWAG	+ 768 + 109,482,240	+ 768 + 71,368,704 + 768	+ 3.072 + 437,928.96	+ 3.072 + 8,921.088 + 0.096

Table 8: Model size comparison when applying masking and finetuning. Numbers are based on BERT-base-uncased. Note that our masking scheme enables sharing parameters across tasks: tasks with the same number of output dimension can use the same classifier layer.

Chapter 5

Discrete and Soft Prompting for Multilingual Models

Discrete and Soft Prompting for Multilingual Models

Mengjie Zhao and **Hinrich Schütze**
CIS, LMU Munich, Germany
mzhao@cis.lmu.de

Abstract

It has been shown for English that discrete and soft prompting perform strongly in few-shot learning with pretrained language models (PLMs). In this paper, we show that discrete and soft prompting perform better than finetuning in multilingual cases: Crosslingual transfer and in-language training of multilingual natural language inference. For example, with 48 English training examples, finetuning obtains 33.74% accuracy in crosslingual transfer, barely surpassing the majority baseline (33.33%). In contrast, discrete and soft prompting outperform finetuning, achieving 36.43% and 38.79%. We also demonstrate good performance of prompting with training data in multiple languages other than English.

1 Introduction

Prompting strongly outperforms finetuning (Devlin et al., 2019) when adapting pretrained language models (PLMs; Devlin et al. (2019); Conneau et al. (2020)) to downstream tasks in the low-resource regime (Brown et al., 2020; Schick and Schütze, 2021; Gao et al., 2020; Tam et al., 2021; Le Scao and Rush, 2021), i.e., *few-shot learning*, a more realistic scenario than having tens of thousands of annotations, *even for English* (Yu et al., 2018; Yin et al., 2020; Ram et al., 2021).

In contrast to finetuning, which learns discriminative classifiers for tasks like natural language inference (NLI; Dagan et al. (2006); Bowman et al. (2015)), prompting reformulates the classification task to generative text-to-text (Raffel et al., 2020) or cloze-style (McCann et al., 2018; Brown et al., 2020) queries which are given to a PLM to answer. For example, the NLI task of assigning premise "They whinnied, eyes wide" and hypothesis "Their eyes were open wide" to class "entailment" can be reformulated as:

They whinnied, eyes wide . Question: Their eyes were open wide ? Answer: _ .

The PLM is requested to fill in, for the blank (_), the word "yes", which is mapped to "entailment".

Prompting makes a human description of the task available in learning. Also, "filling in the blank" is well aligned with the pretraining objective (masked/autoregressive language modelling (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019)), likely to deliver better performance in few-shot learning (Ram et al., 2021).

In this paper, we investigate the effectiveness of prompting in multilingual tasks, which – despite the success of prompting in English – is largely unexplored. We address two main research questions: (RQ1) Does the strong few-shot performance of prompting transfer to other languages from English? (RQ2) As the cost of few-shot non-English annotations is affordable (Garrette and Baldridge, 2013; Lauscher et al., 2020; Zhao et al., 2021), can we directly prompt PLMs in languages other than English or do we have to go through the (generally best resourced) intermediary of English?

In this work, we systematically compare two popular prompting methods – discrete and soft prompting – with finetuning in the few-shot multilingual NLI task and show that *prompting is superior*: (i) The strong few-shot learning performance of prompting transfers to other languages from English: It outperforms finetuning in crosslingual transfer (RQ1; §5.1). (ii) Directly querying the multilingual PLM with few-shot non-English prompts achieves competitive performance, without relying on crosslingual transfer from English (RQ2; §5.2).

2 Related Work

GPT3 (Brown et al., 2020) succeeds in few-shot NLU tasks with "in-context learning": A natural language prompt describing the NLU task is prepended to an input example; GPT3 is then capable of making accurate predictions *without updating its parameters*. However, the number of

parameters in GPT3 is prohibitively large (175B).

Integrating gradient descent into prompting, smaller (w.r.t. GPT3) PLMs also achieve good few-shot performance. Like GPT3, **discrete prompting** uses natural language to describe NLU tasks. Schick and Schütze (2021), Tam et al. (2021), Le Scao and Rush (2021) use human-designed prompts. Gao et al. (2020) leverage T5 (Raffel et al., 2020) to generate prompts. Shin et al. (2020) use extra training data to search tokens for constructing the prompts. Discrete prompting naturally inherits interpretability from the task descriptions.

Soft prompting relaxes the constraint that a prompt needs to be composed of discrete tokens. Instead, it learns the prompt in the continuous space with SGD. Qin and Eisner (2021) and Zhong et al. (2021) learn soft prompts eliciting more knowledge (Petroni et al., 2019) from PLMs than discrete prompts. Similar to soft prompting but with the PLM being frozen, Li and Liang (2021) propose prefix-tuning to encourage PLMs to solve generation tasks with high parameter-efficiency (Houlsby et al., 2019; Zhao et al., 2020). Lester et al. (2021) demonstrate that soft prompting benefits from scaling up the number of PLM parameters. Liu et al. (2021) show that GPT (Radford et al., 2019) can solve NLU tasks (Wang et al., 2019) with soft prompting.

All of this work focuses on English. We show that discrete and soft prompting perform better than finetuning in few-shot crosslingual natural language inference (XNLI; Conneau et al. (2018)) with multilingual PLMs (XLM-RoBERTa; Conneau et al. (2020)). We conduct experiments on NLI because it is one of the most representative and challenging NLU tasks (Dagan et al., 2006; Bowman et al., 2015), and has been commonly used in prior work on prompting.

3 Method

3.1 Finetuning

We follow the standard finetuning method (Devlin et al., 2019): A linear classifier layer is initialized and stacked on top of the PLM; the whole model is then trained on the few-shot NLI dataset (§4).

3.2 Prompting

Discrete prompting (DP). Following Schick and Schütze (2021), Le Scao and Rush (2021), we reformulate the NLI examples (cf. example in §1) into cloze-style questions using a human-designed prompt. Specifically, we ask the PLM to fill in the blank (_) in sentence:

Premise . `Question:` Hypothesis `?` `Answer:` _ .

Premise and Hypothesis are a pair of sentences from the NLI dataset. The gold labels are mapped to words in the PLM vocabulary. Concretely, we use following mapping (**verbalizer**; Schick and Schütze (2021)): "entailment"→ "yes"; "contradiction"→ "no"; "neutral"→ "maybe". The optimization objective is to minimize the cross-entropy loss between the predicted and the gold words representing the three classes.

Soft prompting (**SP**; Li and Liang (2021); Qin and Eisner (2021); Zhong et al. (2021); Liu et al. (2021)) leverages prompts containing "pseudo tokens" that are not part of the PLM vocabulary. In this work, we ask a PLM to fill in the blank (_) in sentence:

Premise . Hypothesis ? <v_1><v_2><v_3><v_4> _ .

where each <v_i>, $i \in \{1, 2, 3, 4\}$ is associated with a randomly initialized trainable vector (in the PLM's lowest embedding layer) $\boldsymbol{v}_i \in \mathbb{R}^d$, where d is the hidden dimension size of the embedding layer. Directly using $\boldsymbol{v}_i$ yields sub-optimal task performance: Li and Liang (2021) reparameterize $\boldsymbol{v}_i$ with another trainable matrix and then feed it forward through an MLP. Here, we adopt Liu et al. (2021)'s approach. They feed $[\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4]$ through an LSTM (Hochreiter and Schmidhuber, 1997) and use the outputs. PLM parameters, LSTM parameters, and $\boldsymbol{v}_i$ are jointly trained. Our SP and DP have the same training objective and verbalizer.

Mixed prompting (MP). We also experiment with a simple combination of DP and SP, by asking the PLM to fill in the blank (_) in sentence:

Premise . `Question:` Hypothesis `?` <v_1><v_2><v_3><v_4> `Answer:` _ .

MP includes human descriptions of NLI as in DP and learns "soft prompts" as in SP.

3.3 Non-English prompting

We also explore the results of prompting the PLM with languages other than English, of which the *few-shot* annotation cost is affordable (Garrette and Baldridge, 2013; Lauscher et al., 2020; Zhao et al., 2021).

For a non-English language $\mathcal{L}$ like Turkish, we translate[1] the English prompting words (§3.2)

[1]We use Google Translate due to the simplicity of our

		Prompt	Verbalizer
EN	DP	Premise . Question: Hypothesis ? Answer: _ .	Entailment → yes
	SP	Premise . Hypothesis ? <v_1>...<v_4> _ .	Contradict → no
	MP	Premise . Question: Hypothesis ? <v_1>...<v_4> Answer: _ .	Neutral → maybe
TR	DP	Premise . Soru: Hypothesis ? Cevap: _ .	Entailment → Evet
	SP	Premise . Hypothesis ? <v_1>...<v_4> _ .	Contradict → hiçbir
	MP	Premise . Soru: Hypothesis ? <v_1>...<v_4> Cevap: _ .	Neutral → belki

Table 1: Prompts and verbalizers in English (EN) and Turkish (TR). "<v_1>...<v_4>"="<v_1><v_2><v_3><v_4>". Appendix §B shows translated prompts/verbalizers of the languages used in our experiments.

"Question" and "Answer" into $\mathcal{L}$, e.g., "Soru" and "Cevap" in Turkish. Correspondingly, the verbalizer maps gold labels into $\mathcal{L}$: "entailment"→ "Evet"; "contradiction"→ "hiçbir"; "neutral"→ "belki". Table 1 presents example prompts and verbalizers.

4 Dataset and Setup

Dataset. We conduct our experiments on natural language inference datasets MNLI and XNLI (Williams et al., 2018; Conneau et al., 2018). MNLI provides multi-genre English NLI sentence pairs. XNLI provides development and test splits of *human-translated parallel* NLI sentence pairs in 15 languages[2] and the *machine-translated* MNLI training sets in 14 languages.

For constructing the *few-shot training set*, we randomly sample without replacement K $\in$ {1, 2, 4, 8, 16, 32, 64, 128, 256} shots *per class* from the EN MNLI training split. Then we retrieve translations of this EN training set from XNLI to create the few-shot training sets in the other languages.

To simulate a realistic low-resource regime (Kann et al., 2019; Perez et al., 2021), we use *few-shot development sets*. For EN, we sample the same number of shots (as training) from the XNLI development split. As a result, a 2-shot experiment uses 2 training and 2 development shots per class. For other languages, we retrieve the translations of the English development set from XNLI. Following Conneau et al. (2018), we report *accuracy* on XNLI test.

Setup. We conduct all experiments using the pretrained XLM-RoBERTa-base model (Conneau et al., 2020) containing 270M parameters trained on 2.5 TB CommonCrawl data in 100 languages. We use PyTorch (Paszke et al., 2019) and the HuggingFace framework (Wolf et al., 2020).[3]

We use batch size 32 for finetuning and 24 for prompting methods due to resource limitations. Following Le Scao and Rush (2021), we use learning rate 1e-5 for both finetuning and prompting. Following the suggestions of Mosbach et al. (2021), Zhang et al. (2021), we train the model with a large number of epochs (50) and select the checkpoint that performs best on the development set. We repeat each experiment 5 times with different random seeds ({1, 2, 3, 4, 5}) and report mean and variance. Appendix §A shows our reproducibility checklist.

5 Experiments

5.1 Zero-shot crosslingual transfer

We first compare prompting with finetuning in *zero-shot crosslingual transfer* (Pires et al., 2019; Conneau et al., 2020; Artetxe and Schwenk, 2019; Hu et al., 2020): The PLM is trained on the EN few-shot dataset and then directly evaluated on the test set of all languages. Table 2 reports the results.

EN results. From column *EN* we observe that: (i) As expected, all four methods benefit from more shots. (ii) Prompting methods (DP/SP/MP) clearly outperform finetuning especially in low-resource regimes. For example, in the 4-shot experiment, SP outperforms finetuning by ≈8 (41.84-33.90) accuracy points. Table 3 displays some examples for which SP outperforms finetuning. The improvements become less significant when more shots are available, e.g., 256. (iii) SP outperforms DP for most choices of shots (except 128), evidencing the strength of relaxing the "discrete token" constraint in DP (Liu et al., 2021; Qin and Eisner, 2021; Zhong et al., 2021). But we give up the interpretability of DP for this better performance. (iv) Performance of MP – the combination of DP and SP – is decent, but not stellar. Future work may explore advanced prompting methods succeeding

prompt. Specialized bilingual dictionaries can also be used.

[2]The languages are English (EN), French (FR), Spanish (ES), German (DE), Greek (EL), Bulgarian (BG), Russian (RU), Turkish (TR), Arabic (AR), Vietnamese (VI), Thai (TH), Chinese (ZH), Hindi (HI), Swahili (SW), and Urdu (UR).

[3]Resources are available at https://github.com/mprompting/xlmrprompt

Shots	Method	AR	BG	DE	EL	*EN*	ES	FR	HI	RU	SW	TH	TR	UR	VI	ZH	$\overline{X}$
-	MAJ	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33
1	FT	32.53	32.63	32.94	32.53	32.91	32.61	32.65	32.87	32.67	32.77	33.11	32.68	32.87	32.69	32.77	32.75
	DP	32.08	33.23	32.97	33.24	33.15	33.78	34.08	33.41	33.78	33.45	33.00	34.01	31.99	32.83	33.64	33.24
	SP	**34.84**	**36.50**	**36.87**	**37.49**	**36.65**	**38.29**	**38.57**	**36.43**	**37.56**	**34.52**	**35.71**	**34.76**	**35.54**	**35.06**	**37.61**	**36.43**
	MP	32.31	32.32	33.03	32.14	33.29	34.02	33.74	34.12	33.03	32.86	32.18	34.59	32.65	32.82	33.35	33.10
2	FT	33.16	33.35	33.82	33.24	33.43	33.31	33.30	33.24	33.29	33.19	33.40	33.04	33.20	33.03	33.29	33.29
	DP	32.90	35.11	34.44	34.69	35.41	35.43	34.77	34.11	34.93	32.97	35.43	**35.19**	32.75	33.28	36.46	34.52
	SP	**35.91**	**38.08**	**38.15**	**38.42**	**37.97**	**38.23**	**38.62**	**36.32**	**39.22**	**34.35**	**37.20**	34.75	**35.52**	**36.67**	**37.71**	**37.14**
	MP	32.76	34.25	34.10	33.26	34.59	33.81	34.33	33.75	34.01	33.88	34.55	34.51	32.59	33.83	35.39	33.97
4	FT	33.86	33.89	33.73	33.63	33.90	33.58	33.55	33.86	33.58	33.75	33.71	33.79	33.67	33.85	33.78	33.74
	DP	35.42	37.64	38.85	37.67	39.50	38.91	38.26	36.43	37.54	34.72	37.76	**37.23**	35.92	36.02	38.74	37.37
	SP	**38.04**	**40.46**	**40.08**	**40.79**	**41.84**	**39.78**	**41.10**	**37.55**	**41.72**	**35.81**	**39.23**	35.88	**37.66**	**37.86**	**39.48**	**39.15**
	MP	33.14	33.79	35.16	33.95	36.26	35.52	35.44	34.63	34.21	33.53	35.96	35.62	33.51	34.06	37.10	34.79
8	FT	32.85	32.75	33.05	32.59	33.06	32.58	32.80	32.89	32.88	32.75	33.14	32.69	33.05	32.83	32.65	32.84
	DP	32.73	34.78	34.79	34.82	36.39	34.97	35.17	33.00	34.59	32.91	35.14	34.13	33.14	33.66	35.56	34.39
	SP	**36.30**	**38.84**	**38.22**	**38.68**	**39.02**	**38.16**	**38.82**	**35.86**	**39.73**	**34.50**	**37.90**	35.11	**35.61**	**37.41**	**37.17**	**37.42**
	MP	32.67	33.24	34.81	33.18	34.78	34.66	34.77	34.76	33.81	33.07	34.46	**35.12**	32.69	33.57	36.34	34.13
16	FT	33.72	34.09	34.28	33.49	34.73	33.82	33.81	33.08	34.06	33.69	33.06	33.57	33.22	34.01	33.46	33.74
	DP	35.07	37.07	37.51	37.43	38.24	36.91	36.61	35.85	36.51	33.84	37.21	35.74	34.86	35.77	**37.86**	36.43
	SP	**38.88**	**40.60**	**40.21**	**40.44**	**39.45**	**39.37**	**40.90**	**36.86**	**40.61**	**37.11**	**39.45**	**36.26**	**35.88**	**38.46**	37.35	**38.79**
	MP	32.46	33.02	33.98	32.59	33.20	34.54	34.39	34.30	33.90	33.28	33.47	34.69	32.67	33.28	35.68	33.70
32	FT	35.84	36.28	36.00	36.11	36.64	36.02	36.47	35.41	35.68	35.33	35.71	35.90	34.81	36.10	36.20	35.90
	DP	**41.80**	**43.51**	**43.49**	**42.50**	43.65	**42.83**	43.90	39.30	42.39	37.51	**40.51**	**42.01**	39.77	41.91	39.94	**41.67**
	SP	40.30	43.38	42.08	42.27	44.72	42.32	42.34	38.91	**43.76**	**37.54**	39.97	38.79	38.83	**42.09**	39.56	41.12
	MP	40.95	42.16	42.61	42.31	**45.52**	41.22	**44.67**	**40.17**	42.18	36.52	40.16	41.21	**40.48**	41.74	**40.89**	41.52
64	FT	40.16	39.56	40.10	39.87	41.68	40.34	39.47	39.53	38.34	39.64	39.18	39.50	39.23	40.85	39.63	39.81
	DP	**45.64**	**47.64**	48.05	**46.94**	48.89	44.95	47.97	**41.61**	**44.85**	**40.98**	**45.65**	**45.67**	43.37	47.30	**45.24**	**45.65**
	SP	43.48	43.81	45.99	43.70	49.04	45.79	46.11	40.86	44.51	40.49	44.68	41.91	40.09	45.25	44.17	43.99
	MP	43.86	46.01	**48.22**	46.79	**51.84**	**46.61**	**48.31**	40.11	44.75	37.84	45.01	44.82	**43.95**	**48.28**	43.03	45.30
128	FT	43.50	45.52	45.60	44.38	46.94	45.75	46.00	42.96	44.94	41.43	43.27	43.67	41.78	44.81	44.79	44.36
	DP	46.23	**50.49**	**50.99**	47.39	**53.68**	48.53	49.28	44.77	**46.93**	42.03	47.95	**49.56**	**44.21**	**48.92**	49.56	48.03
	SP	44.78	46.24	45.30	46.31	49.45	45.80	46.37	43.29	44.95	41.21	45.64	41.93	41.18	44.99	45.73	44.88
	MP	**46.48**	47.98	49.04	**49.09**	52.55	**49.66**	**50.34**	**47.03**	46.40	**42.89**	**48.08**	48.45	44.04	48.15	**50.47**	**48.04**
256	FT	52.13	54.57	54.43	54.00	57.79	**55.89**	55.39	**50.65**	52.90	**50.00**	51.22	**52.31**	**48.57**	54.16	52.10	53.07
	DP	**53.23**	55.59	**55.39**	**55.05**	60.14	50.64	54.43	46.10	51.35	45.26	53.42	50.83	48.42	**55.14**	52.72	52.51
	SP	52.26	**56.04**	53.02	53.12	**60.58**	54.80	**55.79**	49.43	52.49	47.33	**54.52**	52.08	48.48	54.54	**54.59**	**53.27**
	MP	52.77	53.98	50.71	54.63	60.13	51.64	55.32	49.58	**53.50**	45.27	53.37	51.28	47.16	52.34	53.80	52.37

Table 2: Zero-shot crosslingual transfer results in accuracy (%). Each number is the mean performance of 5 runs, when using finetuning (FT), discrete prompting (DP), soft prompting (SP), and mixed prompting (MP). "MAJ": majority baseline; $\overline{X}$: macro average across 15 languages. Please see Appendix Table 7 for variances.

Premise/Hypothesis	Prediction
This was the temper of the times. This wasn't the temper of the times.	"no" (Contradict)
We would go in there. We would enter there at 8pm.	"maybe" (Neutral)
I hope to hear from you soon. I hope we talk soon.	"yes" (Entailment)

Table 3: Qualitative examples for which prompting outperforms finetuning.

in both task performance and interpretability. We focus on DP and SP in following experiments.

Crosslingual transfer results closely follow the trends of EN results: Prompting outperforms finetuning when looking at the macro average $\overline{X}$. One intriguing finding is that DP successfully transfers the learned knowledge to target languages, better than SP in some languages, using the *code-switched* prompt: " Premise . `Question:` Hypothesis `? Answer: _ .` " where Premise and Hypothesis are non-English. Thus, DP is able to leverage the strong crosslingual ability of the multilingual PLM. Like finetuning, prompting does not uniformly benefit the 14 non-English languages. For example, the crosslingual transfer performance of HI/SW/UR is notably inferior compared with other languages.

Overall, prompting outperforms finetuning in zero-shot crosslingual transfer of NLI in the low-resource regimes.

5.2 In-language prompting

We next compare prompting with finetuning when using non-English few-shot datasets. Taking Turkish as an example, recall that we can use the Turkish prompts (§3.3) and few-shot datasets from XNLI (§4) to finetune/prompt the PLM directly.

