STT: Soft Template Tuning for Few-Shot Learning

Anonymous ACL submission

Abstract

With the rapid expansion of large pre-trained language models, fine-tuning all the model parameters for downstream tasks is becoming computationally prohibitive. The recently developed prompt-based methods freeze the entire model parameters and only update the so-called prompt parameters appended to the inputs, significantly reducing the burden of fully fine-tuning. However, standard prompt-based methods mainly consider the case where sufficient data of downstream tasks are available. It is still unclear whether the advantage can be transferred to the few-shot regime, where only limited data are available for each downstream task. Our empirical studies suggest there is still a gap between prompt tuning and fully fine-tuning for few-shot learning. We propose a new prompt-tuning framework, called Soft Template Tuning (STT), to bridge the gap. STT combines manual prompts and auto-prompts, and treats downstream classification tasks as a masked language modeling task. STT can close the gap between fine-tuning and prompt-based methods without introducing additional parameters. Importantly, it can even outperform the time- and resource-consuming fine-tuning method on sentiment classification tasks.

1 Introduction

With the success of pre-trained large language models, an increasing number of techniques have been proposed to adapt these general-propose models to downstream tasks. Starting from GPT (Radford et al., 2018) and BERT (Devlin et al., 2018), **full model fine-tuning** is used as the default method to adapt pre-trained language models to downstream tasks. With the rapid increase of model sizes (Lewis et al., 2019; Liu et al., 2019a; Raffel et al., 2019), it has gradually become more challenging to update all model parameters during fine-tuning.

One extreme case is the GPT-3 model (Brown et al., 2020), where the model size is too large (175B model parameters) to even enable fine-tuning, which sets a barrier for the community to involve in the research. Alternatively, **in-context learning** is proposed, which demonstrates few-shot capabilities without tuning any model parameters. However, although in-context learning enables jumbo language models to be applied on diverse downstream tasks, it is not as effective as fine-tuning, which significantly restricts the advantages and applicability of these large pre-trained language models.

Meanwhile, the ideas have been explored to update a small number of model parameters while keeping most model parameters frozen. (Li and Liang, 2021) proposes **prefix-tuning** and shows strong performance on generative tasks. This method freezes the model parameters and propagates the error during fine-tuning to prefix activation prepended to each layer in the encoder stack, including the input layer. (Hambardzumyan et al., 2021) simplifies this method by restricting the trainable parameters only to the input and output subnetworks of a masked language model, and shows reasonable results on classification tasks. (Lester et al., 2021) proposes a further simplification – **prompt-tuning** for adapting language models, which shows that prompt tuning alone (with no intermediate-layer prefixes) is sufficient to be competitive with model tuning. However, (Lester et al., 2021) only test prompt tuning in the case where each task has enough data for adaptation. It is not clear whether prompt tuning is still effective in the few-shot regime, where only a limited number of samples are available for each task. We find through empirical evidence that prompt tuning lags far behind the fine-tuning in the few-shot regime. This can be explained by the intuition that using only a few samples to learn a relatively small portion of parameters might be too flexible and tend to overfit the few training data, similar to the traditional machine-learning setting.

