Efficient Few-Shot Learning Without Prompts

Lewis Tunstall[1], Nils Reimers[2], Unso Eun Seo Jo[1], Luke Bates[3],
Daniel Korat[4], Moshe Wasserblat[4], Oren Pereg[4]

[1]Hugging Face   [2]cohere.ai

[3]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt

[4]Emergent AI Lab, Intel Labs

[1]`firstname@huggingface.com` [2]`info@nils-reimers.de`

[3]`bates@ukp.informatik.tu-darmstadt.de`

[4]`firstname.lastname@intel.com`

Abstract

Recent few-shot methods, such as parameter-efficient fine-tuning (PEFT) and pattern exploiting training (PET), have achieved impressive results in label-scarce settings. However, they are difficult to employ since they are subject to high variability from manually crafted prompts, and typically require billion-parameter language models to achieve high accuracy. To address these shortcomings, we propose SETFIT (**Se**ntence **T**ransformer **Fi**ne-**t**uning), an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers (ST). SETFIT works by first fine-tuning a pretrained ST on a small number of text pairs, in a contrastive Siamese manner. The resulting model is then used to generate rich text embeddings, which are used to train a classification head. This simple framework requires no prompts or verbalizers, and achieves high accuracy with orders of magnitude less parameters than existing techniques. Our experiments show that SETFIT obtains comparable results with PEFT and PET techniques, while being an order of magnitude faster to train. We also show that SETFIT can be applied in multilingual settings by simply switching the ST body. Our code[1] and datasets[2] are made publicly available.

1 Introduction

Few-shot learning methods have emerged as an attractive solution to label-scarce scenarios, where data annotation can be time-consuming and costly. These methods are designed to work with a small number of labeled training examples, and typically involve adapting pretrained language models (PLMs) for specific downstream tasks.

Today, there exist several approaches to few-shot learning with PLMs. These include in-context learning (ICL), parameter-efficient fine-tuning (PEFT), and pattern exploiting training
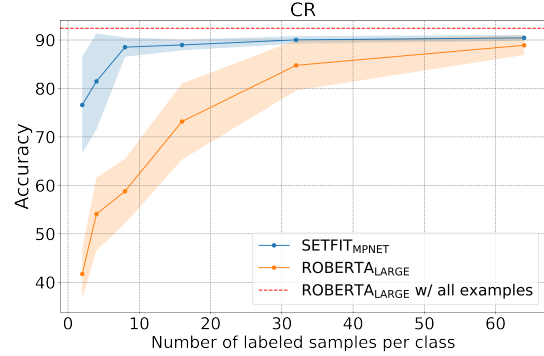

Figure 1: Compared to standard fine-tuning, SETFIT is more sample efficient and exhibits less variability when trained on a small number of labeled examples.

(PET). Unfortunately, these approaches can be impractical for many researchers and practitioners. One disadvantage is that these approaches typically rely on the use of large-scale language models to achieve high performance. For example, T-FEW (Liu et al., 2022) is based on the 11 billion parameter model T0 (Sanh et al., 2021), while GPT-3 (Brown et al., 2020a) is an order of magnitude larger. Secondly, training and deploying these few-shot methods typically requires specialized infrastructure with limited accessibility. Moreover, PET and the prominent PEFT methods require, as part of their training, the input of manually generated prompts, yielding varying outcomes depending on the level of manual prompt-engineering.

In this paper, we propose SETFIT, an approach based on Sentence Transformers (ST) (Reimers and Gurevych, 2019) that dispenses with prompts altogether and does not require large-scale PLMs to achieve high accuracy. For example, with only 8 labeled examples in the Customer Reviews (CR) sentiment dataset, SETFIT is competitive with fine-tuning on the full training set, despite the fine-tuned model being three times larger (see Figure 1).

We demonstrate SETFIT's efficacy in few-shot text classification over a range of NLP datasets

---
[1]https://github.com/huggingface/setfit
[2]https://huggingface.co/setfit

and in multiple scenarios including distillation and non-English data. We compare our method to standard PLM fine-tuning, state-of-the-art PET- and PEFT-based methods such as ADAPET (Tam et al., 2021) and T-FEW (Liu et al., 2022), as well as recent prompt-free techniques such as PERFECT (Karimi Mahabadi et al., 2022a).

We summarize our contributions as follows:

1. We propose SETFIT– a simple and prompt-free method – and provide a comprehensive guide for applying it in practical few-shot settings.

2. We evaluate SETFIT's performance on a number of few-shot text classifications tasks and show that it outperforms the state-of-the-art prompt-free method and ranks alongside much larger prompt-based, few-shot models.

3. We make the code and data used in our work publicly available.

2 Related Work

SETFIT engages with two related lines of literature. We first extend the small but significant body of work on Sentence Transformers (ST) for text classification. Perone et al. (2018) introduced the idea of using sentence embeddings for text classification. Piao (2021) used 'out-of-the-box' STs for text classification without fine-tuning them. SETFIT differs from these works in two aspects: First, we fine-tune the ST in a Siamese manner for a text classification objective showing that it significantly enhances performance; second, we demonstrate this approach in few-shot setups.

SETFIT is also related to the recently emerging few-shot and zero-shot training line of literature as few-shot and zero-shot approaches have recently received a great deal of interest in the research community due to the availability of pretrained language models and the untapped capacity to use them in resource-constrained domains. Specifically, we discuss ICL, PEFT, and prompt-based fine-tuning.