Table 4 shows results of in-language experiments of Turkish, Urdu, Swahili, and Chinese. We make two main observations: (i) Prompting still outperforms finetuning, though the non-English prompts and verbalizers are translated from EN simply using Google Translate. (ii) In-language results are slightly worse but competitive to transfer learning results (Table 2). We conjecture that

Shots	Method	TR	UR	SW	ZH
8	FT	32.71	32.83	32.80	33.31
	DP	**38.02**	**39.33**	**33.84**	**37.46**
	SP	35.41	34.59	33.47	34.39
16	FT	33.00	33.78	33.46	33.56
	DP	39.39	**40.58**	**34.48**	**42.24**
	SP	**40.22**	35.47	33.99	35.64
32	FT	37.15	34.23	34.52	35.38
	DP	**48.79**	**41.67**	**37.52**	**38.04**
	SP	43.62	39.18	36.00	35.13
64	FT	38.87	35.90	36.37	42.14
	DP	**48.97**	**42.34**	37.72	**44.73**
	SP	47.26	39.12	**37.98**	40.75
128	FT	40.84	36.23	36.81	43.16
	DP	**49.73**	**45.22**	**41.26**	**49.24**
	SP	47.68	40.96	40.42	47.17
256	FT	49.41	40.12	42.17	48.98
	DP	**52.61**	**46.10**	**47.69**	**53.11**
	SP	51.21	44.60	46.89	52.76

Table 4: In-language results in accuracy (%). Prompting (DP/SP) outperforms finetuning (FT). Please see Appendix Table 6 for variances.

two factors result in the second observation. First, some languages have a small amount of pretraining data. For example, Swahili has 1.6GB pretraining data while English has 300GB (Conneau et al., 2020). Thus, the PLM may not be well pretrained for solving tasks in Swahili directly. Second, the few-shot training data for non-English languages is machine-translated (§4). With better *few-shot* translations and in-language expertise, prompting possibly could achieve even better results.

Overall, the experimental results show that directly prompting PLMs with non-English languages is also an effective way of solving NLU tasks in low-resource regimes.

6 Conclusion

We showed that prompting performs better than finetuning in few-shot crosslingual transfer and in-language training of multilingual natural language inference. We hope our results will encourage more research about prompting multilingual tasks and models.

Future work may explore using text-to-text models like T5 (Raffel et al., 2020) or other autoregressive PLMs (Lewis et al., 2020). Investigating PLMs containing even more parameters is also promising as shown by Lester et al. (2021).

Acknowledgements

We thank the anonymous reviewers for the insightful comments and suggestions. This work was funded by the European Research Council (ERC #740516).

References

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421, Virtual. PMLR.

Katharina Kann, Kyunghyun Cho, and Samuel R. Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. 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 3342–3349, Hong Kong, China. Association for Computational Linguistics.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,

pages 5203–5212, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3066–3079, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955*.

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*, volume 32. Curran Associates, Inc.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. Association for Computational Linguistics.

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.

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*, volume 32. Curran Associates, Inc.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239, Online. Association for Computational Linguistics.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. 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 1206–1215, New Orleans, Louisiana. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

Mengjie Zhao, Yi Zhu, Ehsan Shareghi, Ivan Vulić, Roi Reichart, Anna Korhonen, and Hinrich Schütze. 2021. A closer look at few-shot crosslingual transfer: The choice of shots matters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5751–5767, Online. Association for Computational Linguistics.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

A Reproducibility Checklist

A.1 Model architecture and number of parameters

We use the `xlm-roberta-base` model (Conneau et al., 2020). It contains 12 Transformer blocks with 768 hidden dimensions. Each block has 12 attention heads. Vocabulary size is 250K. Overall the model has 270M parameters and was pretrained on the on 2.5 TB of newly created clean CommonCrawl data in 100 languages.

Following Liu et al. (2021), we employ a bidirectional LSTM in SP and MP. The hidden dimension is also 768 so the number of LSTM parameters is $2 \times 4 \times (768 \times 768 + 768 \times 768 + 768)$ ≈10M. Another MLP is used to project the concatination of LSTM states back to 768-dimension which has $(768 \times 2 \times 768 + 768)$ ≈1.2M. SP and MP also have four learnable vectors $\boldsymbol{v}_i$ resulting in $768 \times 4 = 3072$ parameters.

A.2 Computing infrastructure

All experiments are conducted on GeForce GTX 1080Ti. For finetuning, we use batch size 32 and 4 GPUs. Because prompting uses the masked language model objective so we use a maximum batch size 24. A single GPU is used for 1-shot experiments. Two and three GPUs are used for 2- and 4-shot experiments. Other experiments use 6 GPUs.

A.3 Evaluation metrics

Our code is available at `https://github.com/mprompting/xlmrprompt`. We use the standard evaluation metric accuracy as Conneau et al. (2018). For finetuning, evaluation script path is `./finetuning/utils/eval_meters.py`. For DP, evaluation script path is `./pet/pet/trainers/meters.py`. For SP/MP, evaluation script path is `./sptuning/pet/trainers/meters.py`.

A.4 Hyperparameter search

We use the same learning rate (1e-5) as Le Scao and Rush (2021) who compare prompting and finetuning in English NLU tasks. No learning rate scheduling is used for clear comparisons. For both finetuning and prompting, the model is trained for 50 epochs and the checkpoint that performs best on development set is selected for performance evaluation.

		Prompt	Verbalizer
TR	DP	P . Soru: H ? Cevap: _ .	Entailment → Evet
	SP	P . H ? <v_1>...<v_4> _ .	Contradict → hiçbir
	MP	P . Soru: H ? <v_1>...<v_4> Cevap: _ .	Neutral → belki
SW	DP	P . Swali: H ? Jibu: _ .	Entailment → ndio
	SP	P . H ? <v_1>...<v_4> _ .	Contradict → hasi
	MP	P . Swali: H ? <v_1>...<v_4> Jibu: _ .	Neutral → labda
ZH	DP	P . 问题: H ? 答案: _ .	Entailment → 是
	SP	P . H ? <v_1>...<v_4> _ .	Contradict → 否
	MP	P . 问题: H ? <v_1>...<v_4> 答案: _ .	Neutral → 也许

Table 5: Prompts and verbalizers in Turkish (TR), Swahili (SW), and Chinese (ZH).

Shots	Method	TR	UR	SW	ZH
8	FT	32.71±0.61	32.83±0.29	32.80±0.56	33.31±0.27
	DP	38.02±1.14	39.33±0.58	33.84±0.44	37.46±0.62
	SP	35.41±0.30	34.59±0.26	33.47±0.34	34.39±0.25
16	FT	33.00±0.93	33.78±0.58	33.46±0.91	33.56±0.50
	DP	39.39±0.81	40.58±0.67	34.48±0.86	42.24±2.66
	SP	40.22±0.50	35.47±0.61	33.99±0.17	35.64±1.03
32	FT	37.15±1.78	34.23±0.85	34.52±1.20	35.38±0.47
	DP	48.79±0.40	41.67±1.39	37.52±1.08	38.04±1.00
	SP	43.62±0.67	39.18±1.09	36.00±1.23	35.13±0.75
64	FT	38.87±0.99	35.90±1.26	36.37±1.13	42.14±1.22
	DP	48.97±0.56	42.34±0.91	37.72±0.81	44.73±0.86
	SP	47.26±0.77	39.12±1.36	37.98±1.48	40.75±1.90
128	FT	40.84±1.45	36.23±0.19	36.81±1.85	43.16±0.95
	DP	49.73±0.73	45.22±0.57	41.26±1.48	49.24±0.89
	SP	47.68±0.68	40.96±1.23	40.42±1.53	47.17±0.54
256	FT	49.41±2.03	40.12±1.77	42.17±1.77	48.98±3.12
	DP	52.61±1.34	46.10±0.40	47.69±1.12	53.11±0.61
	SP	51.21±0.30	44.60±0.91	46.89±0.81	52.76±0.29

Table 6: In-language results in accuracy (%). Prompting (DP/SP) outperforms finetuning (FT). We report mean and variance of 5 runs.

A.5 Datasets and preprocessing

We retrieve the MNLI and XNLI datasets from the official websites: `cims.nyu.edu/~sbowman/multinli` and `cims.nyu.edu/~sbowman/xnli`. We use the tokenizer in the HuggingFace framework (Wolf et al., 2020) to preprocess the texts. In all experiments, the max sequence length is 256.

B Translated Prompts

Table 5 shows the prompts and verbalizers used in in-language experiments. We use Google Translate but more specialized bilingual dictionaries can also be used. For Urdu, we show the prompt and verbalizer in the code repository.

C More Results

Table 6 and Table 7 show performances with variances.

Shots	Method	AR	BG	DE	EL	*EN*	ES	FR	HI	RU	SW	TH	TR	UR	VI	ZH	$\overline{X}$
-	MAJ	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33
1	FT	32.53±1.08	32.63±1.02	32.94±0.50	32.53±1.10	32.91±0.62	32.61±0.89	32.65±0.95	32.87±0.64	32.67±0.94	32.77±0.67	33.11±0.26	32.68±0.88	32.87±0.59	32.69±0.88	32.77±0.86	32.75±0.16
	DP	32.08±0.21	33.23±0.15	32.97±0.25	33.24±0.10	33.15±0.16	33.78±0.08	34.08±0.05	33.41±0.07	33.78±0.11	33.45±0.36	33.00±0.30	34.01±0.07	31.99±0.27	32.83±0.27	33.64±0.06	33.24±0.60
	SP	34.84±1.67	36.50±1.41	36.87±0.48	37.49±0.37	36.65±1.52	38.29±0.40	38.57±0.34	36.43±0.61	37.56±1.30	34.52±1.61	35.71±1.78	34.76±0.34	35.54±0.28	35.06±1.65	37.61±0.40	36.43±1.27
	MP	32.31±0.20	32.32±0.15	33.03±0.15	32.14±0.27	33.29±0.60	34.02±0.11	33.74±0.34	34.12±0.22	33.03±0.42	32.86±0.25	32.18±0.17	34.59±0.27	32.65±0.16	32.82±0.21	33.35±0.06	33.10±0.73
2	FT	33.16±0.56	33.35±1.13	33.82±0.70	33.24±0.82	33.43±1.12	33.31±0.91	33.30±1.18	33.24±0.69	33.29±1.12	33.19±0.84	33.40±0.57	33.04±0.87	33.20±0.53	33.03±0.62	33.29±0.60	33.29±0.18
	DP	32.90±0.74	35.11±0.53	34.44±0.38	34.69±0.51	35.41±0.55	35.43±0.65	34.77±0.55	34.11±0.60	34.93±0.51	32.97±0.37	35.43±0.35	35.19±0.59	32.75±1.22	33.28±0.94	36.46±0.65	34.52±1.07
	SP	35.91±2.09	38.08±1.41	38.15±0.85	38.42±0.99	37.97±1.42	38.23±0.53	38.62±0.52	36.32±0.47	39.22±1.43	34.35±0.44	37.20±1.56	34.75±0.40	35.52±0.66	36.67±1.64	37.71±0.74	37.14±1.43
	MP	32.76±0.21	34.25±0.40	34.10±0.69	33.26±0.55	34.59±0.50	33.81±0.37	34.33±0.44	33.75±0.23	34.01±0.38	33.88±0.36	34.55±0.78	34.51±0.10	32.59±0.60	33.83±0.34	35.39±1.06	33.97±0.69
4	FT	33.86±1.15	33.89±1.34	33.73±1.12	33.63±0.67	33.90±1.22	33.58±0.94	33.55±1.05	33.86±1.48	33.58±1.27	33.75±1.11	33.71±1.28	33.79±1.43	33.67±0.98	33.85±1.53	33.78±1.54	33.74±0.12
	DP	35.42±0.46	37.64±0.39	38.85±0.50	37.67±0.53	39.50±0.37	38.91±0.44	38.26±0.23	36.43±0.38	37.54±0.30	34.72±0.24	37.76±0.56	37.23±0.33	35.92±0.53	36.02±0.56	38.74±0.40	37.37±1.36
	SP	38.04±1.81	40.46±1.67	40.08±1.23	40.79±1.42	41.84±2.10	39.78±1.35	41.10±2.03	37.55±1.05	41.72±1.99	35.81±1.49	39.23±1.27	35.88±1.30	37.66±1.49	37.86±1.56	39.48±1.55	39.15±1.88
	MP	33.14±0.49	33.79±1.26	35.16±1.93	33.95±1.17	36.26±2.38	35.52±0.95	35.44±1.23	34.63±0.73	34.21±1.10	33.53±0.74	35.96±1.36	35.62±1.19	33.51±0.73	34.06±0.89	37.10±0.77	34.79±1.13
8	FT	32.85±0.44	32.75±0.42	33.05±0.46	32.59±0.54	33.06±0.82	32.58±0.39	32.80±0.38	32.89±0.34	32.88±0.37	32.75±0.24	33.14±0.38	32.69±0.66	33.05±0.27	32.83±0.41	32.65±0.31	32.84±0.17
	DP	32.73±0.81	34.78±0.63	34.79±0.56	34.82±0.66	36.39±0.43	34.97±0.90	35.17±0.29	33.00±0.50	34.59±0.43	32.91±0.25	35.14±0.60	34.13±0.60	33.14±0.54	33.66±0.82	35.56±0.51	34.39±1.05
	SP	36.30±0.94	38.84±0.68	38.22±0.34	38.68±0.69	39.02±0.57	38.16±0.86	38.82±1.08	35.86±0.53	39.73±0.45	34.50±0.69	37.90±0.71	35.11±1.05	35.61±1.22	37.41±0.98	37.17±0.64	37.42±1.54
	MP	32.67±0.70	33.24±0.67	34.81±1.45	33.18±1.13	34.78±2.22	34.66±1.05	34.77±0.60	34.76±0.94	33.81±0.37	33.07±0.21	34.46±0.82	35.12±0.41	32.69±0.76	33.57±0.50	36.34±0.82	34.13±1.01
16	FT	33.72±0.84	34.09±0.74	34.28±0.66	33.49±1.15	34.73±0.97	33.82±0.88	33.81±1.13	33.08±0.38	34.06±1.23	33.69±0.88	33.06±0.78	33.57±0.93	33.22±0.45	34.01±0.63	33.46±0.91	33.74±0.44
	DP	35.07±1.55	37.07±2.07	37.51±1.91	37.43±2.39	38.24±2.15	36.91±1.67	36.61±1.26	35.85±1.67	36.51±1.89	33.84±0.78	37.21±1.45	35.74±1.12	34.86±1.75	35.77±1.94	37.86±1.44	36.43±1.18
	SP	38.88±0.92	40.60±0.43	40.21±0.66	40.44±0.87	39.45±0.71	39.37±0.55	40.90±0.43	36.86±0.59	40.61±0.53	37.11±0.61	39.45±0.43	36.26±1.04	35.88±0.47	38.46±0.70	37.35±0.95	38.79±1.65
	MP	32.46±0.19	33.02±0.43	33.98±1.01	32.59±0.34	33.20±0.62	34.54±0.59	34.39±0.62	34.30±0.19	33.90±0.51	33.28±0.56	33.47±1.57	34.69±0.40	32.67±0.42	33.28±0.34	35.68±1.87	33.70±0.88
32	FT	35.84±1.36	36.28±1.47	36.00±0.95	36.11±1.32	36.64±1.46	36.02±1.49	36.47±1.27	35.41±1.26	35.68±1.03	35.33±1.27	35.71±0.98	35.90±1.31	34.81±1.24	36.10±1.59	36.20±1.62	35.90±0.45
	DP	41.80±0.85	43.51±0.94	43.49±0.72	42.50±0.37	43.65±0.46	42.83±0.66	43.90±0.79	39.30±1.68	42.39±0.83	37.51±0.75	40.51±1.02	42.01±0.76	39.77±1.18	41.91±0.60	39.94±1.57	41.67±1.81
	SP	40.30±1.73	43.38±0.52	42.08±0.49	42.27±1.00	44.72±0.90	42.32±0.78	42.34±0.77	38.91±1.53	43.76±0.74	37.54±1.18	39.97±1.20	38.79±0.75	38.83±0.45	42.09±1.27	39.56±1.12	41.12±2.06
	MP	40.95±1.18	42.16±0.97	42.61±1.03	42.31±0.70	45.52±0.56	41.22±0.62	44.67±1.23	40.17±0.68	42.18±0.85	36.52±0.92	40.16±1.17	41.21±0.95	40.48±0.51	41.74±0.83	40.89±0.95	41.52±1.99
64	FT	40.16±2.05	39.56±1.83	40.10±2.42	39.87±1.67	41.68±2.26	40.34±2.60	39.47±2.01	39.53±2.11	38.34±1.68	39.64±1.60	39.18±1.81	39.50±1.91	39.23±1.87	40.85±2.39	39.63±2.12	39.81±0.75
	DP	45.64±0.59	47.64±0.81	48.05±0.66	46.94±0.51	48.89±0.95	44.95±0.85	47.97±0.68	41.61±0.77	44.85±0.61	40.98±1.14	45.65±0.84	45.67±1.15	43.37±0.49	47.30±0.72	45.24±1.17	45.65±2.23
	SP	43.48±0.71	43.81±0.92	45.99±0.56	43.70±0.22	49.04±0.46	45.79±0.91	46.11±0.86	40.86±0.81	44.51±1.26	40.49±0.35	44.68±0.34	41.91±1.50	40.09±0.64	45.25±0.72	44.17±0.86	43.99±2.33
	MP	43.86±1.05	46.01±1.13	48.22±1.20	46.79±1.39	51.84±1.12	46.61±1.39	48.31±0.86	40.11±1.23	44.75±1.24	37.84±0.53	45.01±1.02	44.82±1.22	43.95±1.25	48.28±1.16	43.03±1.50	45.30±3.33
128	FT	43.50±1.77	45.52±2.29	45.60±2.40	44.38±2.56	46.94±2.98	45.75±2.75	46.00±1.91	42.96±2.64	44.94±2.28	41.43±1.88	43.27±2.02	43.67±2.56	41.78±2.46	44.81±2.44	44.79±1.67	44.36±1.52
	DP	46.23±1.06	50.49±0.84	50.99±0.75	47.39±0.80	53.68±0.48	48.53±1.00	49.28±0.56	44.77±0.95	46.93±0.54	42.03±0.68	47.95±0.66	49.56±0.71	44.21±0.84	48.92±1.36	49.56±0.77	48.03±2.83
	SP	44.78±4.43	46.24±3.94	45.30±4.21	46.31±3.70	49.45±5.63	45.80±4.61	46.37±4.48	43.29±4.19	44.95±3.74	41.21±3.54	45.64±3.95	41.93±4.95	41.18±4.09	44.99±4.54	45.73±4.93	44.88±2.13
	MP	46.48±2.40	47.98±1.98	49.04±1.99	49.09±1.32	52.55±0.86	49.66±1.52	50.34±1.28	47.03±1.40	46.40±2.25	42.89±1.95	48.08±1.65	48.45±1.25	44.04±1.99	48.15±2.19	50.47±1.55	48.04±2.38
256	FT	52.13±2.07	54.57±1.95	54.43±2.26	54.00±1.63	57.79±2.36	55.89±2.37	55.39±1.97	50.65±1.71	52.90±2.03	50.00±1.89	51.22±1.92	52.31±2.06	48.57±2.25	54.16±1.71	52.10±2.60	53.07±2.35
	DP	53.23±0.56	55.59±1.40	55.39±1.43	55.05±0.86	60.14±0.67	50.64±0.83	54.43±1.29	46.10±1.77	51.35±0.86	45.26±1.57	53.42±0.86	50.83±0.71	48.42±1.10	55.14±0.72	52.72±0.68	52.51±3.76
	SP	52.26±1.89	56.04±1.71	53.02±2.14	53.12±1.62	60.58±0.86	54.80±1.42	55.79±1.15	49.43±1.87	52.49±1.78	47.33±1.11	54.52±1.21	52.08±1.49	48.48±0.93	54.54±1.05	54.59±1.31	53.27±3.17
	MP	52.77±0.31	53.98±1.26	50.71±2.41	54.63±0.95	60.13±0.40	51.64±2.83	55.32±0.79	49.58±1.87	53.50±0.68	45.27±0.70	53.37±0.23	51.28±1.26	47.16±0.75	52.34±1.15	53.80±0.82	52.37±3.38

Table 7: Zero-shot crosslingual transfer results in accuracy (%). We report mean and variance of 5 runs, when using finetuning (FT), discrete prompting (DP), soft prompting (SP), and mixed prompting (MP). "MAJ": majority baseline; $\overline{X}$: macro average across 15 languages.

Chapter 6

A Closer Look at Few-Shot Crosslingual Transfer: The Choice of Shots Matters

A Closer Look at Few-Shot Crosslingual Transfer: The Choice of Shots Matters

Mengjie Zhao[1*] **Yi Zhu**[2*] **Ehsan Shareghi**[3,2] **Ivan Vulić**[2]
Roi Reichart[4] **Anna Korhonen**[2] **Hinrich Schütze**[1]
[1]CIS, LMU Munich [2]LTL, University of Cambridge
[3]Department of Data Science & AI, Monash University
[4]Faculty of Industrial Engineering and Management, Technion, IIT
mzhao@cis.lmu.de, {yz568,iv250,alk23}@cam.ac.uk,
ehsan.shareghi@monash.edu, roiri@technion.ac.il

Abstract

Few-shot crosslingual transfer has been shown to outperform its zero-shot counterpart with pretrained encoders like multilingual BERT. Despite its growing popularity, little to no attention has been paid to standardizing and analyzing the design of few-shot experiments. In this work, we highlight a fundamental risk posed by this shortcoming, illustrating that the model exhibits a high degree of sensitivity to the selection of few shots. We conduct a large-scale experimental study on 40 sets of sampled few shots for six diverse NLP tasks across up to 40 languages. We provide an analysis of success and failure cases of few-shot transfer, which highlights the role of lexical features. Additionally, we show that a straightforward full model finetuning approach is quite effective for few-shot transfer, outperforming several state-of-the-art few-shot approaches. As a step towards standardizing few-shot crosslingual experimental designs, we make our sampled few shots publicly available.[1]

1 Introduction

Multilingual pretrained encoders like multilingual BERT (mBERT; Devlin et al. (2019)) and XLM-R (Conneau et al., 2020) are the top performers in crosslingual tasks such as natural language inference (Conneau et al., 2018), document classification (Schwenk and Li, 2018; Artetxe and Schwenk, 2019), and argument mining (Toledo-Ronen et al., 2020). They enable transfer learning through language-agnostic representations in crosslingual setups (Hu et al., 2020).