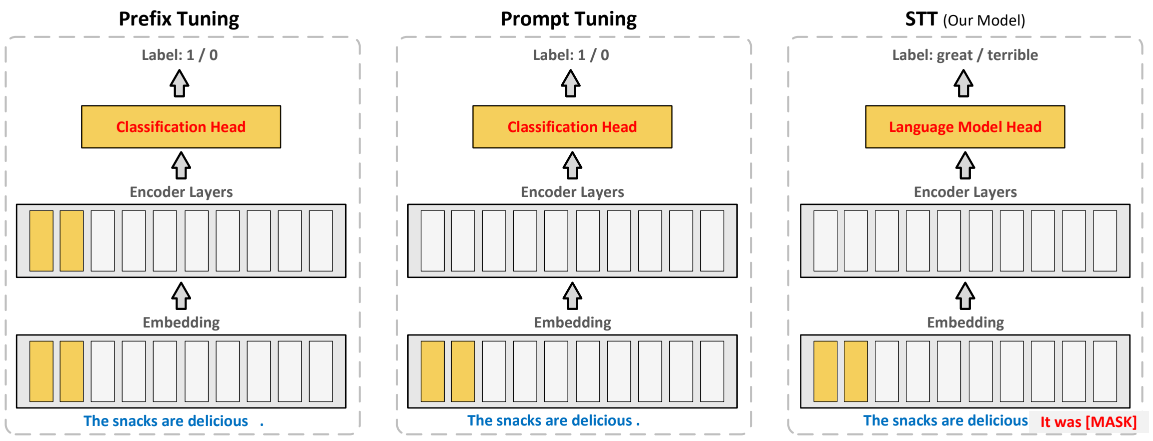

Figure 1: **Model Comparison:** We compare STT with Prefix tuning (Li and Liang, 2021) and Prompt tuning (Lester et al., 2021). The yellow box means parameters need to be updated. The white box means parameters are fixed. STT, on the basis of the Prompt tuning method, adds a human-defined template to the input sentence, and replaces the classifier head with the language model head to predict whether the word in the mask position is great or terrible.

We propose improved techniques to alleviate this problem in the practical scenario where a moderate-sized pre-trained language model (*e.g.*, RoBERTa large (Liu et al., 2019b)) and a small number of examples for each downstream task (*i.e.*, the *few-shot* setting) are accessible. This setting is common in practice: (1) Such types of models are publicly available and the computational resources needed are easily accessible for most researchers; (2) The few-shot settings are realistic, as new tasks usually come with few examples. To address the problems mentioned above, we explore the idea of closing the gap between pre-training and fine-tuning. Specifically, we propose to add manual prompts on the basis of prompt-tuning (Lester et al., 2021) and treat the problem to be a masked language modeling problem, as was done by most pre-training. Since we combine manual template with soft prompt, we call our method Soft Template Tuning (STT). Experiments demonstrate that STT is able to close the gap between prompt-based methods and fine-tuning in the few-shot regime. Remarkably, STT can even outperform the computationally heavy fine-tuning on some tasks, though it tunes much fewer parameters than the fully fine-tuning method.

2 Soft Template Tuning

Our task is to adapt a pre-trained masked language model to downstream tasks with a dataset $\mathcal{D}$. To realize the few-shot regime, we only sample K examples from $\mathcal{D}$ as the training dataset $\mathcal{D}_{\text{train}} = (x_i, y_i)_{i=1}^{K}$, and use the original test dataset $\mathcal{D}_{\text{test}}$ for verification. We define $F : \mathcal{Y} \longrightarrow \mathcal{V}$ as a mapping from the label space $\mathcal{Y}$ to the word space with a vocabulary $\mathcal{V}$ in the pre-trained language model.

Manual Template Design We first construct prompted input $\hat{x}_i$ by adding manual prompts to the input data x_i. Inspired by Gao et al. (2020), we adapt manual prompts for different tasks, which are defined in Table 3 in the Appendix. As an example, we augment an input x_1 (*e.g.*, *The snacks are delicious.*) as follows for the classification task:

$$\hat{x}_1 = \text{[CLS] } x_1 \text{ it was [MASK] . [SEP]}$$

After obtaining $\hat{x}_1$, we translate it into a hidden vector representation with the pre-trained embedding layer of the language model.

Soft Prompt Tuning We next combine the manual template with a soft prompt to enrich the input. We sample M words $\boldsymbol{Z} \triangleq (z_1, \cdots, z_M)$ from the vocabulary of the pre-trained language model, and then initialize a random embedding layer to represent the input $\boldsymbol{Z}$ as hidden vectors, which are learned during the tuning process.