ICL models directly generate predictions based on input-to-output training examples provided as prompts, without any parameter updates. Perhaps the best known example is GPT-3 (Brown et al., 2020b), which achieves remarkable few-shot performance. However, GPT-3 contains 175 billion parameters and requires massive computational resources, prompt engineering, and can only utilize pretrained knowledge.

PEFT methods, such as adapters (Rebuffi et al., 2017), hold the majority of parameters fixed during training and only update small feed-forward networks that are inserted within the larger model architecture. A recent example is T-FEW (Liu et al., 2022), which outperforms GPT-3 at much lower computational cost. It accomplishes this by adding learned vectors that rescale the network's internal activations. T-FEW is 16 times smaller than GPT-3, but is still too large to be utilized as a practical tool in industry. It also requires a set of handcrafted prompts for each dataset.

Another alternative to ICL is prompt-based fine-tuning. This approach converts the downstream classification task into a masked-language modeling (MLM) objective. The model outputs tokens in a cloze-style format that maps to the corresponding labels via a predefined template. A well known example of this method is Pattern Exploiting Training (PET) (Schick and Schütze, 2021b,a) . Like GPT-3, PET relies on manually-crafted prompts, but since the model can be fine-tuned to specific tasks, PET-based approaches typically outperform GPT-3 in few-shot scenarios, even with far smaller PLM backbones. PET has since been extended in two main directions: ADAPET (Tam et al., 2021), which improves PET with a decoupled label objective and label-conditioned MLM objective, and PERFECT (Karimi Mahabadi et al., 2022b) which uses task-specific adapters (Houlsby et al., 2019; Pfeiffer et al., 2021) and multi-token label-embeddings eliminate task prompts and verbalizers.

3 SetFit: Sentence Transformer Fine-Tuning

SETFIT is based on Sentence Transformers (Reimers and Gurevych, 2019) which are modifications of pretrained transformer models that use Siamese and triplet network structures to derive semantically meaningful sentence embeddings. The goal of these models is to minimize the distance between pairs of semantically similar sentences and maximize the distance between sentence pairs that are semantically distant. Standard STs output a fixed, dense vector that is meant to represent textual data and can then be used by machine learning algorithms.
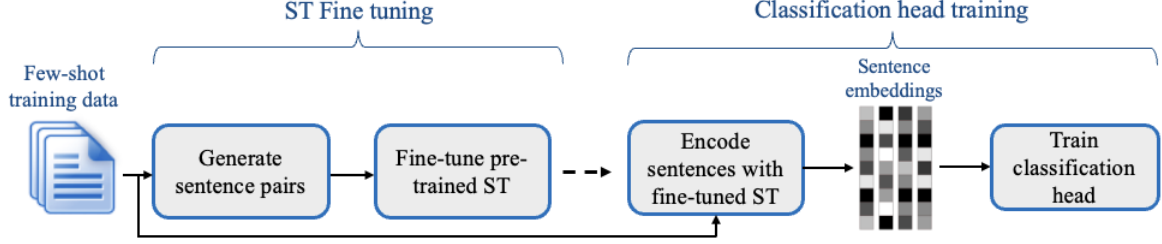
Figure 2: SETFIT 's fine-tuning and training block diagram.

3.1 The SETFIT approach for few-shot text classification

SETFIT uses a two-step training approach in which we first fine-tune an ST and then train a classifier head. In the first step, an ST is fine-tuned on the input data in a contrastive, Siamese manner on sentence pairs. In the second step, a text classification head is trained using the encoded training data generated by the fine-tuned ST from the first step. Figure 2 illustrates this process, and we discuss these two steps in the following sections.

ST fine-tuning To better handle the limited amount of labeled training data in few-shot scenarios, we adopt a contrastive training approach that is often used for image similarity (Koch et al., 2015). Formally, given a small set of K labeled examples $D = \{(x_i, y_i)\}$, where x_i and y_i are sentences and their class labels, respectively. For each class label $c \in C$, we generate a set of R positive triplets $T_p^c = \{(x_i, x_j, 1)\}$, where x_i and x_j are pairs of randomly chosen sentences from the same class c such that $(y_i = y_j = c)$. Similarly, we also generate a set of R negative triplets $T_n^c = \{(x_i, x_j, 0)\}$, where x_i are sentences from class c and x_j are randomly chosen sentences from different classes such that $(y_i = c, \ y_j \neq c)$. Finally, the contrastive fine tuning data set T is produced by concatenating the positive and negative triplets across all class labels; $T = \{(T_p^0, T_n^0), (T_p^1, T_n^1), ..., (T_p^{|C|}, T_n^{|C|})\}$, where $|C|$ is the number of class labels, $|T| = 2R|C|$ is the number of pairs in T and R is a hyperparameter. Unless stated otherwise, we used $R = 20$ in all the evaluations.

This contrastive fine-tuning approach enlarges the size of training data in few-shot scenarios. Assuming that a small number (K) of labeled examples are given for a binary classification task, the potential size of the ST fine-tuning set T is derived from the number of unique sentence pairs that can be generated, namely $K(K-1)/2$, which is sig-

nificantly larger than just K.

Classification head training In this second step, the fine-tuned ST encodes the original labeled training data $\{x_i\}$, yielding a single sentence embedding per training sample; $Emb^{x_i} = ST(x_i)$ where $ST()$ is the function representing the fine-tuned ST. The embeddings, along with their class labels, constitute the training set for the classification head $T^{CH} = \{(Emb^{x_i}, y_i)\}$ where $|T^{CH}| = |D|$. A logistic regression model is used as the text classification head throughout this work.