A widely explored transfer scenario is *zero-shot crosslingual transfer* (Pires et al., 2019; Conneau and Lample, 2019; Artetxe and Schwenk, 2019), where a pretrained encoder is finetuned on abundant task data in the source language (e.g., English) and then directly evaluated on target-language test data, achieving surprisingly good performance (Wu and Dredze, 2019; Hu et al., 2020). However, there is evidence that zero-shot performance reported in the literature has large variance and is often not reproducible (Keung et al., 2020a; Rios et al., 2020); the results in languages distant from English fall far short of those similar to English (Hu et al., 2020; Liang et al., 2020).

Lauscher et al. (2020) stress the importance of *few-shot crosslingual transfer* instead, where the encoder is first finetuned on a source language and then further finetuned with a small amount (10–100) of examples (**few shots**) of the target language. The few shots substantially improve model performance of the target language with negligible annotation costs (Garrette and Baldridge, 2013; Hedderich et al., 2020).

In this work, however, we demonstrate that the gains from few-shot transfer exhibit a *high degree of sensitivity to the selection of few shots*. For example, different choices for the few shots can yield a performance variance of over 10% accuracy in a standard document classification task. Motivated by this, we propose to fix the few shots for fair comparisons between different crosslingual transfer methods, and provide a benchmark resembling the standard "N-way K-shot" few-shot learning configuration (Fei-Fei et al., 2006; Koch et al., 2015). We also evaluate and compare several state-of-the-art (SotA) few-shot finetuning techniques, in order to understand their performance and susceptibility to the variance related to few shots.

We also demonstrate that the effectiveness of few-shot crosslingual transfer depends on the type of downstream task. For syntactic tasks such as named-entity recognition, the few shots can improve results by up to ≈20 F_1 points. For chal-

* Equal contribution.

[1]Code and resources are available at https://github.com/fsxlt

lenging tasks like adversarial paraphrase identification, the few shots do not help and even sometimes lead to worse performance than zero-shot transfer. To understand these phenomena, we conduct additional in-depth analyses, and find that the models tend to utilize shallow lexical hints (Geirhos et al., 2020) in the target language, rather than leveraging abstract crosslingual semantic features learned from the source language.

Our **contributions:** **1)** We show that few-shot crosslingual transfer is prone to large variations in task performance; this property hinders unbiased assessments of the effectiveness of different few-shot methods. **2)** To remedy this issue, we publish fixed and standardized few shots to support fair comparisons and reproducibility. **3)** We empirically verify that few-shot crosslingual transfer has different performance impact on structurally different tasks; we provide in-depth analyses concerning the source of performance gains. **4)** We analyze several SotA few-shot learning methods, and show that they underperform simple full model finetuning. We hope that our work will shed new light on the potential and current difficulties of few-shot learning in crosslingual setups.

2 Background and Related Work

Zero-/Few-Shot Crosslingual Transfer. Multilingual pretrained encoders show strong zero-shot crosslingual transfer (**ZS-XLT**) ability in various NLP tasks (Pires et al., 2019; Hsu et al., 2019; Artetxe and Schwenk, 2019). In order to guide and measure the progress, standardized benchmarks like XTREME (Hu et al., 2020) and XGLUE (Liang et al., 2020) have been developed.

Recently, Lauscher et al. (2020) and Hedderich et al. (2020) extended the focus on few-shot crosslingual transfer (**FS-XLT**): They assume the availability of a handful of labeled examples in a target language,[2] which are used to further finetune a source-trained model. The extra few shots bring large performance gains at low annotation cost. In this work, we systematically analyze this recent FS-XLT scenario.

FS-XLT resembles the intermediate-task transfer (STILT) approach (Phang et al., 2018; Pruksachatkun et al., 2020). In STILT, a pretrained encoder is finetuned on a resource-rich intermediate task, and then finetuned on a (resource-lean) target task. Likewise, FS-XLT focuses on transferring knowledge and general linguistic intelligence (Yogatama et al., 2019), although such transfer is between *languages* in the same task instead of between different tasks.

Few-shot learning was first explored in computer vision (Miller et al., 2000; Fei-Fei et al., 2006; Koch et al., 2015); the aim there is to learn new concepts with only few images. Methods like prototypical networks (Snell et al., 2017) and model-agnostic meta-learning (MAML; Finn et al. (2017)) have also been applied to many monolingual (typically English) NLP tasks such as relation classification (Han et al., 2018; Gao et al., 2019), named-entity recognition (Hou et al., 2020a), word sense disambiguation (Holla et al., 2020), and text classification (Yu et al., 2018; Yin, 2020; Yin et al., 2020; Bansal et al., 2020; Gupta et al., 2020). However, recent few-shot learning methods in computer vision consisting of two simple finetuning stages, first on base-class images and then on new-class few shots, have been shown to outperform MAML and achieve SotA scores (Wang et al., 2020; Chen et al., 2020; Tian et al., 2020; Dhillon et al., 2020). Inspired by this work, we compare various few-shot finetuning methods from computer vision in the context of FS-XLT.

Task Performance Variance. Deep neural networks' performance on NLP tasks is bound to exhibit large variance. Reimers and Gurevych (2017) and Dror et al. (2019) stress the importance of reporting *score distributions* instead of a single score for fair(er) comparisons. Dodge et al. (2020), Mosbach et al. (2021), and Zhang et al. (2021) show that finetuning pretrained encoders with different random seeds yields performance with large variance. In this work, we examine a specific source of variance: We show that the choice of the few shots in crosslingual transfer learning also introduces large variance in performance; consequently, we offer standardized few shots for more controlled and fair comparisons.

3 Method

Following Lauscher et al. (2020) and Hedderich et al. (2020), our FS-XLT method comprises two stages. First, we conduct **source-training**: The pretrained mBERT is finetuned with abundant annotated data in the source language. Similar to Hu et al. (2020), Liang et al. (2020) and due to

[2] According to Garrette and Baldridge (2013), it is possible to collect ≈100 POS-annotated sentences in two hours even for low-resource languages such as Malagasy.

Name	Metric	Task	$\|\mathcal{T}\|$	TS	# of lang.
XNLI	Acc.	Natural language inference	3	No	15
PAWSX	Acc.	Paraphrase identification	2	No	7
MLDoc	Acc.	News article classification	4	Yes	8
MARC	Acc.	Amazon reviews	5	Yes	6
POS	F1	Part-of-speech tagging	17	Yes	29
NER	F1	Named-entity recognition	7	Yes	40

Table 1: Evaluation datasets. $|\mathcal{T}|$: Number of classes (classification tasks) and label set size (POS and NER). TS: availability of a training split in the target language.

the abundant labeled data for many NLP tasks, we choose English as the source in our experiments. Directly evaluating the source-trained model after this stage corresponds to the widely studied ZS-XLT scenario. The second stage is **target-adapting**: The source-trained model from previous stage is adapted to a target language using few shots. We discuss details of sampling the few shots in §4. The development set of the target language is used for model selection in this stage.

4 Experimental Setup

We consider three types of tasks requiring varying degrees of semantic and syntactic knowledge transfer: Sequence classification (**CLS**), named-entity recognition (**NER**), and part-of-speech tagging (**POS**) in up to 40 typologically diverse languages (cf., Appendix §B).

4.1 Datasets and Selection of Few Shots

For the CLS tasks, we sample few shots from four multilingual datasets: News article classification (MLDoc; Schwenk and Li (2018)); Amazon review classification (MARC; Keung et al. (2020b)); natural language inference (XNLI; Conneau et al. (2018); Williams et al. (2018)); and crosslingual paraphrase adversaries from word scrambling (PAWSX; Zhang et al. (2019); Yang et al. (2019)). We use treebanks in Universal Dependencies (Nivre et al., 2020) for POS, and WikiANN dataset (Pan et al., 2017; Rahimi et al., 2019) for NER. Table 1 reports key information about the datasets.

We adopt the conventional few-shot sampling strategy (Fei-Fei et al., 2006; Koch et al., 2015; Snell et al., 2017), and conduct "N-way K-shot" sampling from the datasets; N is the number of classes and K refers to the number of shots per class. A group of N-way K-shot data is referred to as a **bucket**. We set N equal to the number of labels $|\mathcal{T}|$. Following Wang et al. (2020), we sample 40 buckets for each target (i.e., non-English) language of a task to get a reliable estimation of model performance.

CLS Tasks. For MLDoc and MARC, each language has a train/dev/test split. We sample the buckets without replacement from the training set of each target language, so that buckets are disjoint from each other. Target languages in XNLI and PAWSX only have dev/test splits. We sample the buckets from the dev set; the remaining data serves as a single new dev set for model selection during target-adapting. For all tasks, we use $K \in \{1, 2, 4, 8\}$.

POS and NER. For the two structured prediction tasks, "N-way K-shot" is not well-defined because each sentence contains one or more labeled tokens. We use a similar sampling principle as with CLS, where N is the size of the label set for each language and task, but K is set to the minimum number of occurrences for each label. In particular, we utilize the *Minimum-Including Algorithm* (Hou et al., 2020b,a) to satisfy the following criteria when sampling a bucket: 1) each label appears at least K times, and 2) at least one label will appear less than K times if any sentence is removed from the bucket. Appendix §C gives sampling details. In contrast to sampling for CLS, we do not enforce samples from different buckets to be disjoint due to the small amount of data in some low-resource languages. We only use $K \in \{1, 2, 4\}$ and exclude $K = 8$, as 8-shot buckets already have lots of labeled tokens, and thus (arguably) might not be considered few-shot.

4.2 Training Setup

We use the pretrained cased mBERT model (Devlin et al., 2019), and rely on the PyTorch-based (Paszke et al., 2019) HuggingFace Transformers repository (Wolf et al., 2019) in all experiments.

For *source-training*, we finetune the pretrained encoder for 10 epochs with batch size 32. For *target-adapting* to *every* target language, the few-shot data is a sampled bucket in this language, and we finetune on the bucket for 50 epochs with early-stopping of 10 epochs. The batch size is set to the number of shots in the bucket. Each target-adapting experiment is repeated 40 times using the 40 buckets. We use the Adam optimizer (Kingma and Ba, 2015) with default parameters in both stages with learning rates searched over $\{1e-5, 3e-5, 5e-5, 7e-5\}$. For CLS tasks, we use mBERT's `[CLS]` token as the final represen-

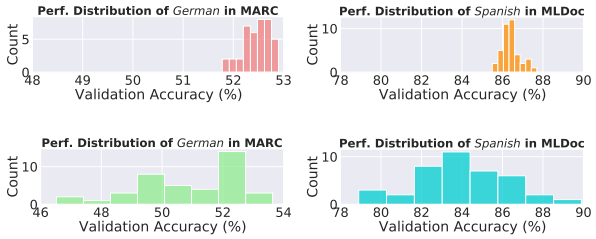


Figure 1: Histograms of dev set accuracies. Top: 40 runs with different random seeds. Bottom: 40 runs with different 1-shot buckets. Left: DE MARC. Right: ES MLDoc. The variance due to buckets is larger.

tation. For NER and POS, following Devlin et al. (2019), we use a linear classifier layer on top of the representation of each tokenized word, which is its last wordpiece (He and Choi, 2020).

We set the maximum sequence length to 128 after wordpiece tokenization (Wu et al., 2016), in all experiments. Further implementation details are shown in our Reproducibility Checklist in Appendix §A.

5 Results and Discussion

5.1 Source-Training Results

The ZS-XLT performance from English (EN) to target languages of the four CLS tasks are shown in the $K = 0$ column in Table 2. For NER and POS, the results are shown in Figure 2.

For XTREME tasks (XNLI, PAWSX, NER, POS), our implementation delivers results comparable to Hu et al. (2020). For MLDoc, our results are comparable to (Dong and de Melo, 2019; Wu and Dredze, 2019; Eisenschlos et al., 2019). It is worth noting that reproducing the exact results is challenging, as suggested by Keung et al. (2020a). For MARC, our zero-shot results are worse than Keung et al. (2020b)'s who use the dev set of each target language for model selection while we use EN dev, following the common true ZS-XLT setup.

5.2 Target-Adapting Results

Variance of Few-Shot Transfer. We hypothesize that FS-XLT suffers from large variance (Dodge et al., 2020) due to the large model complexity and small amount of data in a bucket. To test this empirically, we first conduct two experiments on MLDoc and MARC. First, for a *fixed random seed*, we repeat 1-shot target-adapting 40 times using different 1-shot buckets in German (DE) and Spanish (ES). Second, for a fixed 1-shot bucket, we repeat the same experiment 40 times using random seeds in $\{0 \dots 39\}$. Figure 1 presents the dev set performance distribution of the 40 runs with 40 random seeds (top) and 40 1-shot buckets (bottom).

With exactly the same training data, using different random seeds yields a 1–2 accuracy difference of FS-XLT (Figure 1 top). A similar phenomenon has been observed in finetuning monolingual encoders (Dodge et al., 2020) and multilingual encoders with ZS-XLT (Keung et al., 2020a; Wu and Dredze, 2020b; Xia et al., 2020); we show this observation also holds for FS-XLT. The key takeaway is that varying the buckets is a more severe problem. It causes much larger variance (Figure 1 bottom): The maximum accuracy difference is ≈6 for DE MARC and ≈10 for ES MLDoc. This can be due to the fact that difficulty of individual examples varies in a dataset (Swayamdipta et al., 2020), resulting in different amounts of information encoded in buckets.

This large variance could be an issue when comparing different few-shot learning algorithms. The bucket choice is a strong confounding factor that may obscure the strength of a promising few-shot technique. Therefore, for fair comparison, *it is necessary to work with a fixed set of few shots.* We propose to fix the sampled buckets for unbiased comparison of different FS-XLT methods. We publish the sampled buckets from the six multilingual datasets as a fixed and standardized few-shot evaluation benchmark.

In what follows, each FS-XLT experiment is repeated 40 times using 40 different buckets with the same fixed random seed; we report mean and standard deviation. As noted, the variance due to random seeds is smaller (cf., Figure 1) and has been well studied before (Reimers and Gurevych, 2017; Dodge et al., 2020). In this work, we thus focus our attention and limited computing resources on understanding the impact of buckets, the *newly detected* source of variance. However, we encourage practitioners to report results with both factors considered in the future.

Different Numbers of Shots. A comparison concerning the number of shots (K), based on the few-shot results in Table 2 and Figure 2, reveals that the buckets largely improve model performance on a majority of tasks (MLDoc, MARC, POS, NER) over zero-shot results. This is in line with prior work (Lauscher et al., 2020; Hedderich et al., 2020) and follows the success of work on using bootstrapped data (Chaudhary et al., 2019; Sherborne

		K=0	K=1	K=2	K=4	K=8
MLDoc	EN	96.88	-	-	-	-
	DE	88.30	90.36 ± 1.48	90.77 ± 0.87	91.85 ± 0.83	91.98 ± 0.82
	FR	83.05	88.94 ± 2.46	89.71 ± 1.68	90.80 ± 0.88	91.01 ± 0.94
	ES	81.90	83.99 ± 2.35	85.65 ± 1.60	86.30 ± 1.85	88.46 ± 1.90
	IT	74.13	74.97 ± 2.04	75.29 ± 1.57	76.43 ± 1.41	78.12 ± 1.25
	RU	72.33	77.40 ± 4.27	80.57 ± 1.37	81.33 ± 1.33	81.91 ± 1.21
	ZH	84.38	87.18 ± 1.45	87.31 ± 1.53	88.33 ± 1.11	88.72 ± 1.05
	JA	74.58	76.23 ± 1.59	76.71 ± 2.12	78.60 ± 2.43	81.17 ± 1.72
MARC	EN	64.52	-	-	-	-
	DE	49.62	51.50 ± 1.58	52.76 ± 0.87	52.78 ± 1.00	53.32 ± 0.59
	FR	47.30	49.32 ± 1.34	49.70 ± 1.43	50.64 ± 0.94	51.23 ± 0.76
	ES	48.44	49.72 ± 1.24	49.96 ± 1.12	50.45 ± 1.22	51.25 ± 0.93
	ZH	40.40	43.19 ± 1.76	44.45 ± 1.36	45.40 ± 1.26	46.40 ± 0.93
	JA	38.84	41.95 ± 2.09	43.63 ± 1.30	43.98 ± 0.89	44.44 ± 0.69
XNLI	EN	82.67	-	-	-	-
	DE	70.32	70.58 ± 0.36	70.60 ± 0.34	70.61 ± 0.39	70.70 ± 0.50
	FR	73.57	73.41 ± 0.48	73.74 ± 0.46	73.57 ± 0.49	73.77 ± 0.44
	ES	73.71	73.84 ± 0.40	73.87 ± 0.44	73.74 ± 0.48	73.87 ± 0.46
	RU	68.70	68.81 ± 0.52	68.76 ± 0.54	68.87 ± 0.55	68.81 ± 0.77
	ZH	69.32	69.73 ± 0.94	69.75 ± 0.94	70.56 ± 0.76	70.62 ± 0.86
	AR	64.97	64.75 ± 0.36	64.82 ± 0.23	64.82 ± 0.23	64.94 ± 0.37
	BG	67.58	68.15 ± 0.69	68.19 ± 0.75	68.55 ± 0.67	68.32 ± 0.70
	EL	65.67	65.64 ± 0.40	65.73 ± 0.36	65.80 ± 0.41	66.00 ± 0.53
	HI	56.57	56.94 ± 0.82	57.07 ± 0.82	57.21 ± 1.14	57.82 ± 1.18
	SW	48.08	50.33 ± 1.08	50.28 ± 1.24	51.08 ± 0.62	51.01 ± 0.79
	TH	46.17	49.43 ± 2.60	50.08 ± 2.42	51.32 ± 2.07	52.16 ± 2.43
	TR	60.40	61.02 ± 0.68	61.20 ± 0.61	61.35 ± 0.49	61.31 ± 0.56
	UR	57.05	57.56 ± 0.85	57.83 ± 0.91	58.20 ± 0.93	58.67 ± 1.03
	VI	69.82	70.04 ± 0.59	70.14 ± 0.75	70.23 ± 0.63	70.41 ± 0.70
PAWSX	EN	93.90	-	-	-	-
	DE	83.80	84.14 ± 0.40	84.08 ± 0.42	84.04 ± 0.47	84.23 ± 0.66
	FR	86.90	87.07 ± 0.27	87.06 ± 0.37	87.03 ± 0.31	86.94 ± 0.41
	ES	88.25	87.90 ± 0.54	87.80 ± 0.56	87.84 ± 0.53	87.85 ± 0.75
	ZH	77.75	77.71 ± 0.37	77.63 ± 0.47	77.68 ± 0.51	77.82 ± 0.64
	JA	73.30	73.78 ± 0.75	73.71 ± 1.04	73.48 ± 0.69	73.79 ± 1.28
	KO	72.05	73.75 ± 1.30	73.11 ± 1.05	73.79 ± 0.92	73.31 ± 0.61

Table 2: Zero-shot (column $K = 0$) and few-shot (columns $K > 0$) results (Acc. in %) on the test set for CLS tasks. Green [red]: few-shot transfer outperforms [underperforms] zero-shot transfer.

et al., 2020).

In general, we observe that: **1)** 1-shot buckets bring the largest relative performance improvement over ZS-XLT; **2)** the gains follow the increase of K, but with diminishing returns; **3)** the performance variance across the 40 buckets decreases as K increases. These observations are more pronounced for POS and NER; e.g., 1-shot EN to Urdu (UR) POS transfer shows gains of ≈22 F_1 points (52.40 with zero-shot, 74.95 with 1-shot).

For individual runs, we observe that models in FS-XLT tend to overfit the buckets quickly at small K values. For example, in around 32% of NER 1-shot buckets, the model achieves the best dev score right after the first epoch; continuing the training only degrades performance. Similar observations hold for semantic tasks like MARC, where in 10 out of 40 DE 1-shot buckets, the dev set performance peaks at epoch 1 (cf. learning curve in Appendix §D Figure 6). This suggests the necessity of running the target-adapting experiments on multiple buckets if reliable conclusions are to be drawn.

Different Downstream Tasks. The models for different tasks present various levels of sensitivity to FS-XLT. Among the CLS tasks that require semantic reasoning, FS-XLT benefits MLDoc the most. This is not surprising given the fact that keyword matching can largely solve MLDoc (Artetxe et al., 2020a,b): A few examples related to target language keywords are expected to significantly improve performance. FS-XLT also yields prominent gains on the Amazon review classification dataset MARC. Similar to MLDoc, we hypothesize that just matching a few important opinion and sentiment words (Liu, 2012) in the target language brings large gains already. We provide further qualitative analyses in §5.4.

XNLI and PAWSX behave differently from MLDoc and MARC. XNLI requires higher level semantic reasoning on pairs of sentences. FS-XLT performance improves modestly (XNLI) or even decreases (PAWSX-ES) compared to ZS-XLT, even with large K. PAWSX requires a model to distinguish adversarially designed non-paraphrase sentence pairs with large lexical overlap like "Flights from New York to Florida" and "Flights from Florida to New York" (Zhang et al., 2019). This poses a challenge for FS-XLT, given the small amount of target language information in the buckets. Therefore, *when buckets are small (e.g., $K = 1$) and for challenging semantic tasks like PAWSX, the buckets do not substantially help.* Annotating more shots in the target language is an intuitive solution. Designing task-specific pretraining/finetuning objectives could also be promising (Klein and Nabi, 2020; Ram et al., 2021).

Unlike CLS tasks, POS and NER benefit from FS-XLT substantially. We speculate that there are two reasons: **1)** Both tasks often require little to no high-level semantic understanding or reasoning; **2)** due to i.i.d. sampling, train/dev/test splits are likely to have overlapping vocabulary, and the labels in the buckets can easily propagate to dev and test. We delve deeper into these conjectures in §5.4.

Different Languages. For languages that are more distant from EN, e.g., with different scripts, small lexical overlap, or fewer common typological features (Pires et al., 2019; Wu and Dredze, 2020a), FS-XLT introduces crucial lexical and structural information to guide the update of embedding and transformer layers in mBERT.

We present several findings based on the NER and POS results for a typologically diverse language sample. Figure 2 shows that for languages with non-Latin scripts (different from EN), despite

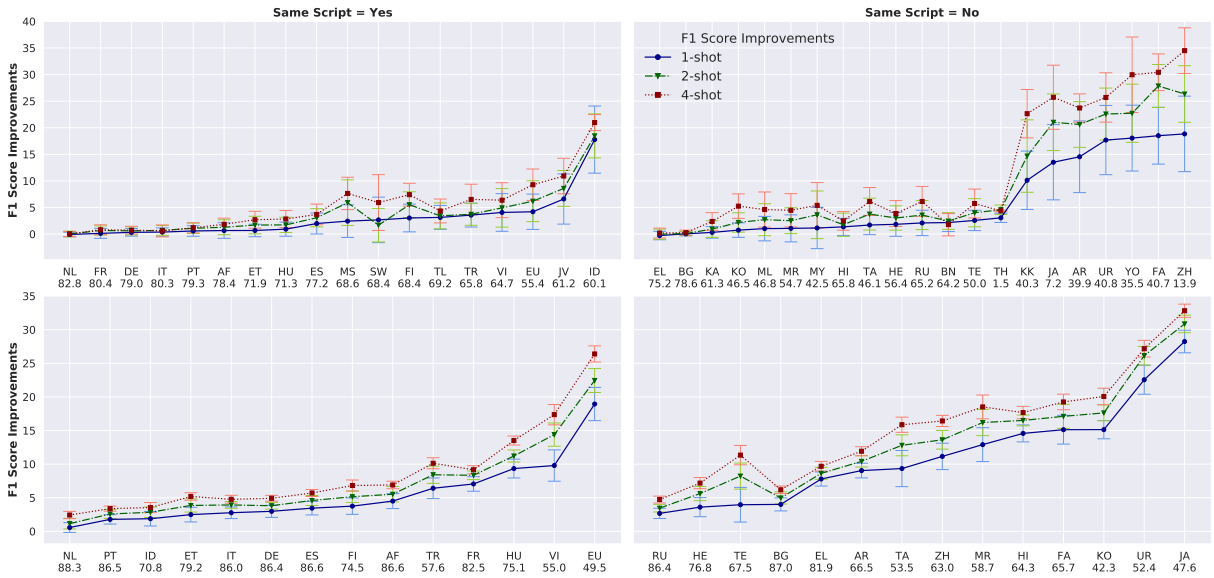


Figure 2: Improvement in F_1 (mean and standard deviation) of FS-XLT over ZS-XLT (numbers shown on x-axis beneath each language) for NER (top) and POS (bottom) for three different bucket sizes. See Appendix §D (Tables 12 and 13) for absolute numerical values.

Task	Factor	S	P
NER	lexical overlap	-0.34	-0.35
	# of common linguistic features	-0.37	-0.10
POS	lexical overlap	-0.63	-0.50
	# of common linguistic features	-0.57	-0.54

Table 3: Correlations between FS-XLT F_1 score gains and the two factors (lexical overlap and the number of common linguistic features with EN) when considered independently for POS and NER: S/R denotes Spearman's/Pearson's ρ. See Footnotes 3, 4 for information on the two factors.

their small to non-existent lexical overlap[3] and diverging typological features (see Appendix §D Tables 9 and 14), the performance boosts are generally larger than those in the same-script target languages: 6.2 vs. 3.0 average gain in NER and 11.4 vs. 5.4 in POS for $K = 1$. This clearly manifests the large information discrepancy between target-language buckets and source-language data. EN data is less relevant to these languages, so they obtain very limited gain from source-training, reflected by their low ZS-XLT scores. With a small amount of target-language knowledge in the buckets, the performance is improved dramatically, highlighting the effectiveness of FS-XLT.

Table 3 shows that, besides script form, lexical overlap and the number of linguistic features common with EN[4] also contribute directly to FS-XLT performance difference among languages: There is a moderate *negative* correlation between F_1 score gains vs. the two factors when considered independently for both syntactic tasks: The fewer overlaps/features a target language shares with EN, the larger the gain FS-XLT achieves.

This again stresses the importance of buckets – they contain target-language-specific knowledge about a task that cannot be obtained by ZS-XLT, which solely relies on language similarity. Interestingly, Pearson's ρ indicates that common linguistic features are much less linearly correlated with FS-XLT gains in NER than in POS.

5.3 Importance of Source-Training

Table 4 reports the performance *drop* when directly carrying out target-adapting, without any prior source-training of mBERT. We show the scores for MLDoc and PAWSX as a simple and a challenging CLS task, respectively. For NER and POS, we select two high- (Russian (RU), ES), mid- (Vietnamese (VI), Turkish (TR)), and low-resource languages (Tamil (TA), Marathi (MR)) each.[5]

The results clearly indicate that omitting the

[3]We define lexical overlap as $\frac{|V|_L \cap |V|_{EN}}{|V|_{EN}}$ where V denotes vocabulary. $|V|_L$ is computed with the 40 buckets of a target language L.

[4]Following Pires et al. (2019), we use six WALS features: 81A (Order of Subject, Object and Verb), 85A (Order of Adposition and Noun), 86A (Order of Genitive and Noun), 87A (Order of Adjective and Noun), 88A (Order of Demonstrative and Noun), and 89A (Order of Numeral and Noun).

[5]The categorization based on resource availability is according to WikiSize (Wu and Dredze, 2020a).

	MLDoc		PAWSX			POS		NER	
	K=1	K=8	K=1	K=8		K=1	K=4	K=1	K=4
DE	-37.73	-7.67	-31.11	-30.82	RU	-15.89	-3.20	-48.19	-35.77
FR	-38.14	-13.21	-33.02	-32.34	ES	-9.51	-0.93	-63.98	-41.53
ES	-33.69	-14.38	-33.76	-33.97	VI	-7.82	-0.36	-54.41	-41.45
IT	-33.63	-12.62	-	-	TR	-15.05	-8.08	-54.35	-34.52
RU	-30.66	-11.08	-	-	TA	-13.72	-4.40	-34.70	-24.81
ZH	-37.31	-12.57	-23.74	-23.65	MR	-11.34	-3.63	-40.10	-25.68
JA	-29.82	-14.32	-20.97	-20.82	-	-	-	-	-
KO	-	-	-19.83	-19.68	-	-	-	-	-

Table 4: Performance drop when conducting target-adapting without source-training.

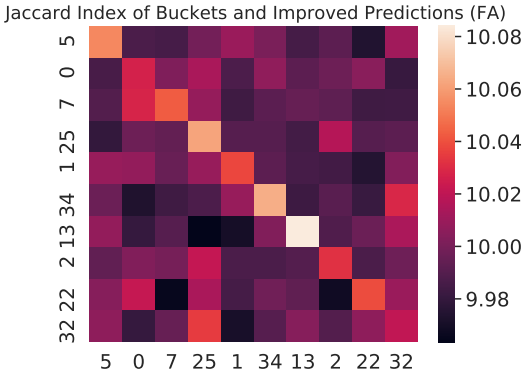


Figure 3: Normalized (with softmax) Jaccard index (%) of a bucket (row) and the improved predictions achieved with 10 buckets (column).

source-training stage yields large performance drops. Even larger variance is also observed in this scenario (cf. Appendix §D Table 11). Therefore, the model indeed learns, when trained on the source language, some transferable crosslingual features that are beneficial to target languages, both for semantic and syntactic tasks.

5.4 Importance of Lexical Features

We now investigate the sources of gains brought by FS-XLT over ZS-XLT.

For syntactic tasks, we take Persian (FA) POS as an example. Figure 3 visualizes the lexical overlap, measured by the Jaccard index, of 10 1-shot buckets (rows) and the improved word-label predictions introduced by target-adapting on each of the buckets (columns). In more detail, for column c, we collect the set (denoted as C_c) of all test set words whose label is incorrectly predicted by the zero-shot model, but correctly predicted by the model trained on the c-th bucket. For row i, we denote with B_i the set of words occurring in bucket i. The figure shows in cell (i, k) the Jaccard index of B_i and C_k. The bright color (i.e., higher lexical overlap) on the diagonal reflects that the improvements

Figure 4: Improvement of word-label predictions introduced by a bucket (x-axis) in FA (top), UR (mid), and HI (bottom), in relation to the words' presence in the bucket (True or False).

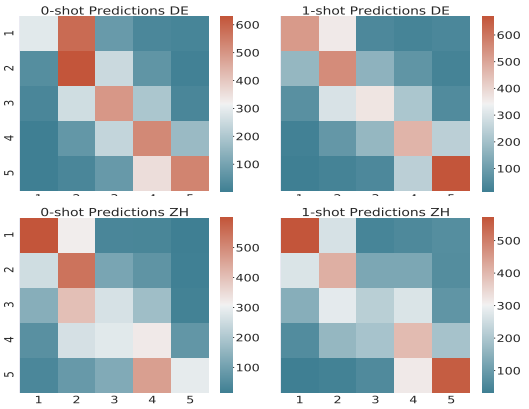


Figure 5: MARC (5 classes) test set prediction confusion matrices. Top: DE. Bottom: ZH. Left: zero-shot models. Right: 1-shot models. Colorbar numbers represent the number of instances in that cell.

introduced by a bucket are mainly[6] those word-label predictions that are lexically more similar to the bucket than to other buckets.

We also investigate the question: How many word-label predictions that are improved after FS-XLT occur in the bucket, i.e., in the training data? Figure 4 plots this for the 40 1-shot buckets in FA, UR, and Hindi (HI). We see that many test words do occur in the bucket (shown in orange), in line with recent findings (Lewis et al., 2021; Elangovan et al., 2021). These analyses shed light on why the buckets benefit NER/POS – which heavily rely on lexical information – more than higher level semantic tasks.

For the CLS task MARC, which requires un-

[6]Note that the sampled buckets for POS are not completely disjoint (cf. sampling strategy in §4).

token	[SEP]	.	nicht	!	Die	sehr
ΔAttn	+4.13	+2.91	+1.84	-1.75	-0.92	-0.81

Table 5: Tokens with the highest attention change from `[CLS]`, comparing zero-shot with a 1-shot DE bucket.

derstanding product reviews, Figure 5 visualizes the confusion matrices of test set predictions for DE and Chinese (ZH) zero- and 1-shot models; axis ticks are review scores in $\{1, 2, 3, 4, 5\}$. The squares on the diagonals in the two left heatmaps show that parameter initialization on EN is a good basis for well-performing ZS-XLT: This is particularly true for DE, which is linguistically closer to EN. Two extreme review scores – 1 (for DE) and 5 (for ZH) – have the largest confusions. The two right heatmaps show that improvements brought by the 1-shot buckets are mainly achieved by correctly predicting more cases of the two extreme review scores: $2 \rightarrow 1$ (DE) and $4 \rightarrow 5$ (ZH). But the more challenging cases (reviews with scores 2, 3, 4), which require non-trivial reasoning, are not significantly improved, or even become worse.

We inspect examples that are incorrectly predicted by the few-shot model (predicting 1), but are correctly predicted by the zero-shot model (predicting 2). Specifically, we compute the difference of where `[CLS]` attends to, before and after adapting the model on a 1-shot DE bucket. We extract and average attentions computed by the 12 heads from the topmost transformer layer.

Table 5 shows that "nicht" ("not") draws high attention change from `[CLS]`. "Nicht" (i.e., negation) by itself is not a reliable indicator of sentiment, so giving the lowest score to reviews solely because they contain "nicht" is not a good strategy. The following review is classified as 1 by the 1-shot model, but 2 is the gold label (as the review is not entirely negative):

"Die Uhr ging nicht einmal eine Minute ... **Optisch allerdings sehr schön**." ("The clock didn't even work one minute ... **Visually, however, very nice**.")

Pretrained multilingual encoders are shown to learn and store "language-agnostic" features (Pires et al., 2019; Zhao et al., 2020); §5.3 shows that source-training mBERT on EN substantially benefits other languages, even for difficult semantic tasks like PAWSX. Conditioning on such language-agnostic features, we expect that the buckets should lead to good understanding and reasoning capabilities for a target language. However, plain few-shot finetuning still relies heavily on unintended shallow lexical cues and shortcuts (Niven and Kao, 2019; Geirhos et al., 2020) that generalize poorly. Other open research questions for future work arise: How do we overcome this excessive reliance on lexical features? How can we leverage language-agnostic features with *few shots*? Our standardized buckets, baseline results, and analyses are the initial step towards researching and answering these questions.

5.5 Target-Adapting Methods

SotA few-shot learning methods (Chen et al., 2019; Wang et al., 2020; Tian et al., 2020; Dhillon et al., 2020) from computer vision consist of two stages: 1) training on base-class images, and 2) few-shot finetuning using new-class images. Source-training and target-adapting stages of FS-XLT, albeit among languages, follow an approach very similar to these methods. Therefore, we test their effectiveness for crosslingual transfer. These methods are built upon cosine similarity that imparts inductive bias about distance and is more effective than a fully-connected classifier layer (FC) with small K (Wang et al., 2020). Following (Chen et al., 2019; Wang et al., 2020; Tian et al., 2020), we freeze the embedding and transformer layers of mBERT, and explore four variants of the target-adapting stage using MARC.

COS+Pooler. We randomly initialize a trainable weight matrix $\mathbf{W} \in \mathbb{R}^{h \times c}$ where h is the hidden dimension size and c is the number of classes. Rewriting $\mathbf{W}$ as $[\mathbf{w}_1, \ldots, \mathbf{w}_i, \ldots, \mathbf{w}_c]$, we compute the logits of an input sentence representation $\mathbf{x} \in \mathbb{R}^h$ (from mBERT) belonging to class i as

$$\alpha \cdot \frac{\mathbf{x}^\intercal \mathbf{w}_i}{\|\mathbf{x}\|_2 \cdot \|\mathbf{w}_i\|_2} ,$$

where α is a scaling hyperparameter, set to 10 in all experiments. During training, $\mathbf{W}$ and mBERT's pooler layer containing a linear layer and a `tanh` non-linearity are updated.

FC+Pooler. During training, we update the linear classifier layer and mBERT's pooler layer.

FC only. During training, we only update the linear classifier layer. This variant largely reduces model complexity and exhibit lower variance when K is small.

FC(reset)+Pooler. Similar to FC+Pooler, but the source-trained linear classifier layer is randomly re-initialized before training.

Table 6 shows the performance of these methods along with full model finetuning (without freezing). *FC+Pooler* performs the best among the

		Full-Model Finetuning		FC only		FC + Pooler		COS + Pooler		FC (reset) + Pooler	
	K=0	K=1	K=8	K=1	K=8	K=1	K=8	K=1	K=8	K=1	K=8
DE	49.62	51.50 ± 1.58	53.32 ± 0.59	50.82 ± 1.17	52.58 ± 0.63	**51.18 ± 1.13**	**53.17 ± 0.58**	37.98 ± 5.53	45.85 ± 2.14	38.52 ± 6.64	49.46 ± 2.21
FR	47.30	49.32 ± 1.34	51.23 ± 0.76	48.19 ± 0.78	49.05 ± 0.93	**48.60 ± 1.02**	**49.97 ± 0.77**	39.93 ± 3.50	44.41 ± 1.95	40.12 ± 5.04	47.77 ± 2.00
ES	48.44	49.72 ± 1.24	51.25 ± 0.93	49.03 ± 0.73	49.69 ± 0.57	**49.28 ± 0.85**	**50.21 ± 0.63**	40.01 ± 4.33	45.35 ± 2.37	40.89 ± 4.96	47.73 ± 2.33
ZH	40.40	43.19 ± 1.76	46.40 ± 0.93	41.90 ± 1.15	43.34 ± 0.88	**42.30 ± 1.37**	**44.42 ± 0.65**	33.10 ± 5.48	38.31 ± 1.87	31.83 ± 7.00	42.07 ± 2.19
JA	38.84	41.95 ± 2.09	44.44 ± 0.69	40.76 ± 1.76	43.14 ± 0.76	**41.40 ± 1.74**	**43.81 ± 0.56**	34.36 ± 4.19	38.95 ± 1.80	32.80 ± 5.17	41.18 ± 1.68

Table 6: Accuracy (%) on MARC when varying classifier head configurations. Full-Model Finetuning updates all parameters during training; the other four methods only update a subset as described in §5.5. The best results (excluding Full-Model Finetuning) are in bold.

four for both $K = 1$ and $K = 8$ in all languages. However, it underperforms the full model finetuning, especially when $K = 8$. *FC only* is sub-optimal; yet the decrease in comparison to *FC+Pooler* is small, highlighting that EN-trained mBERT is a strong feature extractor. *COS+Pooler* and *FC(reset)+Pooler* perform considerably worse than the other two methods and zero-shot transfer – presumably because their new parameters need to be trained from scratch with few shots.

We leave further exploration of other possibilities of exploiting crosslingual features through collapse-preventing regularization (Aghajanyan et al., 2021) or contrastive learning (Gunel et al., 2021) to future work. Integrating prompting (Brown et al., 2020; Schick and Schütze, 2020; Gao et al., 2020; Liu et al., 2021) – a strong performing few-shot learning methodology for NLP – into the crosslingual transfer learning pipeline is also a promising direction.

6 Conclusion and Future Work

We have presented an extensive study of *few-shot crosslingual transfer*. The focus of the study has been on an empirically detected performance variance in few-shot scenarios: The models exhibit a high level of sensitivity to the choice of few shots. We analyzed and discussed the major causes of this variance across six diverse tasks for up to 40 languages. Our results show that large language models tend to overfit to few shots quickly and mostly rely on shallow lexical features present in the few shots, though they have been trained with abundant data in English. Moreover, we have empirically validated that state-of-the-art few-shot learning methods in computer vision do not outperform a conceptually simple alternative: Full model finetuning.

Our study calls for more rigor and accurate reporting of the results of few-shot crosslingual transfer experiments. They should include score distributions over standardized and fixed few shots. To aid this goal, we have created and provided such fixed few shots as a standardized benchmark for six multilingual datasets.

Few-shot learning is promising for crosslingual transfer, because it mirrors how people acquire new languages, and that the few-shot data annotation is feasible. In future work, we will investigate more sophisticated techniques and extend the work to more NLP tasks.

Acknowledgments

This work was funded by the European Research Council: ERC NonSequeToR (#740516) and ERC LEXICAL (#648909). We thank the anonymous reviewers and Fei Mi for their helpful suggestions.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020a. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, Dani Yogatama, Gorka Labaka, and Eneko Agirre. 2020b. A call for more rigor in unsupervised cross-lingual learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7375–7388, Online. Association for Computational Linguistics.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020. Learning to few-shot learn across diverse

natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Aditi Chaudhary, Jiateng Xie, Zaid Sheikh, Graham Neubig, and Jaime Carbonell. 2019. A little annotation does a lot of good: A study in bootstrapping low-resource named entity recognizers. 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 5164–5174, Hong Kong, China. Association for Computational Linguistics.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A closer look at few-shot classification. In *International Conference on Learning Representations*.

Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. 2020. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 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.

Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. 2020. A baseline for few-shot image classification. In *International Conference on Learning Representations*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping.

Xin Dong and Gerard de Melo. 2019. A robust self-learning framework for cross-lingual text classification. 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 6306–6310, Hong Kong, China. Association for Computational Linguistics.

Rotem Dror, Segev Shlomov, and Roi Reichart. 2019. Deep dominance - how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, Florence, Italy. Association for Computational Linguistics.

Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kadras, Sylvain Gugger, and Jeremy Howard. 2019. MultiFiT: Efficient multi-lingual language model fine-tuning. 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 5702–5707, Hong Kong, China. Association for Computational Linguistics.

Aparna Elangovan, Jiayuan He, and Karin Verspoor. 2021. Memorization vs. generalization : Quantifying data leakage in NLP performance evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1325–1335, Online. Association for Computational Linguistics.

L. Fei-Fei, R. Fergus, and P. Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Re-*

search, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. 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 6251–6256, Hong Kong, China. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.

Aakriti Gupta, Kapil Thadani, and Neil O'Hare. 2020. Effective few-shot classification with transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1061–1066, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Han He and Jinho D. Choi. 2020. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'20. Best Paper Candidate.

Michael A. Hedderich, David Adelani, Dawei Zhu, Jesujoba Alabi, Udia Markus, and Dietrich Klakow. 2020. Transfer learning and distant supervision for multilingual transformer models: A study on African languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2580–2591, Online. Association for Computational Linguistics.

Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2020. Learning to learn to disambiguate: Meta-learning for few-shot word sense disambiguation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4517–4533, Online. Association for Computational Linguistics.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020a. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1381–1393. Association for Computational Linguistics.

Yutai Hou, Jiafeng Mao, Yongkui Lai, Cheng Chen, Wanxiang Che, Zhigang Chen, and Ting Liu. 2020b. Fewjoint: A few-shot learning benchmark for joint language understanding. *CoRR*, abs/2009.08138.

Tsung-Yuan Hsu, Chi-Liang Liu, and Hung-yi Lee. 2019. Zero-shot reading comprehension by cross-lingual transfer learning with multi-lingual language representation model. 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 5933–5940, Hong Kong, China. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421, Virtual. PMLR.

Phillip Keung, Yichao Lu, Julian Salazar, and Vikas Bhardwaj. 2020a. Don't use English dev: On the zero-shot cross-lingual evaluation of contextual embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 549–554, Online. Association for Computational Linguistics.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020b. The multilingual Amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4563–4568, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7517–7523, Online. Association for Computational Linguistics.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML 2015 Deep Learning Workshop*.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online. Association for Computational Linguistics.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. XGLUE: A new benchmark datasetfor cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018, Online. Association for Computational Linguistics.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Erik G Miller, Nicholas E Matsakis, and Paul A Viola. 2000. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

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.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. *arXiv preprint arXiv:2101.00438*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Annette Rios, Mathias Müller, and Rico Sennrich. 2020. Subword segmentation and a single bridge language affect zero-shot neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 526–535, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Holger Schwenk and Xian Li. 2018. A corpus for multilingual document classification in eight languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Tom Sherborne, Yumo Xu, and Mirella Lapata. 2020. Bootstrapping a crosslingual semantic parser. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 499–517, Online. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4077–4087. Curran Associates, Inc.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. 2020. Rethinking few-shot image classification: A good embedding is all you need? In *Computer Vision – ECCV 2020*, pages 266–282, Cham. Springer International Publishing.