Inference At inference time, the fine-tuned ST encodes an unseen input sentence (x_i) and produces a sentence embedding. Next, the classification head that was trained in the training step, produces the class prediction of the input sentence based on its sentence embedding. Formally this is $x_i^{pred} = CH(ST(x_i))$, where CH represents the classification head prediction function.

4 Experiments

4.1 Data

We conduct experiments on available text classification datasets. We split the datasets into development and test datasets (See Table 6 in Appendix A). The development datasets are utilized for setting SETFIT's hyperparameters such as the number of training pairs ($|T|$), the loss function and the optimal number of training epochs. In order to test the robustness of SETFIT to various types of text, we choose test datasets that represent different text classification tasks with a varying number of classes. All datasets used are available on the Hugging Face Hub under the SETFIT organisation.[3] In addition we evaluate SETFIT on the RAFT benchmark (Alex et al., 2021), a real-world few-shot text-classification benchmark composed of 11 practical tasks, where each task has only 50 training

---

[3]huggingface.co/SetFit

examples.

4.2 SETFIT models

We evaluate three variations of SETFIT each one uses different underlying model of different size (Shown in Table 1)

Variation	Underlying ST Model	Size*
SETFIT$_{RoBERTa}$	all-roberta-large-v1[4]	355M
SETFIT$_{MPNet}$	paraphrase-mpnet-base-v2[5]	110M
SETFIT$_{MiniLM}$	paraphrase-MiniLM-L3-v2[6]	15M

Table 1: SETFIT model variations using three different underlying ST models. *Number of parameters.

4.3 Baselines

We compare SETFIT's performance against standard transformer fine-tuning and recent best-performing few-shot approaches: ADAPET (Tam et al., 2021), PERFECT (Karimi Mahabadi et al., 2022b), and T-FEW (Liu et al., 2022).

Standard fine-tuning Our first baseline is ROBERTA$_{LARGE}$ (Liu et al., 2019), a standard, encoder-only transformer that is fine-tuned for sequence classification. Since we assume no validation sets, we constructed validation splits by randomly selecting equally sized portions from the train split. We perform a hyperparameter search on the number of epochs in the range [25,75] and pick the best performing model on a validation split.

We use a learning rate of $2e^{-5}$ and batch size of 4 in all our experiments.

ADAPET Pattern exploiting training (PET) (Schick and Schütze, 2021b,a) is a method for improving PLM performance in few-shot setups on downstream tasks by converting textual input into a cloze-style question intended to be reminiscent of the masked language modelling (MLM) objective under which large PLMs such as BERT (Devlin et al., 2019) are trained. To determine SETFIT's performance relative to PET-based approaches, we compare our method to ADAPET (Tam et al., 2021), an extension of PET. In recent work

[4] https://huggingface.co/sentence-transformers/all-roberta-large-v1
[5] https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2
[6] https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L3-v2

(Schick and Schütze, 2021), the authors show that PET-based classification methods excel on the RAFT benchmark, placing second only to much larger models such as T-FEW. In our experiments, we used ADAPET with default hyperparameters and examined its performance with different PLM backbones, reporting the PLM which resulted in the best performance, albert-xxlarge-v2 [7] (see Appendix A.2 in the Appendix for further details).

PERFECT PERFECT (Karimi Mahabadi et al., 2022b) is another cloze-based fine-tuning method, but unlike PET or ADAPET, it does not require handcrafted task prompts and verbalizers. Instead, PERFECT uses task-specific adapters (Houlsby et al., 2019; Pfeiffer et al., 2021) and multi-token label-embeddings which are independent from the language model vocabulary during fine-tuning. To run PERFECT on our test datasets, we adapted the configurations provided in the PERFECT codebase.

T-FEW T-FEW (Liu et al., 2022) is a PEFT-based few-shot learning method based on T0 (Sanh et al., 2021). The authors provide two versions of T-FEW: 11 and 3 billion parameters. Due to compute constraints, we were unable to run the 11 billion version, which requires an 80GB A100 GPU. Running tests on T-FEW as opposed to SETFIT posed several hurdles. First, because T-FEW's performance varies significantly depending on the input prompts, we run each experiment using 5 random seeds, and report the median result, as in the original paper. Second, T-FEW relies on dataset-specific prompts, made available on P3 (Public Pool of Prompts) (Bach et al., 2022). Only one of our test datasets had prompts in P3. For the rest of the datasets, we adapt standardized P3 prompts of similar tasks or implement prompts ourselves (See Appendix A.3).

4.4 Experimental Setup

Systematically evaluating few-shot performance can be challenging, because fine-tuning on small datasets may incur instability (Dodge et al., 2020; Zhang et al., 2021). To address this issue, in our experiments we use 10 random training splits for each dataset and sample size. These splits are used as training data across all tested methods. For each method, we report the average measure (depending on the dataset) and the standard deviation across these splits. We fine-tune SETFIT's ST model using

[7] https://huggingface.co/albert-xxlarge-v2

cosine-similarity loss with a learning rate of $1e^{-3}$, a batch size of 16 and a maximum sequence length of 256 tokens, for 1 epoch.