Orith Toledo-Ronen, Matan Orbach, Yonatan Bilu, Artem Spector, and Noam Slonim. 2020. Multilingual argument mining: Datasets and analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 303–317, Online. Association for Computational Linguistics.

Xin Wang, Thomas Huang, Joseph Gonzalez, Trevor Darrell, and Fisher Yu. 2020. Frustratingly simple few-shot object detection. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9919–9928, Virtual. PMLR.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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*, abs/1910.03771.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020a. Are all languages created equal in multilingual bert? In *Proceedings of the 5th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2020, Online, July 9, 2020*, pages 120–130. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020b. Do explicit alignments robustly improve multilingual encoders? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4471–4482, 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, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Patrick Xia, Shijie Wu, and Benjamin Van Durme. 2020. Which *BERT? A survey organizing contextualized encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7516–7533, Online. Association for Computational Linguistics.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Wenpeng Yin. 2020. Meta-learning for few-shot natural language processing: A survey.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239, Online. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. 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 1206–1215, New Orleans, Louisiana. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. 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 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Wei Zhao, Steffen Eger, Johannes Bjerva, and Isabelle Augenstein. 2020. Inducing language-agnostic multilingual representations. *arXiv preprint arXiv:2008.09112*.

A Reproducibility Checklist

A.1 mBERT Architecture and Number of Parameters

We use the "bert-base-multilingual-cased" model[7]. It contains 12 Transformer blocks with 768 hidden dimensions. Each block has 12 self attention heads. The model is pretrained on the concatenation of the Wikipedia dump of 104 languages.

There are about 179 million parameters in mBERT. For all the tasks, we use a linear output layer. Denoting the output dimension of a task as m, e.g., $m = 2$ for PAWSX. Then we have in total 179 million + $768 \times m + m$ parameters for the task.

A.2 Computing Infrastructure

All experiments are conducted on GeForce GTX 1080Ti. In the source-training stage, we use 4 GPUs with per-GPU batch size 32. In the target-adapting stage, we use a single GPU and the batch size is equal to the number of examples in a bucket.

A.3 Evaluation Metrics and Validation Performance

We follow the standard evaluation metrics used in XTREME (Hu et al., 2020) and they are shown in Table 1; evaluation functions in `scikit-learn` (Pedregosa et al., 2011) and `seqeval` (https://github.com/chakki-works/seqeval) are used. Link to code: `code/utils/eval_meters.py`.

The validation performance of the English-trained models are shown in the first row of Table 7; the optimal learning rate for each task is shown in the second row.

MLDoc	MARC	XNLI	PAWSX	POS	NER
98.1	65.1	83.5	94.5	95.6	84.3
1e-5	1e-5	3e-5	1e-5	1e-5	1e-5

Table 7: Source-training validation performance (%) and the optimal learning rate.

For all the FS-XLT experiments, we enclosed the validation scores in https://github.com/fsxlt/running-logs.

A.4 Hyperparameter Search

For both source-training and target-adapting, the only hyperparameter we search is learning rate (from $\{1e-5, 3e-5, 5e-5, 7e-5\}$) to reduce the sensitivity of our results to hyperparameter selection.

[7] https://github.com/google-research/bert/blob/master/multilingual.md

Algorithm 1: Minimum-including

Require: # of shot K, language data $\mathcal{D}$, label set $\mathcal{L}_\mathcal{D}$

Initialize a bucket $\mathcal{S} = \{\}$, $\text{Count}_{\ell_j} = 0$ $(\forall \ell_j \in \mathcal{L}_\mathcal{D})$

for ℓ *in* $\mathcal{L}_\mathcal{D}$ **do**
  while $\textit{Count}_\ell < K$ **do**
    From $\mathcal{D}$, randomly sample a $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ pair that $\boldsymbol{y}^{(i)}$ includes ℓ
    Add $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ to $\mathcal{S}$
    Update all Count_{ℓ_j} $(\forall \ell_j \in \mathcal{L}_\mathcal{D})$

for *each* $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ *in* $\mathcal{S}$ **do**
  Remove $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ from $\mathcal{S}$
  Update all Count_{ℓ_j} $(\forall \ell_j \in \mathcal{L}_\mathcal{D})$
  if *any* $\textit{Count}_{\ell_j} < K$ **then**
    Put $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ back to $\mathcal{S}$
    Update all Count_{ℓ_j} $(\forall \ell_j \in \mathcal{L}_\mathcal{D})$

Return $\mathcal{S}$

A.5 Datasets and Preprocessing

For tasks (XNLI, PAWSX, POS, NER) covered in XTREME (Hu et al., 2020), we utilize the provided preprocessed datasets. Our MLDoc dataset is obtained from https://github.com/facebookresearch/MLDoc. We retrieve MARC from docs.opendata.aws/amazon-reviews-ml/readme.html. Table 8 shows example entries of the datasets. It is worth noting that MARC is a single sentence review classification task, however, we put the "review title" and "product category" in the "Text B" field, following Keung et al. (2020b).

We utilize the tokenizer in the HuggingFace `Transformers` package (Wolf et al., 2019) to preprocess all the texts. In all experiments, we use 128 maximum sequence length and truncate from the end of a sentence if its length exceeds the limit.

B Languages

We work on 40 languages in total. They are shown in Table 9, together with their ISO 639-1 codes, writing script, and language features from WALs (https://wals.info/) used in our experiments.

C Minimum-Including Algorithm

We utilize the *Minimum-including Algorithm* from Hou et al. (2020a,b) for sampling the buckets of POS and NER which have several labels in a sentence. Denoting as $\boldsymbol{x}$ a sentence that consists of an array of words $(x_1, \ldots, x_n)$, and the array $\boldsymbol{y}$ that consists of a series of labels $(y_1, \ldots, y_n)$. We sample the buckets by using Algorithm 1. Note that we

MARC	Text A	Très mignons et de bonne qualité. La figurine est assez imposante mais conforme à la taille indiquée dans le descriptif.
	Text B	Jolis détails . home
XNLI	Text A	Ich musste anfagen Seminare zu belegen .
	Text B	Ich brauchte keine Vorbereitung .
PAWSX	Text A	Lo entrenó John Velázquez y en sus carreras más importantes lo montó el jinete Dale Romans.
	Text B	Lo entrenó John Velázquez, y el jinete Dale Romans lo montó en las carreras más importantes.
POS	Text A	(Lo,PRON), (sanno,VERB), (oramai,ADV), (quasi,ADV), (tutti,PRON), (che,SCONJ), (un,DET), (respiro,NOUN), (affannoso,ADJ) ...
NER	Text A	(Sempat,O), (pindah,O), (ke,O), (HJK,B-ORG), (dan,O), (1899,B-ORG), (Hoffenheim,I-ORG), (yang,O), (meminjamkannya,O), (ke,O) ...

Table 8: Example entries of the datasets. We convert the raw text to the mBERT format "Text A" and "Text B" (Devlin et al., 2019). For POS and NER, we list (word, tag) pairs in the sentence. Following Schwenk and Li (2018), we provide document indices of MLDoc for retrieving the documents from RCV1 and RCV2.

Language	Writing Script	81A Order of Subject, Object and Verb	85A Order of adposition and noun	86A Order of genitive and noun	87A Order of adjective and noun	88A Order of demonstrative and noun	89A Order of numeral and noun
English (EN)	Latin	SVO	Prepositions	No dominant order	Adjective-noun	Demonstrative-noun	Numeral-noun
Afrikaans (AF)	Latin	-	-	-	-	-	-
Arabic (AR)	Arabic	VSO	Prepositions	Noun-genetive	Noun-adjective	Demonstrative-noun	Numeral-noun
Bulgarian (BG)	Cyrillic	SVO	Prepositions	No dominant order	Adjective-noun	Demonstrative-noun	Numeral-noun
Bengali (BN)	Brahmic	SOV	-	-	-	-	-
German (DE)	Latin	No dominant order	Prepositions	Noun-genetive	Adjective-noun	Demonstrative-noun	Numeral-noun
Greek (EL)	Greek	No dominant order	Prepositions	Noun-genetive	Adjective-noun	Demonstrative-noun	Numeral-noun
Spanish (ES)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Demonstrative-noun	Numeral-noun
Estonian (ET)	Latin	SVO	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Basque (EU)	Latin	SOV	Postpositions	Genetive-noun	Noun-adjective	Noun-demonstrative	Numeral-noun
Persian (FA)	Perso-Arabic	SOV	Prepositions	Noun-genetive	Noun-adjective	Demonstrative-noun	Numeral-noun
Finnish (FI)	Latin	SVO	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
French (FR)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Demonstrative-noun	Numeral-noun
Hebrew (HE)	Hebrew	SVO	Prepositions	Noun-genetive	Noun-adjective	Noun-demonstrative	Numeral-noun
Hindi (HI)	Devanagari	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Hungarian (HU)	Latin	No dominant order	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Indonesian (ID)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Noun-demonstrative	Numeral-noun
Italian (IT)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Demonstrative-noun	Numeral-noun
Japanese (JA)	Ideograms	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Javanese (JV)	Latin	-	-	-	-	-	-
Georgian (KA)	Georgian	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Kazakh (KK)	Cyrillic	-	-	-	-	-	-
Korean (KO)	Hangul	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Malayalam (ML)	Brahmic	SOV	-	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Marathi (MR)	Devanagari	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Malay (MS)	Latin	-	-	-	-	-	-
Burmese (MY)	Brahmic	SOV	Postpositions	Genetive-noun	Noun-adjective	Demonstrative-noun	Noun-numeral
Dutch (NL)	Latin	No dominant order	Prepositions	Noun-genetive	Adjective-noun	Demonstrative-noun	Numeral-noun
Portuguese (PT)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Demonstrative-noun	-
Russian (RU)	Cyrillic	SVO	Prepositions	Noun-genetive	Adjective-noun	Demonstrative-noun	Numeral-noun
Swahili (SW)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Noun-demonstrative	Noun-numeral
Tamil (TA)	Brahmic	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Telugu (TE)	Brahmic	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Thai (TH)	Brahmic	SVO	Prepositions	Noun-genetive	Noun-adjective	Noun-demonstrative	Noun-numeral
Tagalog (TL)	Latin	VSO	-	Noun-genetive	No dominant order	Mixed	Numeral-noun
Turkish (TR)	Latin	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Urdu (UR)	Perso-Arabic	SOV	Postpositions	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun
Vietnamese (VI)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Noun-demonstrative	Numeral-noun
Yoruba (YO)	Latin	SVO	Prepositions	Noun-genetive	Noun-adjective	Noun-demonstrative	Noun-numeral
Chinese (ZH)	Chinese ideograms	SVO	No dominant order	Genetive-noun	Adjective-noun	Demonstrative-noun	Numeral-noun

Table 9: All languages for the experiments along with their ISO 639-1 codes, writing script, and linguistic features. "-" denotes lacking feature information from WALS.

sample with replacement for POS and NER.

D Additional Results

D.1 Learning Curve

Figure 6 visualizes the averaged learning curve of 10 out of 40 German 1-shot MARC buckets for which the best dev performance is obtained at epoch 1.

D.2 Numerical Values

The numerical values of the POS and NER FS-XLT results are shown in Table 13 and Table 12. The absolute performances of few-shot transfer without English source-training are shown in Table 11. The lexical overlap of target languages with EN for NER and POS is shown in Table 14.

292	584	78	27	19	526	361	43	31	40
45	630	250	64	11	176	554	162	80	28
24	259	497	196	24	65	298	369	218	50
4	69	237	525	165	22	87	176	471	244
6	25	75	357	537	16	27	42	245	670
599	316	36	33	16	570	262	45	56	67
255	543	112	70	20	269	416	126	125	65
136	401	266	174	23	143	284	219	270	84
60	262	283	322	73	63	163	190	395	189
38	83	127	462	290	32	39	59	314	555

Table 10: Numerical value of the confusion matrices in Figure 5. For 1-shot confusion matrices (right), we average results of 5 buckets and then round to integers.

	MLDoc		PAWSX			POS		NER	
	K=1	K=8	K=1	K=8		K=1	K=4	K=1	K=4
DE	52.63 ± 8.98	84.31 ± 3.60	53.03 ± 1.67	53.41 ± 1.47	RU	73.18 ± 4.42	86.65 ± 1.32	19.11 ± 6.94	35.57 ± 6.23
FR	50.80 ± 8.50	77.80 ± 4.44	54.05 ± 1.33	54.60 ± 0.97	ES	80.54 ± 4.17	90.26 ± 0.99	15.21 ± 5.98	39.37 ± 5.33
ES	50.30 ± 8.30	74.08 ± 6.48	54.14 ± 1.53	53.88 ± 1.72	VI	56.97 ± 5.16	72.00 ± 1.99	14.36 ± 4.28	29.63 ± 5.55
IT	41.34 ± 6.82	65.50 ± 4.21	-	-	TR	48.96 ± 3.15	59.65 ± 1.83	15.02 ± 5.58	37.81 ± 5.63
RU	46.74 ± 9.48	70.83 ± 5.63	-	-	TA	49.12 ± 4.67	64.96 ± 2.16	13.11 ± 4.55	27.42 ± 4.82
ZH	49.87 ± 10.44	76.15 ± 5.10	53.97 ± 1.79	54.17 ± 1.38	MR	60.26 ± 5.72	73.58 ± 2.39	15.68 ± 7.09	33.50 ± 6.02
JA	46.41 ± 6.59	66.85 ± 6.54	52.81 ± 0.96	52.97 ± 1.15	-	-	-	-	-
KO	-	-	53.92 ± 0.78	53.63 ± 0.99	-	-	-	-	-

Table 11: Target-adapting results without source-training. Numbers are mean and standard deviation of 40 runs.

	K=0	K=1	K=2	K=4
EN	95.39	-	-	-
AF	86.60	91.10 ± 1.11	92.12 ± 1.15	93.50 ± 0.56
AR	66.55	75.64 ± 1.09	77.01 ± 0.84	78.52 ± 0.67
BG	87.02	91.01 ± 0.97	91.97 ± 0.90	93.18 ± 0.56
DE	86.38	89.38 ± 0.90	90.21 ± 0.50	91.32 ± 0.43
EL	81.89	89.69 ± 1.05	90.53 ± 0.89	91.58 ± 0.72
ES	86.64	90.05 ± 1.01	91.19 ± 0.74	92.31 ± 0.52
ET	79.17	81.69 ± 1.09	83.05 ± 0.98	84.39 ± 0.56
EU	49.51	68.44 ± 2.47	71.94 ± 1.78	75.89 ± 1.20
FA	65.73	80.82 ± 2.14	82.81 ± 1.79	84.95 ± 1.16
FI	74.49	78.25 ± 1.22	79.65 ± 0.85	81.32 ± 0.82
FR	82.54	89.55 ± 1.08	90.84 ± 0.64	91.66 ± 0.60
HE	76.79	80.40 ± 1.42	82.42 ± 1.06	83.98 ± 0.83
HI	64.29	78.87 ± 1.26	80.80 ± 0.80	81.97 ± 0.92
HU	75.10	84.44 ± 1.40	86.31 ± 0.90	88.61 ± 0.67
ID	70.80	72.68 ± 1.08	73.64 ± 0.78	74.34 ± 0.75
IT	85.97	88.77 ± 0.87	89.93 ± 0.50	90.77 ± 0.59
JA	47.60	75.84 ± 1.68	78.46 ± 1.31	80.42 ± 0.98
KO	42.29	57.43 ± 1.36	59.92 ± 1.18	62.37 ± 1.22
MR	58.70	71.60 ± 2.52	74.89 ± 1.95	77.21 ± 1.77
NL	88.35	88.97 ± 0.73	89.55 ± 0.79	90.83 ± 0.54
PT	86.45	88.18 ± 0.70	88.98 ± 0.66	89.78 ± 0.38
RU	86.36	89.07 ± 0.76	89.85 ± 0.57	91.13 ± 0.51
TA	53.51	62.84 ± 2.69	66.30 ± 1.56	69.36 ± 1.13
TE	67.48	71.46 ± 2.58	75.72 ± 1.94	78.84 ± 1.44
TR	57.58	64.01 ± 1.53	66.02 ± 1.28	67.73 ± 0.82
UR	52.40	74.95 ± 2.15	78.53 ± 1.38	79.57 ± 1.24
VI	54.96	64.79 ± 2.33	69.39 ± 1.73	72.36 ± 1.51
ZH	63.01	74.15 ± 1.96	76.62 ± 1.39	79.42 ± 0.83

Table 12: Zero- (column K=0) and few- (columns K>0) shot cross-lingual transfer results (%) on POS test set.

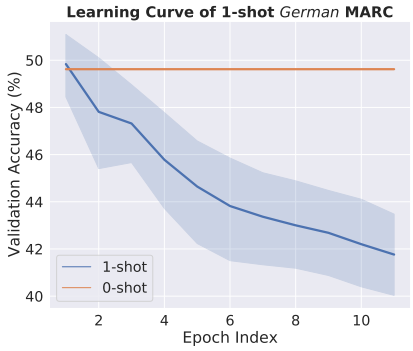


Figure 6: Early stopped 1-shot transfer (EN → DE) learning curve. The English-trained model overfits the 1-shot bucket quickly, showing decreasing dev performance during training.

	K=0	K=1	K=2	K=4
EN	83.65	-	-	-
AF	78.36	79.07 ± 1.47	79.69 ± 1.40	80.24 ± 1.16
AR	39.91	54.44 ± 6.74	60.51 ± 4.30	63.61 ± 2.65
BG	78.59	78.65 ± 0.38	78.70 ± 0.39	78.87 ± 0.48
BN	64.17	66.37 ± 1.69	66.66 ± 1.57	65.98 ± 2.11
DE	79.00	79.33 ± 0.71	79.61 ± 0.76	79.74 ± 0.73
EL	75.20	74.93 ± 0.79	75.18 ± 0.95	75.40 ± 0.93
ES	77.16	79.19 ± 1.97	80.28 ± 1.71	80.90 ± 1.94
ET	71.88	72.58 ± 1.17	73.60 ± 1.65	74.60 ± 1.59
EU	55.35	59.60 ± 3.32	61.59 ± 3.84	64.68 ± 2.96
FA	40.73	59.20 ± 5.34	68.55 ± 4.04	71.13 ± 3.45
FI	68.43	71.43 ± 2.61	73.92 ± 2.44	75.81 ± 2.15
FR	80.38	80.54 ± 0.93	81.08 ± 0.85	81.22 ± 0.93
HE	56.36	58.24 ± 2.25	59.43 ± 2.29	60.27 ± 2.43
HI	65.84	67.16 ± 1.61	67.56 ± 2.18	68.29 ± 1.76
HU	71.28	72.23 ± 1.33	73.03 ± 1.44	74.14 ± 1.61
ID	60.10	77.87 ± 6.31	78.57 ± 4.14	81.07 ± 1.50
IT	80.30	80.68 ± 0.79	81.00 ± 0.92	80.90 ± 1.12
JA	7.16	20.71 ± 7.07	28.23 ± 5.32	32.93 ± 6.03
JV	61.18	67.80 ± 4.72	69.79 ± 3.37	72.12 ± 3.34
KA	61.26	61.62 ± 1.09	62.25 ± 1.56	63.68 ± 1.66
KK	40.29	50.42 ± 5.49	54.97 ± 6.81	62.94 ± 4.55
KO	46.50	47.25 ± 1.36	48.69 ± 1.82	51.76 ± 2.30
ML	46.77	47.83 ± 2.30	49.51 ± 3.01	51.41 ± 3.31
MR	54.70	55.78 ± 2.54	57.22 ± 2.43	59.18 ± 3.13
MS	68.61	71.04 ± 3.07	74.51 ± 4.28	76.25 ± 3.04
MY	42.45	43.55 ± 3.88	46.03 ± 4.48	47.81 ± 4.28
NL	82.77	82.73 ± 0.43	82.83 ± 0.54	82.82 ± 0.46
PT	79.28	79.89 ± 0.99	80.39 ± 0.98	80.49 ± 0.95
RU	65.20	67.30 ± 2.38	68.78 ± 2.73	71.34 ± 2.82
SW	68.36	71.07 ± 4.28	70.08 ± 3.15	74.33 ± 5.25
TA	46.12	47.81 ± 1.81	49.86 ± 2.99	52.23 ± 2.63
TE	50.02	52.57 ± 1.91	54.02 ± 2.65	55.75 ± 2.72
TH	1.53	4.56 ± 4.87	6.08 ± 4.88	5.87 ± 4.14
TL	69.23	72.34 ± 2.25	72.63 ± 2.43	73.55 ± 2.25
TR	65.78	69.37 ± 2.24	69.53 ± 2.07	72.33 ± 2.85
UR	40.77	58.48 ± 6.51	63.38 ± 4.88	66.49 ± 4.64
VI	64.67	68.77 ± 3.54	69.64 ± 3.63	71.08 ± 3.28
YO	35.48	53.55 ± 6.19	58.22 ± 5.47	65.46 ± 7.10
ZH	13.95	32.84 ± 7.10	40.34 ± 5.32	48.49 ± 4.30

Table 13: Zero- (column K=0) and few- (columns K>0) shot cross-lingual transfer results (%) on NER test set.

	NER			POS		
	K=1	K=2	K=4	K=1	K=2	K=4
AF	4.54	8.75	13.44	4.97	6.11	7.90
AR	0.65	0.95	1.57	3.51	4.49	5.30
BG	0.98	2.19	3.23	-	-	-
BN	0.39	0.77	0.80	-	-	-
DE	8.75	13.20	20.61	9.36	15.33	21.48
EL	1.45	1.84	3.59	1.96	2.87	3.04
ES	6.29	10.59	19.66	10.00	17.53	22.63
ET	4.80	5.96	11.24	5.81	9.22	13.17
EU	3.77	5.55	12.31	2.60	3.45	4.69
FA	0.27	0.44	1.01	0.37	0.37	0.41
FI	5.61	9.05	15.66	4.59	7.03	8.78
FR	6.26	10.83	19.01	15.60	25.23	37.39
HE	0.86	1.90	3.23	1.22	1.93	2.26
HI	0.95	1.16	1.99	0.44	0.27	0.51
HU	5.07	9.19	14.35	3.18	3.92	4.15
ID	5.34	9.82	16.94	9.39	13.78	21.75
IT	7.89	10.94	21.27	11.99	16.15	21.35
JA	1.75	2.02	2.14	2.60	3.68	5.00
JV	2.49	3.05	3.44	-	-	-
KA	1.99	4.00	5.78	-	-	-
KK	0.89	1.22	2.11	-	-	-
KO	1.48	1.54	3.32	2.33	3.85	5.67
ML	0.36	1.04	1.30	-	-	-
MR	0.53	0.56	0.71	0.24	0.24	0.24
MS	4.86	7.44	13.70	-	-	-
MY	0.21	0.36	0.42	-	-	-
NL	7.18	10.65	20.14	7.94	11.42	16.79
PT	6.29	11.00	19.13	8.88	13.38	20.13
RU	1.60	2.34	3.77	4.15	6.11	9.32
SW	5.90	8.10	12.37	-	-	-
TA	0.65	1.54	2.08	1.32	1.28	1.62
TE	0.77	0.80	1.19	0.20	0.20	0.20
TH	1.63	1.87	2.08	-	-	-
TL	4.83	8.96	14.98	-	-	-
TR	4.89	8.48	16.43	2.09	2.26	3.01
UR	0.30	0.27	0.68	0.74	1.35	2.16
VI	4.33	8.39	13.41	1.62	2.16	2.90
YO	1.90	2.58	2.88	-	-	-
ZH	1.81	1.99	2.14	3.04	4.86	7.33

Table 14: Lexical overlap (per-mille) of target languages with EN for NER and POS using different K-shot buckets.

Chapter 7

LMTurk: Few-Shot Learners as Crowdsourcing Workers in a Language-Model-as-a-Service Framework

LMTurk: Few-Shot Learners as Crowdsourcing Workers in a Language-Model-as-a-Service Framework

Mengjie Zhao[†] **Fei Mi**[‡] **Yasheng Wang**[‡] **Minglei Li**[*]
Xin Jiang[‡] **Qun Liu**[‡] **Hinrich Schütze**[†]
[†]CIS, LMU Munich [‡]Huawei Noah's Ark Lab [*]Huawei Technologies Co., Ltd.
mzhao@cis.lmu.de, {mifei2,wangyasheng,jiang.xin,qun.liu}@huawei.com

Abstract

Vast efforts have been devoted to creating high-performance few-shot learners, i.e., large-scale pretrained language models (PLMs) that perform well with little downstream task training data. Training PLMs has incurred significant cost, but utilizing the few-shot learners is still challenging due to their enormous size. This work focuses on a crucial question: How to make effective use of these few-shot learners? We propose LMTurk, a novel approach that treats few-shot learners as crowdsourcing workers. The rationale is that crowdsourcing workers are in fact few-shot learners: They are shown a few illustrative examples to learn about a task and then start annotating. LMTurk employs few-shot learners built upon PLMs as workers. We show that the resulting annotations can be utilized to train models that solve the task well and are small enough to be deployable in practical scenarios. Active learning is integrated into LMTurk to reduce the amount of queries made to PLMs, minimizing the computational cost of running PLM inference passes. Altogether, LMTurk is an important step towards making effective use of current PLMs.[1]

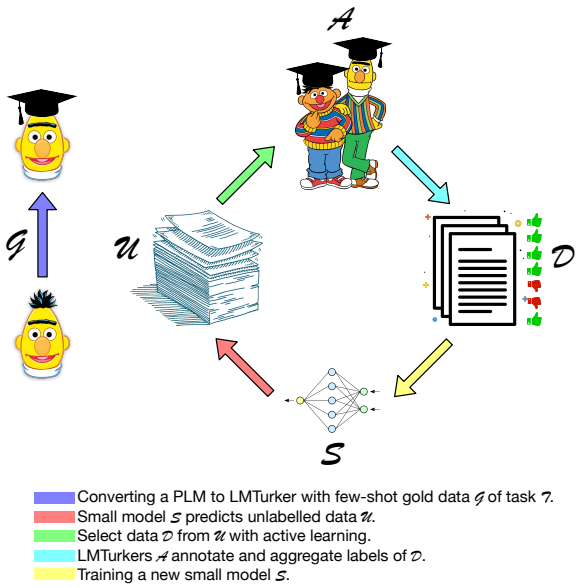


Figure 1: LMTurk overview; best viewed in color. We few-shot adapt PLMs to task $\mathcal{T}$ (left) and then use them as crowdsourcing workers in active learning. We show that these PLM workers are effective in training a small model $\mathcal{S}$ through a customized active learning loop (right). LMTurk is a novel way to take advantage of large-scale PLMs: It creates models small enough to be deployed in resource-limited real-world settings.

1 Introduction

Equipped with prolific linguistic features (Liu et al., 2019; Tenney et al., 2019; Belinkov and Glass, 2019; Rogers et al., 2020) and rich world knowledge (Petroni et al., 2019; Poerner et al., 2020; Kassner et al., 2021), large-scale pretrained language models (PLMs) have been shown to be versatile: They are now basic building blocks (Bommasani et al., 2021) of systems solving diverse NLP tasks in many languages (Wang et al., 2018, 2019; Hu et al., 2020; Xu et al., 2020; Khashabi et al., 2021; Park et al., 2021; Adelani et al., 2021).

Recent work shows that PLMs are effective *few-shot learners* (Brown et al., 2020; Schick and Schütze, 2021b; Gao et al., 2021; Tam et al., 2021) through *priming* (Brown et al., 2020; Tsimpoukelli et al., 2021) or *prompting* (Li and Liang, 2021; Liu et al., 2021b; Lester et al., 2021; Zhao and Schütze, 2021). Developing few-shot learners is crucial because current NLP systems require much more data than humans (Yin et al., 2020). Few-shot learners tend to perform well; however, they still fall behind systems trained with abundant data. Furthermore, the enormous size of PLMs hinders their deployment in practice. For example, it is challenging to fit the 11 billion T5-XXL (Raffel et al., 2020) model on a single regular GPU.

Our goal in this paper is to devise methods that make *more effective use of current few-shot learners*. This is crucial because an increasing number

[1]Resources are available at: github.com/lmturk

of gigantic few-shot learners are trained; how to use them effectively is thus an important question. In particular, we want an alternative to hard-to-deploy huge models. At the same time, we want to take full advantage of the PLMs' strengths: Their versatility ensures wide applicability across tasks; their vast store of knowledge about language and the world (learned in pretraining) manifests in the data efficiency of few-shot learners, reducing labor and time consumption in data annotation.

In this work, we propose **LMTurk**, **L**anguage **M**odel as mechanical **Turk**. Our basic idea (see Figure 1) is that, for an NLP task $\mathcal{T}$, *we treat few-shot learners as non-expert workers*, resembling crowdsourcing workers that annotate resources for human language technology. We are inspired by the fact that we can view a crowdsourcing worker as a type of few-shot learner: A few examples demonstrating $\mathcal{T}$ teach her enough about $\mathcal{T}$ to conduct effective annotation. For example, Snow et al. (2008) train workers with a few examples of annotating emotion; He et al. (2015) conduct short training sessions for workers before annotation; Lee et al. (2021) train workers with learning curricula.

Snow et al. (2008) pioneered crowdsourcing in NLP (Howe et al., 2006; Howe, 2008), motivated by the high cost of TreeBank annotation (Marcus et al., 1993; Miller et al., 1993). Crowdsourcing organizes human workers over the Web to annotate data. Workers need not be experts to be effective, resulting in reduced *per-label cost*. Active learning (Hachey et al., 2005; Felder and Brent, 2009) can be incorporated (Laws et al., 2011) to further decrease annotation cost, by lowering *the number of labels* to be annotated. LMTurk treats PLM-based few-shot learners as non-expert workers that produce training sets, which are then used to train a small machine learning model $\mathcal{S}$ specialized for $\mathcal{T}$. This scenario is analogous to active learning. We achieve two benefits: (i) low annotation cost because humans only need to annotate a few shots of data; (ii) solving practical NLP tasks with small models that are more real-world deployable.

LMTurk resonates with Laws et al. (2011)'s earlier idea of combining crowdsourcing and active learning. They consider human workers as "noisy annotators" while we explore the utilization of modern NLP few-shot learners (built upon machine learning models) as workers – which have the advantage of being free, instantly interactive, fast, responsive, and non-stopping.

Our **contributions**: (i) We propose LMTurk, a method that uses few-shot learners as crowdsourcing workers. Figure 1 shows the overview of LMTurk. (ii) We vary an array of important design choices, identifying strengths and weaknesses of LMTurk. (iii) Unlike much work on active learning in a synthetic oracle setting, we develop methods for handling the varying quality of annotation that does not come from an oracle. (iv) We extensively evaluate LMTurk on five datasets, showing that LMTurk can guide a small model $\mathcal{S}$ to progressively improve on $\mathcal{T}$. $\mathcal{S}$ can then be deployed in practical scenarios. (v) This is the first work showing that few-shot learners give rise to effective NLP models through crowdsourcing and active learning – with the benefits of low annotation cost and practical deployability.

2 Related Work

Few-shot learners in NLP. Significant progress has been made in developing (Devlin et al., 2019; Peters et al., 2018; Yang et al., 2019; Brown et al., 2020), understanding (Liu et al., 2019; Tenney et al., 2019; Belinkov and Glass, 2019; Hewitt and Liang, 2019; Hewitt and Manning, 2019; Zhao et al., 2020a; Rogers et al., 2020), and utilizing (Houlsby et al., 2019; Zhao et al., 2020b; Brown et al., 2020; Li and Liang, 2021; Schick and Schütze, 2021a; Lester et al., 2021; Mi et al., 2021a) PLMs. Brown et al. (2020), Schick and Schütze (2021a), and Liu et al. (2021b) show that PLMs can serve as data-efficient few-shot learners, through priming or prompting (Liu et al., 2021a). For example, GPT3 achieves near state-of-the-art performance on COPA (Roemmele et al., 2011) with only 32 annotated data.

However, little to no work discusses or explores the actual *practical utility* of these few-shot learners. We aim to develop effective methods of utilizing them in practical scenarios.

Crowdsourcing has a long history in human language technology (Alonso et al., 2008; Callison-Burch, 2009; Trautmann et al., 2020); specialized workshops were organized (Callison-Burch and Dredze, 2010; Paun and Hovy, 2019). It has numerous applications (Yuen et al., 2011), but we focus on its application as voting systems. To reduce *per-label* cost, crowdsourcing organizes non-expert human workers distributed across the Web for annotation, instead of employing linguistic experts (Jamison and Gurevych, 2015; Bhardwaj et al., 2019;

Nangia et al., 2021). Snow et al. (2008) show that averaging ten crowdsourced labels matches an expert-level label for recognizing textual entailment (Dagan et al., 2006). Paun et al. (2018) show that incorporating structure in annotation models is important. Measuring label disagreements is also crucial (Dumitrache et al., 2021).

LMTurk utilizes NLP few-shot learners as non-expert workers. The few-shot training data can be viewed as the examples shown to humans before annotating. The process is free, fast, responsive, and non-stopping.

Active learning (AL; Cohn et al. (1996); Settles (2009)) strives to reduce *the number of examples* to be annotated via identifying informative examples with acquisition functions. Settles and Craven (2008) evaluate AL algorithms for sequence labeling. Zhang et al. (2017); Shen et al. (2017); Siddhant and Lipton (2018) apply AL to deep neural networks. Simpson and Gurevych (2018) devise a scalable Bayesian preference learning method for identifying convincing arguments. Lee et al. (2020) propose to consider user feedback in AL systems. Ein-Dor et al. (2020) explore AL for BERT. Schröder and Niekler (2020) review text classification with AL. Liang et al. (2020); Margatina et al. (2021) integrate contrastive learning into AL. Zhang and Plank (2021) identify examples with datamap (Swayamdipta et al., 2020).

We incorporate AL in LMTurk to reduce the amount of examples to be annotated by PLMs, reducing the computational cost of running several inference passes. This contributes to a more environmentally friendly (Strubell et al., 2019; Schwartz et al., 2020; Patterson et al., 2021) scenario.

Perhaps closest to our work, Yoo et al. (2021) conduct data augmentation via priming GPT3 and Wang et al. (2021) mix human- and GPT3-annotated data, focusing on cost analysis. GPT3 is utilized in a Language-Model-as-a-Service form by OpenAI, which is not free.[2] Also, strategies of priming GPT3 may not generalize well to other PLMs. For example, priming strategies have to adapt to GPT3's maximum sequence length. However, maximum sequence length – as a hyperparameter – could vary across PLMs. In this work, we prompt publicly available free PLMs. This also makes the process more flexible; for example, the PLM can be updated with gradient descent.

[2] https://beta.openai.com/pricing

3 LMTurk

3.1 Training few-shot learners

We first adapt a PLM to task $\mathcal{T}$ with a few-shot human-labeled gold dataset $\mathcal{G} = \{\mathcal{G}_{train}; \mathcal{G}_{dev}\}$ of $\mathcal{T}$. This procedure mimics one of the initial but crucial steps in crowdsourcing: A few example annotations are shown to the workers, demonstrating $\mathcal{T}$; workers learn about the task and then start annotating (Snow et al., 2008; He et al., 2015; Roit et al., 2020; Trautmann et al., 2020; Lee et al., 2021).

We achieve this adaptation through P-Tuning (Liu et al., 2021b). Taking movie review classification as an example, the goal is to associate a binary label y from {-1, +1} to an input sentence $\mathbf{x} = (x_1, ..., x_n)$ where x_i refers to a token. Unlike finetuning and its variants (Devlin et al., 2019; Houlsby et al., 2019; Zhao et al., 2020b) that train a classifier head, P-Tuning reformulates a sentence into a cloze-style query; the PLM is then requested to respond to the query with an answer selected from a list of candidates. Concretely, an input pair

(x, y) = ("watching it leaves you giddy.", -1)

is reformulated to:

"[v] watching it leaves you giddy. It is [MASK] ."

in which the underlined tokens are prompting words that give the model a hint about $\mathcal{T}$. "[v]" – whose trainable embedding vector is randomly initialized – is a prompting token injecting extra free parameters. The PLM is then requested to pick a word from {"bad", "good"} to fill in the position of "[MASK]". A mapping {"bad" $\rightarrow$ -1, "good" $\rightarrow$ +1} is used to transform the selected answer to a label such that standard evaluation measures like accuracy can be computed. Prompting has been shown to effectively adapt a PLM to $\mathcal{T}$ *with only a few annotations*; see (Liu et al., 2021a) for a comprehensive review of prompting. We refer to a PLM adapted to $\mathcal{T}$ as an **LMTurker** A.

We select prompting words and mappings based on the small development set $\mathcal{G}_{dev}$. §4.2 provides details on prompting and datasets.

3.2 Aggregating annotations

Individual workers are subject to annotation biases (Snow et al., 2008); therefore, crowdsourcing often collects labels from several workers (Yuen et al., 2011) for an example $\mathbf{x}$ and then aggregates them for quality control (Alonso et al., 2008). It is straightforward to obtain a group of LMTurkers

$\mathcal{A} = \{A_1, ..., A_k\}$, by adapting the PLM to $\mathcal{T}$ with k different prompts. A querying sentence $\mathbf{x}$ is then annotated by every LMTurker, resulting in a list of labels $\mathbf{y} = [y_1, ..., y_k]$. We evaluate different methods aggregating $\mathbf{y}$ to a single label $\hat{y}$.

BestWorker. Among the k LMTurkers, we pick the one performing best on the dev set $\mathcal{G}_{dev}$.

MajorityVoting. We select the most frequent label in $\mathbf{y} = [y_1, ..., y_k]$ as $\hat{y}$.

To estimate an LMTurker's confidence on label y_i, we compare the logits[3] computed by the PLM:

$$y_i = \arg\max(\text{logit}(y^1), ..., \text{logit}(y^N)),$$

where N refers to the label set size, e.g., N=2 for y from {-1, +1}. We then can evaluate several methods of aggregating annotations according to PLM logits.

LogitVoting. We average the logits from all k LMTurkers $\{A_1, ..., A_k\}$ to compute $\hat{y}$:

$$\hat{y} = \arg\max(\tfrac{1}{k}\textstyle\sum_{i=1}^{k} \text{logit}(y_i^1), ..., \tfrac{1}{k}\sum_{i=1}^{k} \text{logit}(y_i^N)).$$

WeightedLogitVoting. We use LMTurkers' performance on $\mathcal{G}_{dev}$ to weight their logits and then aggregate the predictions:

$$\hat{y} = \arg\max(\textstyle\sum_{i=1}^{k} w_i \text{logit}(y_i^1), ..., \sum_{i=1}^{k} w_i \text{logit}(y_i^N))$$
$$w_i = f(A_i, \mathcal{G}_{dev}) / \textstyle\sum_{i=1}^{k} f(A_i, \mathcal{G}_{dev})$$

where $f(A_i, \mathcal{G}_{dev})$ is the performance of the ith LMTurker A_i on $\mathcal{G}_{dev}$.

We collect and aggregate annotations from five LMTurkers, i.e., we use k=5 in our experiments.

3.3 Training a small model $\mathcal{S}$

After adapting LMTurkers to $\mathcal{T}$ through prompting with the few-shot gold dataset $\mathcal{G}$, we next train a small model $\mathcal{S}$ specialized to solve $\mathcal{T}$. Though large PLMs are versatile and strong performers, training and inference are faster and more efficient for small models: They are more deployable in resource-restricted scenarios, e.g., on edge devices (Jiao et al., 2020).

We mimic pool-based active learning (AL; Settles (2009)) to train $\mathcal{S}$. The motivation is to avoid frequent querying of LMTurkers $\mathcal{A}$ because energy and time consumption of PLM inference is costly when the number of queries and $|\mathcal{A}|$ are large.

Concretely, pool-based AL assumes a large collection of unlabeled data $\mathcal{U} = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ for $\mathcal{T}$. $\mathcal{S}$ is first trained with $\mathcal{G} = \{\mathcal{G}_{train}; \mathcal{G}_{dev}\}$. After that, a group of examples $\mathcal{B}$ from $\mathcal{U}$ is sampled (c.f. §3.3.1), which LMTurkers annotate. Next, the annotated and aggregated examples $\mathcal{B}'$ are concatenated with $\mathcal{G}$ to train $\mathcal{S}$. The procedure is repeated iteratively, such that the training data for $\mathcal{S}$ keeps expanding. We denote as $\mathcal{S}^j$ the model trained after the jth iteration. Note that $\mathcal{S}$ is trained from scratch in each iteration (Cohn et al., 1994).

3.3.1 AL acquisition function

At the beginning of the jth iteration, a straightforward strategy of sampling $\mathcal{B}$ from $\mathcal{U}$ is **random sampling**. AL promises to select a more informative $\mathcal{B}$ such that the trained $\mathcal{S}^j$ performs better, under the same budget. These strategies – or *acquisition functions* – rely on $\mathcal{S}^{j-1}$, i.e., $\mathcal{S}$ from the previous iteration: $\mathcal{S}^{j-1}$ is employed to infer $\mathcal{U}$ to obtain labels and logits $\mathcal{P}^{j-1} = \{(y_1, \mathbf{c}_1), ..., (y_M, \mathbf{c}_M)\}$; each $\mathbf{c}_i$ contains the logits of the N labels; $y_i = \arg\max(\mathbf{c}_i)$. We explore two common AL acquisition functions: Entropy (Roy and McCallum, 2001) and LeastConfident (Lewis and Gale, 1994).

Entropy selects from $\mathcal{P}^{j-1}$ examples with the largest prediction entropy, computed using $\mathbf{c}$. Large entropy of an example $\mathbf{x}$ implies that $\mathcal{S}^{j-1}$ is unsure about which label to select; $\mathbf{x}$ is then a query made to LMTurkers to obtain its label $\hat{y}$. $(\mathbf{x}, \hat{y})$ is subsequently added to $\mathcal{G}_{train}$ for training $\mathcal{S}^j$.

LeastConfident selects from $\mathcal{P}^{j-1}$ examples for which the maximum logit in $\mathbf{c}$ is the smallest. Selected examples are then annotated and added to $\mathcal{G}_{train}$ for training $\mathcal{S}^j$.

Our AL setup is fairly standard, both in terms of acquisition functions and iterative enlargement by new sampled data $\mathcal{B}$ at iteration j labeled by $\mathcal{S}^{j-1}$.

3.3.2 Considering annotation quality

As in any realistic AL scenario, annotations are not perfect: LMTurkers do not score perfectly on $\mathcal{T}$. As a result, *annotation quality of LMTurkers needs to be taken into consideration before training* $\mathcal{S}^j$. Denoting the training data of $\mathcal{S}^j$ as $\mathcal{D}^j$, we explore a strategy of processing $\mathcal{D}^j$, based on LMTurker logits $\mathbf{l}$.

InstanceTresholding. We preserve examples $(\mathbf{x}, \hat{y}, \mathbf{l}) \in \mathcal{D}^j$ for which entropy computed on $\mathbf{l}$ is smallest. $\mathcal{G}^{train}$ is always preserved because it is human-labeled gold data. Note that this is different from the strategy of sampling $\mathcal{B}$, where we select from $\mathcal{P}^{j-1}$ examples to which $\mathcal{S}^{j-1}$ is most unsure

[3] Calibration can be conducted to further improve the estimation (Guo et al., 2017). We leave this to future work.

(computed with **c**). We evaluate[4] the effectiveness of processing $\mathcal{D}^j$ before training $\mathcal{S}^j$ in §5.6.