5   Results

Table 2 shows a comparison between SETFIT$_{\text{MPNET}}$ and the baselines for $N = 8$ and $N = 64$ labeled training samples per class. For reference purposes, standard fine-tuning results using the full training data are also shown (in all cases higher scores indicates stronger performance; see Table 6 in Appendix A for dataset metric details). We find that SETFIT$_{\text{MPNET}}$ significantly outperforms the FINETUNE baseline for $N = 8$ by an average of 19.3 points. However, as the number of training samples increases to $N = 64$, the gap decreases to 5.6 points.

Similarly, we find that SETFIT$_{\text{MPNET}}$ outperforms PERFECT by 13.6 and 2.6 points. SETFIT$_{\text{MPNET}}$ also outperforms ADAPET by 4.0 and 1.5 points for $N = 8$ and $N = 64$ respectively. For $N = 8$, SETFIT$_{\text{MPNET}}$ is on par with T-FEW 3B whereas for $N = 64$ SETFIT$_{\text{MPNET}}$ outperforms T-FEW 3B by 5 points on average, despite being prompt-free and more than 27 times smaller.

RAFT results   The test datasets listed in Table 2 were not specifically designed for few-shot benchmarking. In order to better benchmark SETFIT, we used the RAFT benchmark (Alex et al., 2021) which is specifically designed for benchmarking few-shot methods. Table 3 shows the average accuracy of SETFIT$_{\text{MPNET}}$ and SETFIT$_{\text{ROBERTA}}$ and four prominent methods. SETFIT$_{\text{ROBERTA}}$ outperforms GPT3 and PET by 8.6 and 1.7 points respectively while alleviating the need for hand crafting prompts. It also surpasses the human baseline in 7 out of 11 tasks. SETFIT$_{\text{ROBERTA}}$ falls short of T-FEW 11B by 4.5 points. however, SETFIT$_{\text{ROBERTA}}$ is more than 30 times smaller than T-FEW 11B, does not require manual prompt crafting and is much more efficient in training and inference (see Table 5).

6   Multilingual Experiments

To determine SETFIT's performance in a multilingual, few-shot text classification scenario, we conducted development and test experiments on multilingual datasets and compared SETFIT to standard transformer fine-tuning and ADAPET. To the best of our knowledge, this is the first work to examine ADAPET on non-English data (see Appendix A for details).

Experimental Setup   For the multilingual experiments, we use the Multilingual Amazon Reviews Corpus (MARC) (Keung et al., 2020). This dataset consists of Amazon reviews in six languages (English, Japanese, German, French, Spanish, and Chinese), where each review is labeled according to a 5-star rating scale. We chose this corpus for its typological diversity in order to examine the generalizability of SETFIT and other methods across a variety of languages.

For the SETFIT underlying model, we use paraphrase-multilingual-mpnet-base-v2,[8] which is a multilingual version of paraphrase-mpnet-base-v2 that is trained on parallel data in over 50 languages.

For the FINETUNE and ADAPET baselines, we use XLM-ROBERTA$_{\text{BASE}}$ (Conneau et al., 2019),[9] which has a similar size to the SETFIT model. We compare the performance of each method using the same settings as (Conneau et al., 2019):

- **each:** Train and evaluate on monolingual data to measure per-language performance.

- **en:** Train on the English training data and then evaluate on each language's test set.

- **all:** Train on all the training data and evaluate on each language's test set.

Method   For SETFIT standard fine-tuning, and ADAPET, we adopt the same methodology and hyperparameters used for the monolingual English experiments in 4. We evaluate each method in the few-shot regime ($N = 8$ samples per class) and compare against performance of fine-tuning on the full training set of 20,000 examples.

Results   Table 4 shows the results of SETFIT standard fine-tuning, and ADAPET on each language in MARC, where a higher MAE indicates weaker performance. In the few-shot regime of $N = 8$ samples per class, we find that SETFIT significantly outperforms FINETUNE and ADAPET in all settings (each, en, all), with the best average performance obtained when training on English data only.

---

[8]huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
[9]huggingface.co/xlm-roberta-base

Method	SST-5	AmazonCF	CR	Emotion	EnronSpam	AGNews	Average†
			$\lvert N \rvert = 8^*$				
FINETUNE	$33.5_{2.1}$	$9.2_{4.9}$	$58.8_{6.3}$	$28.7_{6.8}$	$85.0_{6.0}$	$81.7_{3.8}$	$43.0_{5.2}$
PERFECT	$34.9_{3.1}$	$18.1_{5.3}$	$81.5_{8.6}$	$29.8_{5.7}$	$79.3_{7.4}$	$80.8_{5.0}$	$48.7_{6.0}$
ADAPET	$50.0_{1.9}$	$19.4_{7.3}$	$91.0_{1.3}$	$46.2_{3.7}$	$85.1_{3.7}$	$\mathbf{85.1}_{2.7}$	$58.3_{3.6}$
T-FEW 3B	$\mathbf{55.0}^{\star}_{1.4}$	$19.0_{3.9}$	$\mathbf{92.1}_{1.0}$	$\mathbf{57.4}_{1.8}$	$\mathbf{93.1}_{1.6}$	–	$\mathbf{63.4}_{1.9}$
SETFIT$_{\text{MPNET}}$	$43.6_{3.0}$	$\mathbf{40.3}_{11.8}$	$88.5_{1.9}$	$48.8_{4.5}$	$90.1_{3.4}$	$82.9_{2.8}$	$62.3_{4.9}$
			$\lvert N \rvert = 64^*$				
FINETUNE	$45.9_{6.9}$	$52.8_{12.1}$	$88.9_{1.9}$	$65.0_{17.2}$	$95.9_{0.8}$	$88.4_{0.9}$	$69.7_{7.8}$
PERFECT	$49.1_{0.7}$	$\mathbf{65.1}_{5.2}$	$92.2_{0.5}$	$61.7_{2.7}$	$95.4_{1.1}$	$\mathbf{89.0}_{0.3}$	$72.7_{1.9}$
ADAPET	$54.1_{0.8}$	$54.1_{6.4}$	$92.6_{0.7}$	$72.0_{2.2}$	$96.0_{0.9}$	$88.0_{0.6}$	$73.8_{2.2}$
T-FEW 3B	$\mathbf{56.0}_{0.6}$	$34.7_{4.5}$	$\mathbf{93.1}_{1.0}$	$70.9_{1.1}$	$\mathbf{97.0}_{0.3}$	–	$70.3_{1.5}$
SETFIT$_{\text{MPNET}}$	$51.9_{0.6}$	$61.9_{2.9}$	$90.4_{0.6}$	$\mathbf{76.2}_{1.3}$	$96.1_{0.8}$	$88.0_{0.7}$	$\mathbf{75.3}_{1.3}$
			$\lvert N \rvert = Full^{**}$				
FINETUNE	59.8	80.1	92.4	92.6	99.0	93.8	84.8