3.4 Summary of LMTurk

LMTurk can be viewed as intermediate between self training (Yarowsky, 1995; Abney, 2004; Lee et al., 2013; Mi et al., 2021b) and AL. Unlike self training, LMTurk employs *external* models provide labels to $\mathcal{S}$. Different from the artificial setup used in many AL experiments, the provided labels *do not have oracle quality*; so $\mathcal{S}$ must use the annotations more carefully. We next conduct experiments investigating the effectiveness of LMTurk.

4 Datasets and Setup

4.1 Dataset

We evaluate LMTurk on five datasets: Binary (SST2) and fine-grained (five classes) sentiment classification (SST5) with the Stanford Sentiment TreeBank (Socher et al., 2013); news article topic classification with the AG's News Corpus (AGNews; Zhang et al. (2015)); recognizing textual entailment (RTE; Dagan et al. (2006)); assessing linguistic acceptability (CoLA; Warstadt et al. (2019)). Appendix §A reports dataset statistics. SST2/SST5 and AGNews are widely used in crowdsourcing and AL (Laws et al., 2011; Ein-Dor et al., 2020; Margatina et al., 2021; Zhang and Plank, 2021). RTE and CoLA assess the models' ability to understand textual entailment and linguistic phenomena – as opposed to text categorization. We report Matthew's correlation coefficient for CoLA and accuracy for the others (Wang et al., 2018).

Few-shot datasets. Recall LMTurk uses a small human-annotated dataset $\mathcal{G} = \{\mathcal{G}_{train}; \mathcal{G}_{dev}\}$. Denoting n as the number of shots *per class*, we sample $\mathcal{G}^n_{train}$ and $\mathcal{G}^n_{dev}$ for each of $n \in \{8, 16, 32\}$. For SST2, RTE, and CoLA, we use the train and dev sets of GLUE (Wang et al., 2018); $\mathcal{G}^n_{train}$ and $\mathcal{G}^n_{dev}$ are sampled from the train set; the dev set is used as the test set. For SST5 and AGNews, we use the official datasets; $\mathcal{G}^n_{train}$ ($\mathcal{G}^n_{dev}$) is sampled from the train (dev) set; we report performance on the test set. We repeat the sampling process with three random seeds.

4.2 Training setup

Brown et al. (2020) show that large model size is necessary for strong few-shot performance. We use ALBERT-XXLarge-v2 (Lan et al., 2020) – of size 223M parameters – as our large PLM, which is adapted to be an LMTurker A of $\mathcal{T}$ with $\mathcal{G}$. With parameter reuse, ALBERT-XXLarge-v2 outperforms larger models like the 334M BERT-large (Devlin et al., 2019). In contrast, $\mathcal{S}$ must be small to be deployable in practical scenarios. We use TinyBERT-General-4L-312D (Jiao et al., 2020), which has 14.5M parameters.

We train – with prompting – the large PLM with $\mathcal{G}$ for 100 batch steps using batch size 16, AdamW (Loshchilov and Hutter, 2019) and learning rate 5e-4 with linear decay. We prompt the large PLM five times to obtain five LMTurkers; Appendix §C shows prompting details. At each iteration, we finetune $\mathcal{S}$ for 20 epochs using batch size 32, Adam (Kingma and Ba, 2015) and learning rate 5e-5. Each experiment is run with three different random seeds. We use PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020).

[4]Motivated by Wang et al. (2017), we also investigate the effectiveness of weighting training examples. However, we do not observe noticeable improvements of task performance. We list more details in Appendix §E.

	Schick and Schütze (2021a,b)	Gao et al. (2021)	Ours
SST2	n/a	93.0±0.6	93.08±0.62
SST5	n/a	49.5±1.7	46.70±0.93
RTE	69.8	71.1±5.3	70.88±1.70
AGN.	86.3±0.0	n/a	87.71±0.07
CoLA	n/a	21.8±15.9	19.71±1.89

Table 1: LMTurkers achieve comparable few-shot performance with the literature. We refer to *PET* results in Schick and Schütze (2021a,b) and results of *Prompt-based FT (auto) + demonstrations* in Gao et al. (2021).

5 Experiment

5.1 Few-shot performance (non-iterative)

We compare few-shot performance of LMTurkers and the small model $\mathcal{S}$ when *only* $\mathcal{G}$ *is used*. LMTurker performance is comparable to prior work (Schick and Schütze, 2021a,b; Gao et al., 2021) as shown in Table 1.

Figure 2 compares performance of LMTurkers and $\mathcal{S}$. Appendix §B Table 3 reports numeric values. LMTurkers perform clearly better than $\mathcal{S}$ on CoLA, SST5, AGNews, and SST2; e.g., for SST2, for train/dev size 16, LMTurker accuracy is 93.08% vs. 75.83% for $\mathcal{S}$. LMTurkers' superiority over $\mathcal{S}$ on RTE is modest. As an inference task, RTE is more challenging than classification (e.g., AGNews). We hypothesize that current few-shot learners require more data than $\mathcal{G}^{32}$ to process difficult tasks better than $\mathcal{S}$. Scaling up to even larger PLMs is also a promising direction (Brown et al., 2020;

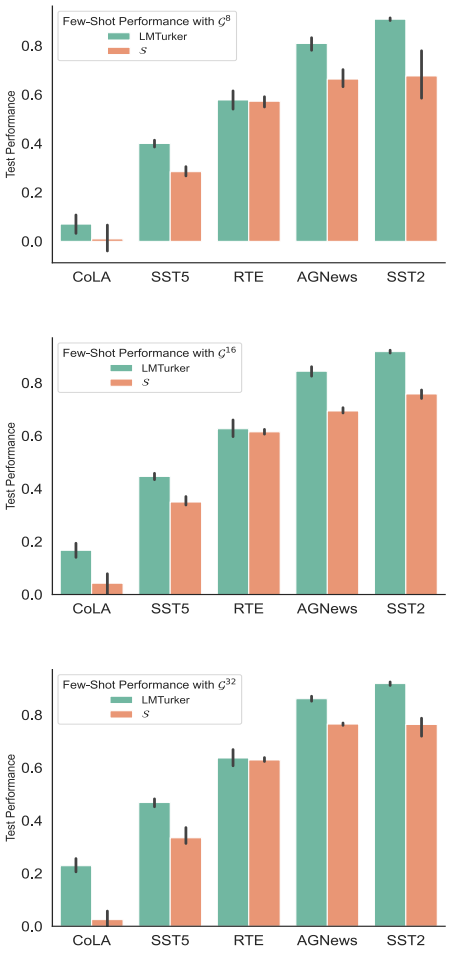


Figure 2: *Few-shot* test set performance of LMTurkers and $\mathcal{S}$. We use the few-shot gold datasets $\mathcal{G}^8$ (top), $\mathcal{G}^{16}$ (middle), and $\mathcal{G}^{32}$ (bottom).

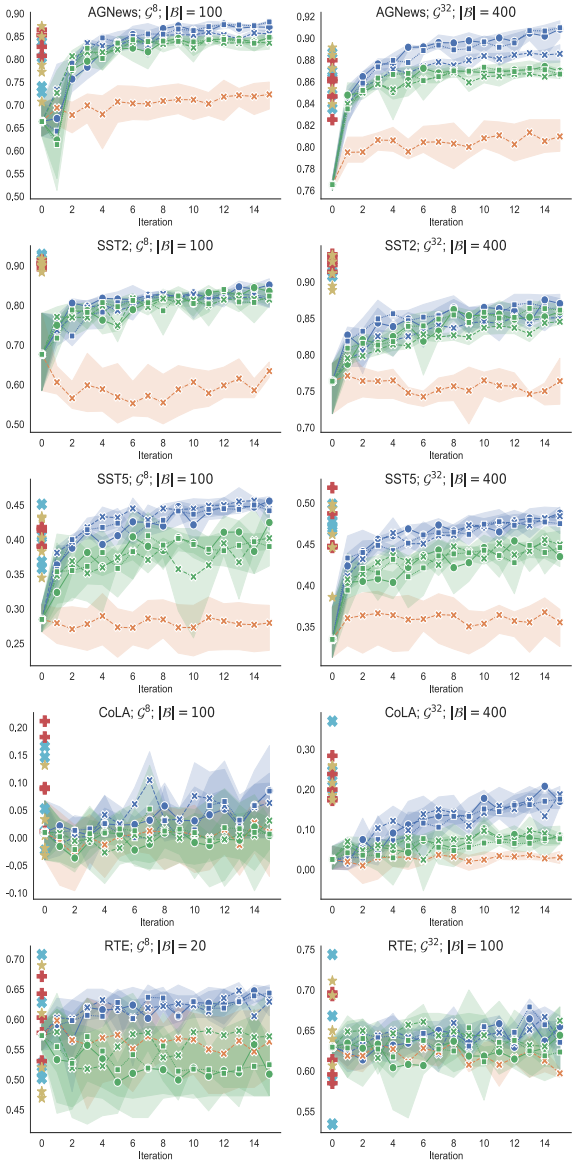


Figure 3: Improving $\mathcal{S}$ with active learning (blue), self training (orange), and LMTurk (green). Free markers at step zero show LMTurker performances; colors distinguish random seeds. Three acquisition functions are: Entropy (●), LeastConfident (■), random sampling (✖). At iteration j, each experiment is repeated three times; we show mean and standard deviation. Appendix Figure 9 visualizes more results.

Lester et al., 2021).

Overall, LMTurkers outperform $\mathcal{S}$ with clear margins, evidencing that their annotations can serve as supervisions for training $\mathcal{S}$. We next conduct iterative training to improve performance of $\mathcal{S}$ on $\mathcal{T}$ with supervisions from LMTurkers.

5.2 Iterative training

We investigate the effectiveness of LMTurk by simulating scenarios analogous to active learning. Concretely, we compare three schemes of annotating the sampled data $\mathcal{B}$ at each annotation iteration j:

- Active learning (AL). We use $\mathcal{B}$'s *gold labels* to show how $\mathcal{S}$ performs with expert annotations. Gold labels are ideal, but costly because expert annotators need to be employed.

- Self training (ST). We use $\mathcal{S}^{j-1}$, the model trained in the previous iteration, to annotate $\mathcal{B}$ (Yarowsky, 1995; Abney, 2004; Lee et al., 2013). ST trades supervision quality for annotation cost; no extra cost is introduced. Because there is no external supervision, ST is expected to be a baseline.

- LMTurk. We query the LMTurkers to annotate $\mathcal{B}$. LMTurkers are machine learning models, so there is no human labor. Based on the findings in Figure 2, LMTurker supervisions

are expected to have better quality than those of ST. Yet LMTurk could fall behind AL because LMTurker labels are not gold labels.

When sampling $\mathcal{B}$ from $\mathcal{U}$ at each iteration j, we consider the strategies described in §3.3. We employ Random for all three schemes and Entropy/LeastConfident for AL/LMTurk. Entropy and LeastConfident rely on $\mathcal{S}^{j-1}$. Regarding the number of sampled examples, we experiment with $|\mathcal{B}|$=100 and $|\mathcal{B}|$=400 for SST2, SST5, AGNews, CoLA. Due to RTE's small size, we use $|\mathcal{B}|$=20 and $|\mathcal{B}|$=100. We run for 15 iterations of improving $\mathcal{S}$. To aggregate annotations from LMTurkers, we use MajorityVoting (§3.2), which is widely used in crowdsourcing. See §5.3 for a comparison of various aggregation methods.

Figure 3 **compares AL, ST, and LMTurk.** ST (orange) noticeably helps $\mathcal{S}$ to perform progressively better on AGNews, e.g., when comparing $\mathcal{S}^{15}$ to $\mathcal{S}^{0}$ shown in the first row, especially when $|\mathcal{B}|$=400. However, we do not identify clear improvements when looking at other tasks. Except for RTE-$\mathcal{G}^8$, ST clearly falls behind AL and LMTurk. This inferior performance meets our expectation because there is no external supervision assisting $\mathcal{S}$ to perform better on $\mathcal{T}$. In what follows, we omit ST for clearer visualization and discussion.

AL (blue) performs the best in most experiments. However, this comes with extra costs that are not negligible: *At each iteration*, human annotators need to annotate 100–400 sentences.

LMTurk (green) holds a position between AL and ST on AGNews, SST2, SST5, and CoLA. Somehow surprisingly, LMTurk performs almost comparably to AL on SST2. Unlike AL, LMTurk requires very little human labor; the only human annotation throughout the entire process is the few-shot gold dataset $\mathcal{G}$. In contrast, AL has high human annotation cost, e.g., 1000–4000 examples by iteration ten. LMTurk also shows clear performance improvements over ST.

Results on RTE are noisy; we conjecture this is due to its very small test set (277 examples). We do not observe performance improvement of $\mathcal{S}$ along the iterations in experiment RTE-$\mathcal{G}^{32}$-$|\mathcal{B}|$=100, likely due to saturated task performance: TinyBERT-General-4L-312D ($\mathcal{S}$) achieves 66.6% on RTE for the full train set (Jiao et al., 2020).

Comparing sampling strategies. Entropy (●) and LeastConfident (■) outperform random sampling (✖) in AGNews and SST2 with noticeable

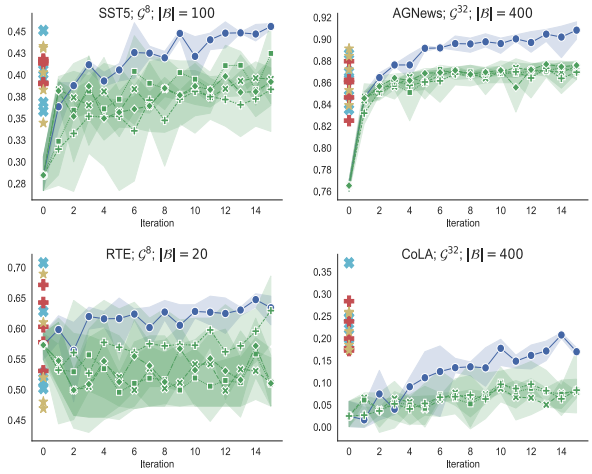


Figure 4: Comparing strategies of aggregating LMTurker annotations. We compare LMTurk (green) with AL (blue). Strategies: LogitVoting (✖), MajorityVoting (■), WeightedLogitVoting (♦), BestWorker (✚). AL uses gold labels without aggregation (●).

margins – for both AL and LMTurk, especially when $|\mathcal{B}|$=400. They also surpass random sampling when using LMTurk for SST5 and CoLA with $\mathcal{G}^8$. In other words, Entropy and LeastConfident assist LMTurk to achieve the same performance as of using random sampling, but with fewer annotations. For example in AGNews-$\mathcal{G}^8$-$|\mathcal{B}|$=100, LeastConfident at iteration six already achieves comparable performance as random sampling at iteration eleven. This is economically and environmentally beneficial because the number of queries made to LMTurkers, i.e., the cost of running inference passes on the array of large PLMs, is significantly reduced.

Overall, we show that LMTurk can be used to create datasets for training a specialized model $\mathcal{S}$ of solving $\mathcal{T}$ in practical scenarios. To reduce computational cost, we use only Entropy in what follows.

5.3 Design choice 1: Aggregation strategies

Figure 4 compares effectiveness of different strategies of aggregating LMTurker annotations (§3.2). Looking at SST5 and AGNews results (top two images), we observe that committee-style aggregation (LogitVoting (✖), MajorityVoting (■), and WeightedLogitVoting (♦)) generally outperforms BestWorker (✚), which simply relies on the LMTurker performing best on $\mathcal{G}_{dev}$. LMTurkers perform well on these two datasets as shown by the free markers at iteration zero; ensembling their predictions results in higher-quality datasets.

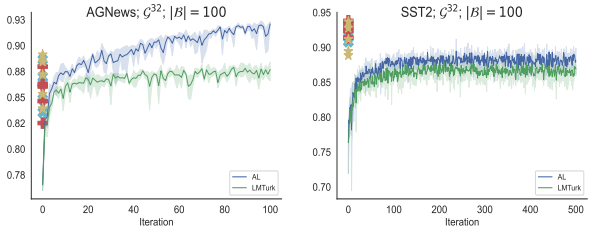


Figure 5: Running more iterations of improving $\mathcal{S}$ with AL and LMTurk. Sampling strategy Entropy is used for both methods; WeightedLogitVoting is used for aggregating LMTurker annotations.

In contrast, BestWorker (✚) has stellar performance on RTE (bottom-left), outperforming committee-style aggregation. Note that even the LMTurkers do not perform really well in this experiment, as shown by the free markers at iteration zero – some LMTurkers even perform worse than $\mathcal{S}$. Ensembling these low-quality annotations seems a worse option than simply relying on the best LMTurker. For CoLA, we observe comparable performance of different aggregation strategies.

5.4 Design choice 2: More iterations

We hypothesize that AL performance is an upper bound for performance when $\mathcal{S}$ is trained with LMTurker annotations – recall that the AL annotations are gold labels. Figure 5 compares AL and LMTurk when running 100 iterations of improving $\mathcal{S}$ on AGNews and 500 iterations on SST2. As expected, AL outperforms LMTurk because the pool of human-annotated data expands. The performance of $\mathcal{S}$ progressively approaches that of the LMTurkers; LMTurk performs comparably to AL in SST2, however, no human labor is required.

5.5 Design choice 3: Distilling logits

We can view LMTurk as a kind of distillation (Hinton et al., 2015): The ability of LMTurkers to solve $\mathcal{T}$ is progressively transferred to $\mathcal{S}$. In this section, we explore the utility of distillation: We train $\mathcal{S}$ with predicted logits[5] instead of discrete labels from LMTurkers. Concretely, we train $\mathcal{S}$ by reducing the KL divergence between its predicted probability distribution (over the label set) and the probability distribution from LMTurkers.

[5]Distilling with intermediate activations likely to further improve performance of $\mathcal{S}$. However, note that PLM intermediate activations are not always available in a Language-Model-as-a-Service framework.

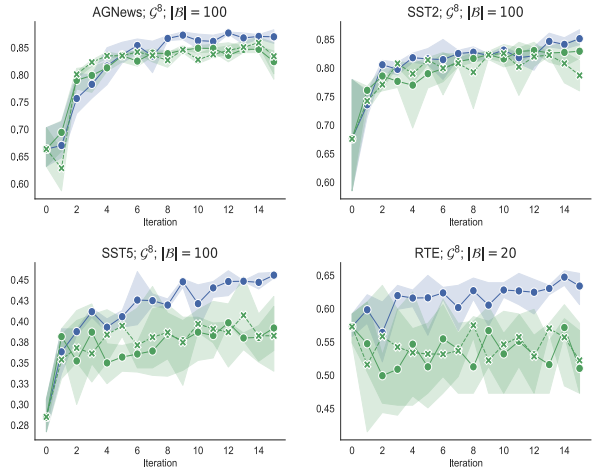


Figure 6: Performance of AL and LMTurk with discrete labels (●) vs. with KL divergence (✖). Entropy is used as the sampling strategy and WeightedLogitVoting is used to aggregate worker annotations.

Figure 6 shows that training $\mathcal{S}$ with KL divergence noticeably improves over discrete labels on AGNews and SST5. This is expected: AGNews and SST5 have larger label set size (four and five) such that the probability distribution over the label set is more informative than that of the binary classification tasks SST2 and RTE.

5.6 Design choice 4: Quality-based filtering

One key difference between AL and LMTurk is that LMTurkers are not oracles: Their labels are not perfect. Hence, it is reasonable to consider processing the training data, denoted as $\mathcal{D}^j$, for $\mathcal{S}^j$, instead of using it indiscriminately as in AL.

InstanceTresholding (§3.3.2) preserves annotations in $\mathcal{D}^j$ for which LMTurkers have the smallest prediction entropy. Concretely, we rank all annotations $(\mathbf{x}, \hat{y}, \mathbf{l}) \in \mathcal{D}^j$ by *entropy*($\mathbf{l}$) and then keep the τ percent smallest. Note that we always preserve the human-labeled few-shot data $\mathcal{G}_{train}$. We experiment with $\tau \in \{10\%, \ldots, 90\%, 100\%\}$.

Figure 7 left shows the performance of $\mathcal{S}$; Figure 7 right tracks the status of $\mathcal{D}^j$. To measure quality, we compute the accuracy of LMTurker annotations on $\mathcal{D}^j$ (compared to gold labels); see the lineplots and the left y-axis. We also report the size of $\mathcal{D}^j$ as scatter plots (right y-axis).

We observe that τ=10%, i.e., keeping only the 10% most certain examples, gives the worst performance. This is most obvious at iteration three for SST2: The performance drops to near the majority baseline (≈50%). This is because $\mathcal{D}^3$ is small and

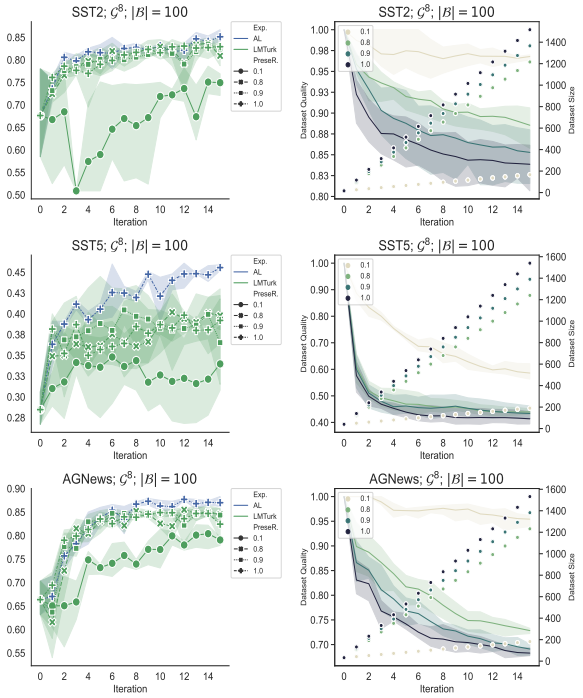


Figure 7: Training $\mathcal{S}$ with examples for which LMTurkers have low entropy. We report performance of $\mathcal{S}$ (left), number and quality (measured by accuracy) of the preserved examples (right) at each iteration.

unbalanced: It has eight negative (from $\mathcal{G}^{train}$) and 38 positive examples. However, using all the LMTurker annotations (τ=100%) may not be optimal either. This is noticeable when looking at SST5: τ=90% and τ=80% are better options.