Table 2: SETFIT performance score and standard deviation compared to the baselines across 6 test datasets for three training set sizes $\lvert N \rvert$. *Number of training samples per class. **Entire available training data used. †The AGNews dataset is excluded from the average score to enable fair comparison with T-FEW (which has AGNews in its training set). ⋆The inputs of SST-5 (but not its labels) appeared in T-FEW's training set, as part of Rotten Tomatoes dataset.

Rank	Method	Score	Size*
1	YIWISE	76.8	-
2	T-FEW 11B	75.8	11B
4	Human baseline	73.5	-
6	SETFIT$_{\text{ROBERTA}}$	71.3	355M
9	PET	69.6	235M
11	SETFIT$_{\text{MPNET}}$	66.9	110M
12	GPT-3	62.7	175B

Table 3: SETFIT compared to prominent methods on the RAFT leaderboard (as of Sept. 5, 2022). *Number of parameters.

7 SETFIT Model Efficiency

7.1 Few-shot distillation

We have shown that SETFIT achieves state-of-the-art results in few-shot setups using underlying base models such as paraphrase-mpnet-base-v2 and ROBERTA$_{\text{LARGE}}$, containing 110M parameters and 355M parameters respectively; but in real-world deployments, where cost and sustainability are prioritized, the use on even more efficient models is desirable. Previous works have shown model distillation to be effective in reducing computational load while preserving much of the original model's performance (Ba and Caruana, 2014; Hinton et al., 2015). In this section we evaluate the performance of SETFIT as a student model compared to a standard transformer student model in few-shot distillation setups when the amount of unlabeled training data is limited.

Experimental Setup For the distillation tests we use the datasets AGNews, Emotion and SST-5 described in Appendix A.1. For the SETFIT teacher we chose SETFIT$_{\text{MPNET}}$, which contains 110M parameters, whereas for the SETFIT student we chose SETFIT$_{\text{MINILM}}$, which is a much smaller model (15M parameters). For fair comparison, we use as the baseline student MiniLM-L3-H384-uncased[10], a standard transformer encoder of the same size as our SETFIT student model. For each of the three datasets we train the SETFIT teacher model using only 16 labeled samples per class, and the student models are trained using the same 16 labeled samples per class together with various amounts of additional unlabeled data. We follow the same data-split policy and SETFIT training parameters' settings described in Section 4.4.

Method The SETFIT student is trained using sentence pairs and the level of similarity between each pair as input. The similarity is generated by using the underlying ST of the teacher to produce sen-

[10] huggingface.co/nreimers/
MiniLM-L3-H384-uncased

Method	Train	En	De	Ja	Zh	Fr	Es	Average
				$\lvert N\rvert = 8^*$				
FINETUNE	each	$122.9_{14.0}$	$119.9_{13.6}$	$120.5_{8.0}$	$128.6_{10.7}$	$123.2_{13.0}$	$116.3_{8.3}$	$121.9_{11.3}$
	en	$115.9_{11.3}$	$115.2_{12.0}$	$121.6_{12.3}$	$123.0_{8.8}$	$117.3_{13.0}$	$113.1_{12.4}$	$117.7_{11.6}$
	all	$117.8_{4.9}$	$116.3_{9.7}$	$121.5_{12.4}$	$120.5_{6.7}$	$117.3_{9.9}$	$110.1_{9.5}$	$117.2_{8.8}$
ADAPET	each	$129.9_{13.6}$	$136.4_{10.6}$	$130.4_{13.4}$	$135.0_{10.9}$	$141.8_{10.1}$	$136.0_{10.4}$	$134.9_{11.5}$
	en	$138.9_{17.8}$	$151.5_{17.8}$	$160.8_{16.7}$	$158.8_{16.3}$	$152.0_{15.7}$	$149.8_{17.1}$	$152.0_{16.9}$
	all	$150.8_{12.0}$	$136.2_{7.0}$	$150.8_{10.0}$	$152.8_{10.2}$	$140.0_{14.0}$	$145.1_{4.5}$	$146.0_{11.3}$
SETFIT	each	$82.9_{4.3}$	$\mathbf{80.0}_{2.4}$	$95.5_{2.8}$	$95.3_{2.8}$	$85.3_{6.0}$	$\mathbf{80.8}_{5.4}$	$86.6_{4.9}$
	en	$\mathbf{82.6}_{4.8}$	$83.4_{5.9}$	$\mathbf{93.2}_{6.6}$	$\mathbf{93.9}_{3.6}$	$\mathbf{82.2}_{4.8}$	$83.4_{5.9}$	$\mathbf{86.4}_{5.2}$
	all	$83.0_{5.3}$	$84.0_{7.6}$	$97.1_{9.2}$	$97.4_{6.5}$	$83.5_{6.5}$	$84.9_{6.1}$	$88.3_{6.9}$
				$\lvert N\rvert = Full^{**}$				
FINETUNE	each	46.2	$\mathbf{43.7}$	$\mathbf{46.8}$	$\mathbf{56.6}$	$\mathbf{47.8}$	$\mathbf{45.3}$	$\mathbf{47.7}$
	en	$\mathbf{46.1}$	46.6	61.0	69.4	55.6	52.9	55.3
	all	46.6	49.4	61.0	69.4	55.6	55.0	56.2

Table 4: Average performance (MAE $\times$ 100) on the Multilingual Amazon Reviews Corpus for two training set sizes $\lvert N\rvert$. * No. of training samples per class. **Entire available training data used (20,000 samples).
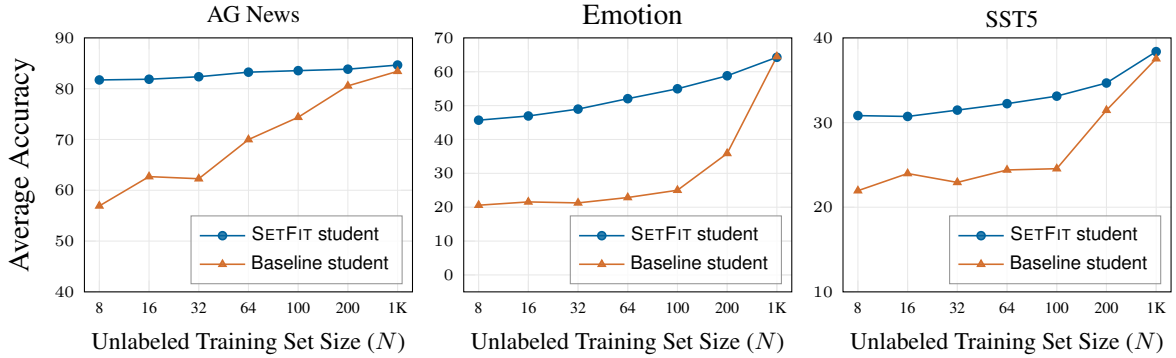

Figure 3: Average accuracy as a function of the unlabeled training set size N of the SETFIT student and the baseline student on AG News, Emotion and SST5 datasets.

tence embeddings for each pair and to calculate the cosine-similarity between them. The underlying ST of the SETFIT student is trained to mimic the ST of the teacher output by minimizing the error between the SETFIT teacher-produced cosine-similarity and its output. The classification head of the student is then trained using the embeddings produced by the student's ST and the logits produced by the SETFIT teacher classification head. The baseline student is trained to mimic the teacher output by minimizing the error between the logits produced by the SETFIT teacher classification head and its output.

Results Figure 3 shows a comparison between the SETFIT student model and the baseline student model for various amounts of unlabeled training data (N). The SETFIT student significantly outperforms the baseline student when only small amounts of unlabeled data are available. For example, for $N = 8$, the SETFIT student outperforms the baseline student by **24.8**, **25.1**, and **8.9** average accuracy on the AGNews, Emotion and SST5 datasets, respectively. As N increases, the performance gains decrease and are on par for $N = 1K$.

7.2 Computational costs

Comparing the relative computational costs of SET-FIT versus PET and PEFT methods isn't straightforward since each method typically has different hardware and memory requirements.

To simplify the comparison, we follow the approach adopted by Liu et al. (2022) and use FLOPs-per-token estimates to compare SETFIT to T-FEW. These estimates can be obtained from Kaplan et al.

(2020), who show that encoder-only models with N parameters have approximately $2N$ FLOPs-per-token for inference and $6N$ FLOPs-per-token for training. The resulting cost for inference and training is then given by:

$$\mathcal{C}_{\text{inf}} = 2N \cdot \ell_{\text{seq}} ,$$
$$\mathcal{C}_{\text{train}} = 6N \cdot \ell_{\text{seq}} \cdot n_{\text{steps}} \cdot n_{\text{batch}} ,$$

where ℓ_{seq} is the input sequence length, n_{steps} is the number of training steps, and n_{batch} is the batch size. For encoder-decoder models like T-FEW, these estimates are halved, since the model only processes each token with either the encoder or decoder.

For the inference and training estimates shown in Table 5, we use $\ell_{\text{seq}} = 38$ and $\ell_{\text{seq}} = 54$ as the input sequence length for SETFIT$_{\text{MPNET}}$ (T-FEW); this is the median number of tokens across all the test datasets in Table 2. We also use $n_{\text{steps}} = 1000$ and $n_{\text{batch}} = 8$ for all training estimates. As shown in the table, the SETFIT$_{\text{MPNET}}$ model is approximately an order of magnitude faster at inference and training than T-FEW, despite having comparable performance on the test datasets of Table 2. SETFIT$_{\text{MINILM}}$ is two orders of magnitude faster than T-FEW, with an average score reduction of 3.1 accuracy points. Moreover, the storage cost of the SETFIT models (70MB and 420MB respectively) is 163 to 26 times smaller than the T0-3B checkpoint used by T-FEW 3B (11.4GB), making these models much better suited for real-world deployment.