We see that there is a trade-off between $\mathcal{D}^j$'s quality and size from Figure 7 right. Being conservative, i.e., preserving only a handful of annotations from LMTurkers, results in a small, but high-quality $\mathcal{D}^j$; using all the annotations indiscriminately leads to a large $\mathcal{D}^j$ with low quality. This experiment highlights a key difference between LMTurk and AL: LMTurker annotations are not perfect and taking the annotation quality into consideration when training $\mathcal{S}$ is crucial.

6 Conclusion

In this work, our focus is the research question: *How to make effective use of current few-shot learners?* We propose LMTurk, a simple yet effective method that considers PLM-based few-shot learners as non-expert annotators in crowdsourcing; active learning strategies are incorporated to reduce the cost of annotation. We further show that processing the annotations from LMTurkers can be beneficial.

Future work may combine LMTurker annotations with human annotators in a human-in-the-loop setup (Monarch, 2021) to increase the overall utility of invested resources (Bai et al., 2021). Scaling up to even larger PLMs likely to further boost model performances (Kaplan et al., 2020; Brown et al., 2020) Applying LMTurk to multilingual few-shot learners (Zhao et al., 2021; Winata et al., 2021; Lin et al., 2021) is also promising.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and suggestions. MZ and HS were supported by the European Research Council (ERC# 740516) and the German Federal Ministry of Education and Research (BMBF, grant #01IS18036A).

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365–395.

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen H. Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Rabiu Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. MasakhaNER: Named Entity Recognition for African Languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131.

Omar Alonso, Daniel E. Rose, and Benjamin Stewart. 2008. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15.

Fan Bai, Alan Ritter, and Wei Xu. 2021. Pre-train or annotate? domain adaptation with a constrained budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages

5002–5015, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Sangnie Bhardwaj, Samarth Aggarwal, and Mausam Mausam. 2019. CaRB: A crowdsourced benchmark for open IE. 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 6262–6267, Hong Kong, China. Association for Computational Linguistics.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 286–295, Singapore. Association for Computational Linguistics.

Chris Callison-Burch and Mark Dredze, editors. 2010. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, Los Angeles.

David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine learning*, 15(2):201–221.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

Anca Dumitrache, Oana Inel, Benjamin Timmermans, Carlos Ortiz, Robert-Jan Sips, Lora Aroyo, and Chris Welty. 2021. Empirical methodology for crowdsourcing ground truth. *Semantic Web*, 12(3):1–19.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Richard M Felder and Rebecca Brent. 2009. Active learning: An introduction. *ASQ higher education brief*, 2(4):1–5.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 144–151, Ann Arbor, Michigan. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. 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 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Jeff Howe. 2008. *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House.

Jeff Howe et al. 2006. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Emily Jamison and Iryna Gurevych. 2015. Noise or additional information? leveraging crowdsource annotation item agreement for natural language tasks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 291–297, Lisbon, Portugal. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

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*.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.

Daniel Khashabi, Arman Cohan, Siamak Shakeri, Pedram Hosseini, Pouya Pezeshkpour, Malihe Alikhani, Moin Aminnaseri, Marzieh Bitaab, Faeze Brahman, Sarik Ghazarian, Mozhdeh Gheini, Arman Kabiri, Rabeeh Karimi Mahabagdi, Omid Memarrast, Ahmadreza Mosallanezhad, Erfan Noury, Shahab Raji, Mohammad Sadegh Rasooli, Sepideh Sadeghi, Erfan Sadeqi Azer, Niloofar Safi Samghabadi, Mahsa Shafaei, Saber Sheybani, Ali Tazarv, and Yadollah Yaghoobzadeh. 2021. ParsiNLU: A suite of language understanding challenges for Persian. *Transactions of the Association for Computational Linguistics*, 9:1147–1162.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active learning with Amazon Mechanical Turk. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

Ji-Ung Lee, Jan-Christoph Klie, and Iryna Gurevych. 2021. Annotation curricula to implicitly train non-expert annotators. *arXiv preprint arXiv:2106.02382*.

Ji-Ung Lee, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4233–4247, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Weixin Liang, James Zou, and Zhou Yu. 2020. ALICE: Active learning with contrastive natural language explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4380–4391, Online. Association for Computational Linguistics.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2021. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fei Mi, Yitong Li, Yasheng Wang, Xin Jiang, and Qun Liu. 2021a. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. *arXiv preprint arXiv:2109.04645*.

Fei Mi, Wanhao Zhou, Lingjing Kong, Fengyu Cai, Minlie Huang, and Boi Faltings. 2021b. Self-training improves pre-training for few-shot learning in task-oriented dialog systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1887–1898.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Robert Munro Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.

Nikita Nangia, Saku Sugawara, Harsh Trivedi, Alex Warstadt, Clara Vania, and Samuel R. Bowman. 2021. What ingredients make for an effective crowdsourcing protocol for difficult NLU data collection tasks? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1221–1235, Online. Association for Computational Linguistics.

Sungjoon Park, Jihyung Moon, Sung-Dong Kim, Won Ik Cho, Jiyoon Han, Jangwon Park, Chisung Song, Junseong Kim, Yongsook Song, Tae Hwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Young kuk Jeong, Inkwon Lee, Sang gyu Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin Shin, Seonghyun Kim, Lucy Park, Alice H. Oh, Jung-Woo Ha, and Kyunghyun Cho. 2021. KLUE: Korean language understanding evaluation. *ArXiv*, abs/2105.09680.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Silviu Paun and Dirk Hovy, editors. 2019. *Proceedings of the First Workshop on Aggregating and Analysing Crowdsourced Annotations for NLP*. Association for Computational Linguistics, Hong Kong, China.

Matthew E. 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.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 803–818, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

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.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2:441–448.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Christopher Schröder and Andreas Niekler. 2020. A survey of active learning for text classification using deep neural networks. *arXiv preprint arXiv:2008.07267*.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green AI. *Communications of the ACM*, 63(12):54–63.

Burr Settles. 2009. Active learning literature survey.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, Honolulu, Hawaii. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Aditya Siddhant and Zachary C. Lipton. 2018. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium. Association for Computational Linguistics.

Edwin Simpson and Iryna Gurevych. 2018. Finding convincing arguments using scalable Bayesian preference learning. *Transactions of the Association for Computational Linguistics*, 6:357–371.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Dietrich Trautmann, Johannes Daxenberger, Christian Stab, Hinrich Schütze, and Iryna Gurevych. 2020. Fine-grained argument unit recognition and classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9048–9056.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. In *Advances in Neural Information Processing Systems*.

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*, volume 32. Curran Associates, Inc.

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.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488, Copenhagen, Denmark. Association for Computational Linguistics.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. 2021. Language models are few-shot multilingual learners. *arXiv preprint arXiv:2109.07684*.

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.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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*, volume 32. Curran Associates, Inc.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239, Online. Association for Computational Linguistics.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2011. A survey of crowdsourcing systems. In *2011*

IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, pages 766–773.

Mike Zhang and Barbara Plank. 2021. Cartography active learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 395–406, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Ye Zhang, Matthew Lease, and Byron C. Wallace. 2017. Active discriminative text representation learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3386–3392. AAAI Press.

Mengjie Zhao, Philipp Dufter, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2020a. Quantifying the contextualization of word representations with semantic class probing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1219–1234, Online. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020b. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

Mengjie Zhao and Hinrich Schütze. 2021. Discrete and soft prompting for multilingual models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8547–8555, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mengjie Zhao, Yi Zhu, Ehsan Shareghi, Ivan Vulić, Roi Reichart, Anna Korhonen, and Hinrich Schütze. 2021. A closer look at few-shot crosslingual transfer: The choice of shots matters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5751–5767, Online. Association for Computational Linguistics.

A Reproducibility Checklist

A.1 Computing infrastructure

We use four Tesla V100 GPUs to prompt each of the LMTurkers, and a single Tesla V100 GPU is used when finetuning the small model $\mathcal{S}$.

A.2 Datasets

For SST2, CoLA, and RTE, we use the official datasets available on the benchmark website `gluebenchmark.com`. We download SST5 dataset from `nlp.stanford.edu/sentiment` and AGNews from the link provided by Zhang et al. (2015).

The number of testing examples of each dataset is shown in Table 2. Note that for SST2, CoLA, and RTE, $\mathcal{G}^{dev}$ is sampled from the training set, and the dev set is used as the test set.

CoLA	SST5	RTE	AGNews	SST2
1042	2210	277	7600	872

Table 2: Number of testing examples.

B Numerical Results

Table 3 reports the numerical value of Figure 2.

C Prompting Details

For each task, we list the five prompts employed to adapt a PLM to a LMTurker. "[v]" is a prompting token whose trainable embedding vector is randomly initialized.

For **SST5**, we use following prompts:

- "[v] **x** It is [MASK]."
- "[v] **x** Such a [MASK] movie."
- "**x** [v] It is pretty [MASK]."
- "It is [MASK] because **x** [v]"
- "**x** So it is [MASK]. [v]"

and the PLM picks a word from {"crap", "bad", "normal", "good", "perfect"}. to fill the position of "[MASK]". The mapping {"crap" $\rightarrow$ 1, "bad" $\rightarrow$ 2, "normal" $\rightarrow$ 3, "good" $\rightarrow$ 4, "perfect" $\rightarrow$ 5 } is used to convert model predictions to numerical values.

For **SST2**, we use following prompts:

- "[v] **x** It is [MASK]."
- "[v] **x** Such a [MASK] movie."
- "**x** [v] It is pretty [MASK]."
- "It is [MASK] because **x** [v]"
- "**x** So it is [MASK]. [v]"

and the PLM picks a word from {"bad", "good"} to fill the position of "[MASK]". The mapping {"bad" $\rightarrow$ 0, "good" $\rightarrow$ 1} is used.

For **AGNews**, we use following prompts:

- "[v] **x** It is about [MASK]."
- "**x** [v] Topic: [MASK]."
- "**x** [v] The text is about [MASK]."
- "**x** Topic: [MASK]. [v]"
- "**x** [v] [MASK]."

and the PLM picks a word from {"world", "sports", "economy", "technology"} to fill the position of "[MASK]". The mapping {"world" $\rightarrow$ 1, "sports" $\rightarrow$ 2, "economy" $\rightarrow$ 3, "technology" $\rightarrow$ 4 } is used.

For **CoLA**, we use following prompts:

- "[v] **x** It sounds [MASK]."
- "[v] **x** The sentence is [MASK]."
- "[v] **x** It is a [MASK] sentence."
- "**x** [v] [MASK]."
- "[v] **x** [MASK]."

and the PLM picks a word from {"wrong", "ok"} to fill the position of "[MASK]". The mapping {"wrong" $\rightarrow$ 0, "okay" $\rightarrow$ 1} is used.

For **RTE**, we use following prompts:

- "**p** Question: **h**? [v] Answer: [MASK]."
- "**p** [SEP] **h**? [MASK]. [v]"
- "**p** [SEP] **h**? [v] answer: [MASK]."
- "**p** [SEP] In short **h**. [MASK]. [v]"
- "[v] **p** [SEP] In short **h**. [MASK]."

where **p** and **h** refer to premise and hypothesis. The PLM picks a word from {"No", "Yes"} to fill the position of "[MASK]". The mapping {"No" $\rightarrow$ 0, "Yes" $\rightarrow$ 1} is used.

	$\mathcal{G}^8$		$\mathcal{G}^{16}$		$\mathcal{G}^{32}$	
	Workers	$\mathcal{S}$	Workers	$\mathcal{S}$	Workers	$\mathcal{S}$
SST2	91.13±0.52 91.63±0.68 90.18±1.00 90.83±0.58 90.52±1.84	67.63±8.01	91.93±1.09 93.08±0.62 91.74±1.04 90.79±0.47 91.67±1.36	75.83±1.35	91.97±0.83 91.70±1.78 91.21±1.83 91.13±0.24 93.23±0.37	76.37±3.16
SST5	41.37±1.55 42.32±2.04 40.57±2.70 37.69±1.34 38.05±2.60	28.47±1.61	45.16±2.13 45.96±2.12 46.70±0.93 42.53±2.43 42.96±0.69	34.97±1.51	45.91±0.96 48.64±0.59 50.53±0.94 43.32±3.42 45.72±1.43	33.47±2.79
RTE	68.95±1.47 54.99±3.76 62.70±1.33 50.42±2.07 51.99±4.45	57.30±1.79	68.35±2.29 57.64±3.23 70.88±1.70 58.60±1.62 57.88±2.83	61.50±0.78	71.72±1.96 58.48±3.59 68.47±1.19 59.33±4.72 60.41±2.47	62.93±0.74
AGNews	75.39±5.25 85.40±1.43 78.83±4.77 85.07±1.09 79.95±0.86	66.37±2.95	83.06±0.83 87.71±0.07 83.59±2.96 87.69±0.04 80.15±3.38	69.40±0.93	84.92±0.28 87.79±1.08 87.39±1.29 87.17±0.67 83.32±0.59	76.53±0.41
CoLA	0.14±1.43 2.42±4.84 7.40±8.12 9.91±7.98 15.33±2.15	0.97±4.40	11.81±7.82 15.23±7.07 19.71±1.89 17.14±2.48 19.66±0.48	4.27±3.26	19.88±3.30 22.51±0.96 26.34±1.54 18.15±0.63 27.58±7.09	2.50±2.41

Table 3: Few-shot performance of the five LMTurkers and the small model $\mathcal{S}$. Each experiment is repeated three times and we report mean and standard deviation.

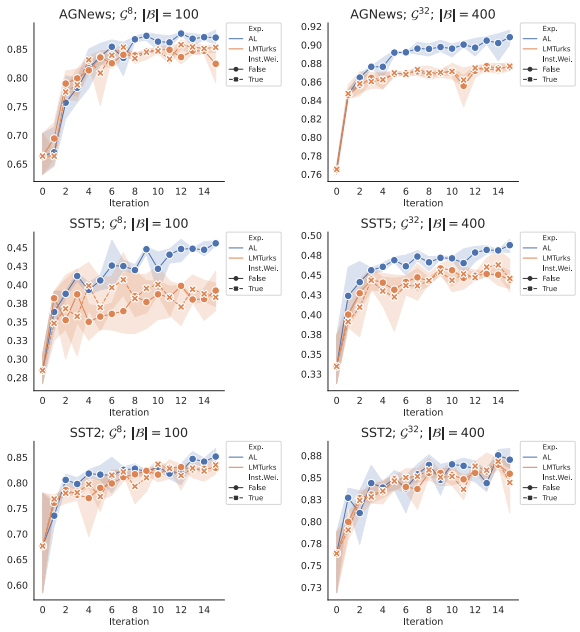


Figure 8: Weighting the training instances from LMTurkers.

D More Visualizations

Figure 9 visualizes the performance of $\mathcal{S}$ when different $|\mathcal{G}|$ and $|\mathcal{B}|$ are used.

E Instance Weighting

Following Wang et al. (2017), we associate each example $(\mathbf{x}, \hat{y}, \mathbf{l}) \in \mathcal{D}^j$ with weight 1-$entropy(\mathbf{l})$ when computing the loss during training $\mathcal{S}^j$. We can interpret this weight as a measure of the certainty of the LMTurkers ensemble.

Figure 8 reports the performance of $\mathcal{S}$ when using instance weighting, however, the impacts are less noticeable.

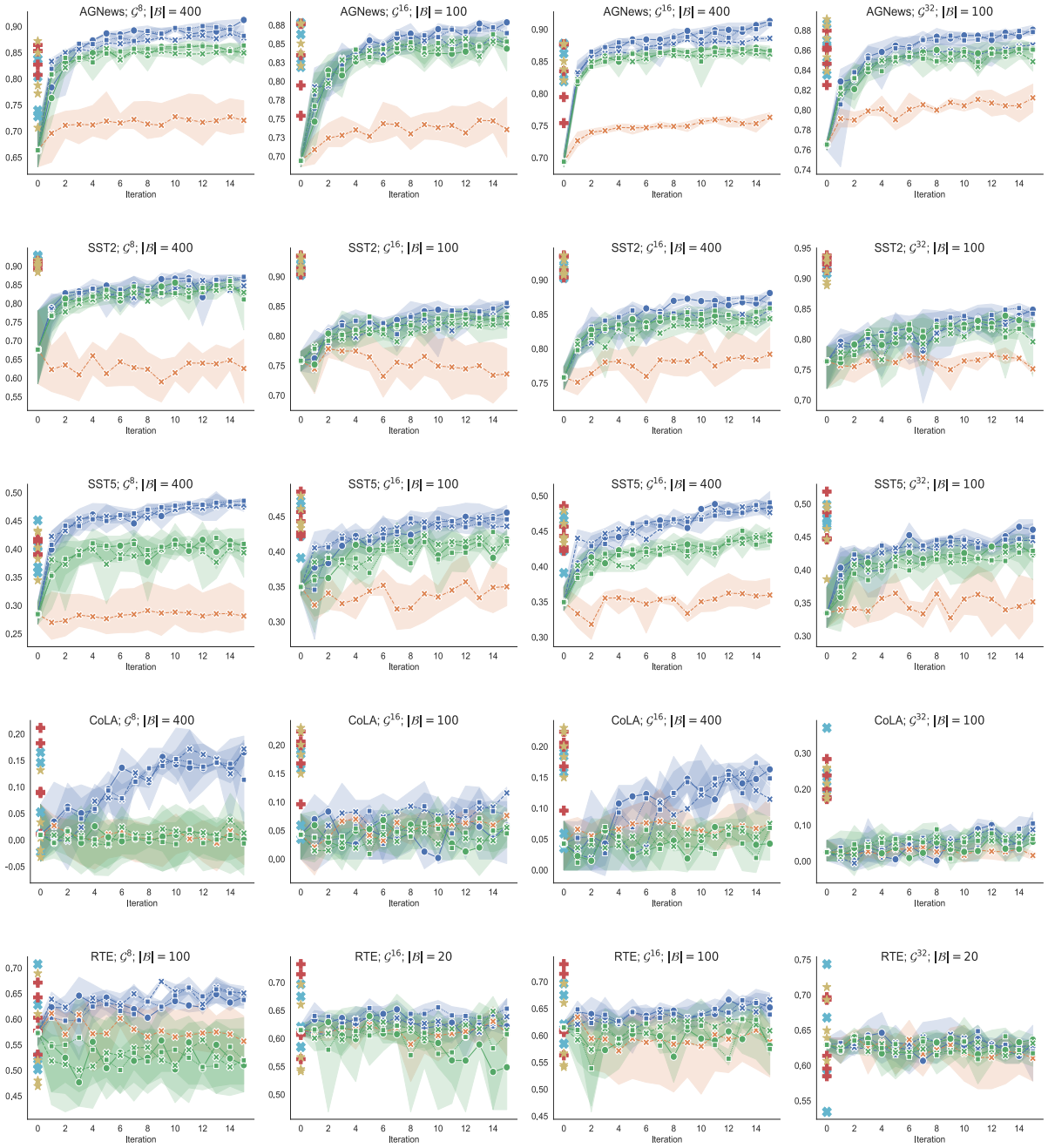


Figure 9: Improving $\mathcal{S}$ with active learning (blue), self training (orange), and LMTurk (green). Free markers at step zero show LMTurker performances; colors distinguish random seeds. Three acquisition functions are: Entropy (●), LeastConfident (■), random sampling (✖). At iteration j, each experiment is repeated three times; we show mean and standard deviation. We evaluate different $|\mathcal{G}|$ and $|\mathcal{B}|$.

Bibliography

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3159–3166.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 822–827, Baltimore, Maryland. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. Association for Computational Linguistics.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings

of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Alexey Bochkovskiy, Chien-Yao Wang, and H. Liao. 2020. Yolov4: Optimal speed and accuracy of object detection. *ArXiv*, abs/2004.10934.

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.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Kenneth Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011a. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011b. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018a. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 32:7059–7069.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

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.

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.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Philipp Dufter and Hinrich Schütze. 2020. Identifying elements essential for BERT's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4423–4437, Online. Association for Computational Linguistics.

Philipp Dufter, Mengjie Zhao, Martin Schmitt, Alexander Fraser, and Hinrich Schütze. 2018a. Embedding learning through multilingual concept induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1520–1530, Melbourne, Australia. Association for Computational Linguistics.

Philipp Dufter, Mengjie Zhao, and Hinrich Schütze. 2018b. Multilingual embeddings jointly induced from contexts and concepts: Simple, strong and scalable. *arXiv preprint arXiv:1811.00586*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of*

the European Chapter of the Association for Computational Linguistics, pages 462–471, Gothenburg, Sweden. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. 2021. Pre-trained models: Past, present and future. *AI Open*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

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.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. 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 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Geoffrey E Hinton. 1984. Distributed representations.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual bert: An empirical study. In *International Conference on Learning Representations*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on*

Machine Learning, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1188–1196, Bejing, China. PMLR.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4470–4481, Brussels, Belgium. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27:2177–2185.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 765–774, Valencia, Spain. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel bible corpus. *Oceania*, 135(273):40.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6297–6308, Red Hook, NY, USA. Curran Associates Inc.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. 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. 2013c. 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, Atlanta, Georgia. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

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, Doha, Qatar. Association for Computational Linguistics.

Matthew E. 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.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. 2019. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019a. Language models are unsupervised multitask learners.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

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.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Hinrich Schütze. 1992. Word space. In *Advances in neural information processing systems*, pages 895–902.

Hinrich Schütze. 2017. Nonsymbolic text representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 785–796, Valencia, Spain. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63(12):54–63.

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.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

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.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas. Association for Computational Linguistics.

Wikipedia contributors. 2021. Recurrent neural network. [Online; accessed 22-December-2021].

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. 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*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *arXiv preprint arXiv:2105.13626*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246, Berlin, Germany. Association for Computational Linguistics.

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*, volume 32. Curran Associates, Inc.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239, Online. Association for Computational Linguistics.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

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 *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.