These estimates are borne out by comparing the time needed to train each method to convergence on $N = 8$ examples. For our datasets, SETFIT$_{\text{MPNET}}$ takes approximately 30 seconds to train on a p3.2xlarge AWS instance (16GB GPU memory), at a cost of $0.025 per split. On the other hand, T-FEW 3B requires at least 40GB GPU memory, and training on a p4d.24xlarge AWS instance takes approximately 700 seconds, at a cost of $0.7 per split.

8 Conclusion

This paper introduces SETFIT, a new few-shot text classification approach. We show that SETFIT has several advantages over comparable approaches such as T-FEW, ADAPET and PERFECT. In particular, SETFIT is much faster at inference and training; SETFIT requires much smaller base models to be performant, not requiring external compute; and

Method	Inf. FLOPs	Train FLOPs	Speed-up	Score
T-FEW 3B	1.6e11	3.9e15	1x	$63.4_{1.9}$
SETFIT$_{\text{MPNET}}$	8.3e9	2.0e14	19x	$62.3_{4.9}$
SETFIT$_{\text{MINILM}}$[†]	1.3e9	3.2e13	123x	$60.3_{1.6}$

Table 5: Relative computational cost and average scores of SETFIT and T-FEW using $|N| = 8$ on the test datasets listed in Table 2. [†]Trained in the distillation setup as described in Section 7.1, using $|N| = 8$ for teacher training and the rest of the available training data as unlabeled student training data. For fixed n_{steps} and n_{batch}, the relative speed-up $(N' \cdot \ell'_{\text{seq}})/(2N \cdot \ell_{\text{seq}})$ is the same for inference and training.

SETFIT is additionally not subject to the instability and inconvenience of prompting. We have also demonstrated that SETFIT is a robust few-shot text classifier in languages other than English across varying typologies. Finally, SETFIT has proven useful in few-shot distillation setups.

Acknowledgements

References

Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C. Jess Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, Michael Noetel, and Andreas Stuhlmüller. 2021. RAFT: A real-world few-shot text classification benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan

Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts.

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. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam Mc-Candlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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. *arXiv preprint arXiv:2002.06305*.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. Cite arxiv:1503.02531 Comment: NIPS 2014 Deep Learning Workshop.

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. *CoRR*, abs/1902.00751.

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*, KDD '04, page 168–177, New York, NY, USA. Association for Computing Machinery.

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. *CoRR*, abs/2001.08361.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. 2022a. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3638–3652, Dublin, Ireland. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. 2022b. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3638–3652, Dublin, Ireland. Association for Computational Linguistics.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual amazon reviews corpus.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. 2021. I wish I would have loved this one, but I didn't - A multilingual dataset for counterfactual detection in product reviews. *CoRR*, abs/2104.06893.

Christian S. Perone, Roberto Pereira Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR*, abs/1806.06259.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Guangyuan Piao. 2021. Scholarly text classification with sentence bert and entity embeddings. In *Trends and Applications in Knowledge Discovery and Data Mining*, pages 79–87, Cham. Springer International Publishing.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry,

Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

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.

Timo Schick and Hinrich Schütze. 2021. True few-shot learning with prompts - A real-world perspective. *CoRR*, abs/2111.13440.

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.

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.

I. Androutsopoulos V. Metsis and G. Paliouras. 2006. Spam filtering with naive bayes - which naive bayes? In *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006)*.

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*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *NIPS*.

A Appendix

A.1 Datasets

Table 6 shows the development and test datasets that are used for setting SETFIT's hyperparameters. Following is a description of the datasets used:

SST2   The Stanford Sentiment Treebank 2 is a collection of single sentence movie reviews with positive-negative sentiment class labels. (Socher et al., 2013).

IMDB   The Internet Movie Database dataset is a collection of single sentence movie reviews with positive-negative sentiment class labels. (Maas et al., 2011).

BBC News   The BBC News dataset is a collection of articles from the news outlet BBC with one of 5 topic classifications: Politics, Sports, Entertainment, Tech, and Business. (Greene and Cunningham, 2006).

Enron Spam   The Enron spam email dataset consists of emails from the internal Enron correspondence channel where emails are classified as spam or not spam. (V. Metsis and Paliouras, 2006).

Student Question Categories[11]   This is a set of questions from university entrance exams in India that are classified into 4 subjects: Math, Biology, Chemistry, Physics.

TREC-QC   The Text Retrieval Conference Question Answering dataset.

Toxic Conversations[12]   The Toxic Conversations dataset is set of comments from Civil Comments, a platform for reader comments for independent news outlets. Human raters have given them toxicity attributes.

Amazon Polarity[13]   The Amazon Polarity dataset consists of customer reviews from *Amazon* taken over 18 years with binary sentiment labels. Examples are either positive ("Great Read") or negative ("The Worst!") labelled. (Zhang et al., 2015a).

---

[11]www.kaggle.com/datasets/mrutyunjaybiswal/iitjee-neet-aims-students-questions-data

[12]https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification/data

[13]hf.co/datasets/amazon_polarity

Following is a description of the test datasets:

Stanford Sentiment Treebank-5 (SST5)   The SST-5 dataset is the fine-grained version of the Stanford Sentiment Treebank, where each example is given one of five labels: very positive, positive, neutral, negative, very negative.

Amazon Counterfactual   The Amazon Counterfactual dataset is set of *Amazon* customer reviews with professionally labeled binary labels of counterfactual detection. Counterfactual statements are statements that denote something that did not happen or cannot (e.g. "They are much bigger than I thought they would be."). We used the English subset for our experiments. (O'Neill et al., 2021).

Customer Reviews   The Customer Reviews (Hu and Liu, 2004) dataset is part of the of SentEval (Conneau and Kiela, 2018) benchmark. It is composed of positive and negative opinions mined from the web and written by customers about a variety of products.

Emotion[14]   The Emotion dataset consists of tweets from *Twitter* that display clear emotions (e.g. "i am now nearly finished [with] the week detox and i feel amazing"). Labels are one of six categories: anger, fear, joy, love, sadness, and surprise. (Saravia et al., 2018).

AG News   AG News is a dataset of news titles from AG news with one of 4 classifications (World, Entertainment, Sports, and Business). (Zhang et al., 2015b).

A.2 ADAPET Training Procedure

By default, ADAPET assumes access to a training, development, and test dataset. It trains for $1,000$ batches, runs predictions on the development data every 250 batches and checkpoints, keeping the model state which performed best on the development dataset. In our case, where we assume few-shot training and no development data, we ran ADAPET for $1,000$ batches and disabled the checkpointing, using the model state that resulted after training for $1,000$ batches. For the English data in Table 2, we used the pattern "[TEXT1] this is [LBL]", where "[TEXT1]" and "[LBL]" are placeholders for a given piece of text and the corresponding label, respectively. We constructed the

---

[14]hf.co/datasets/emotion

Dataset Name	Type of Task	Cls.*	Label Dist.**	Metric	Split
SST5	Sentiment	5	Approx. equal	Accuracy	Test
Amazon Counterfactual	Counterfactual	2	10% counterfactual	MCC	Test
CR	Sentiment	2	Equal	Accuracy	Test
Emotion	Emotion	6	Equal	Accuracy	Test
Enron Spam	Unwanted Language	2	Equal	Accuracy	Test
AG News	Topic	4	Equal	Accuracy	Test
SST2	Sentiment	2	Equal	Accuracy	Dev
IMDB	Sentiment	2	Equal	Accuracy	Dev
BBC News	Topic	5	Equal	Accuracy	Dev
Student Question Categories	Topic	4	Approx.Equal	Accuracy	Dev
TREC-QC	Topic	50	N/A	Accuracy	Dev
Toxic Conversations	Unwanted Language	2	8% Toxic	Avg. Precision	Dev
Amazon Polarity	Sentiment	2	Equal	Accuracy	Dev

Table 6: English datasets used for development and test experiments. *No. of classes per dataset. **Distribution of the examples across classes.

verbalizer from the "label" and "label text" columns that are available in all of our datasets. For the multilingual datasets in Table 4, we used the same pattern, but asked native speakers of each language to translate this pattern into their language. We additionally constructed the verbalizer by mapping labels to a star rating, for example, $0 = 1\ star$ and $4 = 5\ stars$, again asking native speakers of each language to translate the verbalizer into their language.

A.3 Prompts used in T-Few

The **Emotion** dataset is the only one that had existing prompts in P3 (Public Pool of Prompts) (Bach et al., 2022). For three other datasets, we had to adapt existing prompts designed for similar datasets on P3, by making minimal required changes to address the differences in data domains or label names:

- Prompts for **Enron Spam**, a spam e-mail detection dataset, were adapted from `sms_spam` dataset prompts.

- **CR** prompts were adapted from `amazon_polarity`.

- **SST5** prompts were adapted from `yelp_review_full`.

The **Amazon Counterfactual** dataset does not have any relevant prompts on P3. Hence, we manually generated prompts ourselves, based on standard practices for prompt creation published in P3.

We also added two new prompts for **SST5**, to make it compatible with the label names of **SST5**. Following is a list of prompts we created for each dataset:

Amazon Counterfactual Prompts

Input template:
```
{{ text }} Is the statement factual?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
Yes ||| No
```

Input template:
```
{{ text }} Does the statement describe
a fact?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
Yes ||| No
```

Input template:
```
{{ text }} Is the statement
non-counterfactual or counterfactual?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
non-counterfactual ||| counterfactual
```

Input template:

```
{{ text }} Is the statement
counterfactual?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
No ||| Yes
```

Input template:
```
{{ text }} Does the sentence express
an event that did not happen?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
No ||| Yes
```

Input template:
```
{{ text }} Does this describe an
actual event?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
Yes ||| No
```

Input template:
```
{{ text }} Does the sentence contain
events that did not or cannot take
place?
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
Yes ||| No
```

Input template:
```
Is the label for the following
sentence non-counterfactual or
counterfactual?  {{ text }}
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
non-counterfactual ||| counterfactual
```

New prompts for SST5

Input template:
```
How do you feel about the following
sentence?  {{ text }}
```

Target template:
```
{{ answer_choices[label] }}
```

Answer choices template:
```
very negative ||| negative ||| neutral
||| positive ||| very positive
```

Input and target templates:
```
{{ text }} This movie is a very
{{answer_choices[label]}} one
```

Answer choices template:
```
terrible ||| bad ||| okay ||| good |||
great
```