We then concatenate the hidden representations of the manual prompts and learnable prompts to form a new hidden vector, which is fed as the input to the transformer layers of language model. Instead of adopting the classification objective in the standard prompt-based methods, we close the gap between pre-training and fine-tuning by treating the tuning task as an masked language model (MLM) task. The probability of predicting the corresponding class y is defined as

$$\begin{aligned} p(y \mid x) &= p(\text{[MASK]} = F(y) \mid \hat{x}, \boldsymbol{Z}) \\ &= \frac{\exp(\boldsymbol{w}_{F(y)} \cdot \boldsymbol{h}_{\text{[MASK]}})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{F(y')} \cdot \boldsymbol{h}_{\text{[MASK]}})}, \end{aligned} \quad (1)$$

	Models	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	TREC (acc)	SNLI (acc)	QNLI (acc)	QQP (F1)	Average
One-shot	Fine-tuning	53.1±4.4	23.4±1.8	53.3±3.6	52.8±2.8	51.3±1.8	28.7±10.6	35.2±2.6	51.0±1.0	45.4±11.4	43.8
	Prefix-tuning	51.0±2.8	22.9±3.9	51.2±2.2	51.0±2.7	53.0±1.9	**27.9**±6.1	33.5±0.8	49.8±1.6	46.4±4.9	42.9
	Prompt-tuning	51.5±2.2	22.7±2.8	51.5±2.1	52.1±3.5	51.9±2.1	17.3±3.3	**35.1**±2.0	50.6±0.5	**47.8**±8.7	42.3
	STT (ours)	**57.3**±5.6	**23.5**±3.3	**52.1**±1.3	**55.5**±5.1	**55.2**±3.7	26.0±11.7	33.8±1.5	**51.8**±1.0	45.4±11.9	**44.5**
Two-shot	Fine-tuning	55.2±8.8	27.0±2.8	55.9±3.0	58.5±3.4	55.6±6.3	43.8±6.9	36.4±3.5	52.3±1.2	53.4±4.6	48.7
	Prefix-tuning	51.7±2.4	23.5±2.3	52.1±1.8	53.4±3.6	55.3±4.4	**35.8**±4.0	**34.4**±0.6	51.3±2.5	46.0±6.8	44.8
	Prompt-tuning	51.9±5.0	19.9±2.4	51.6±2.3	53.5±2.1	52.7±3.7	22.3±9.0	34.4±1.8	50.2±1.5	**49.0**±12.9	42.8
	STT (ours)	**61.9**±2.7	**26.0**±3.5	**53.9**±4.4	**60.3**±4.2	**60.6**±3.3	23.6±5.5	33.4±1.1	**51.6**±0.9	48.1±9.4	**46.6**
Five-shot	Fine-tuning	60.0±5.1	32.3±2.1	58.2±3.7	63.9±6.0	59.2±1.5	61.3±12.2	36.8±2.6	53.9±3.3	53.5±5.2	53.2
	Prefix-tuning	54.9±5.8	25.4±1.3	54.3±3.0	56.4±4.0	56.2±3.7	**29.6**±14.0	34.4±1.1	51.5±1.0	**46.0**±10.3	45.4
	Prompt-tuning	52.2±6.7	20.6±3.5	55.6±6.4	56.4±4.4	53.7±2.2	21.2±5.7	**35.8**±1.6	51.8±1.7	47.8±9.5	43.9
	STT (ours)	**67.7**±5.1	**30.6**±2.1	**62.1**±3.0	**62.9**±4.1	**61.5**±2.3	25.4±3.9	33.1±0.6	**51.9**±0.7	44.3±12.9	**48.9**

Table 1: Evaluation Results (mean and variance over 5 random trails) for one-shot, two-shot, and five-shot settings. The best results across the prompt-based methods (Prefix-tuning, Prompt-tuning, and STT) are shown in bold. Our STT approach exceeds both Prefix-tuning and Prompt-tuning in most tasks and on average.

Models	Embedding Layers	Transformer Layers	Head Layers[1]	Total
Prefix-tuning	0.026M	20.752M	1.052M	**21.83M**
Prompt-tuning	0.026M	0M	1.052M	**1.08M**
STT (ours)	0.026M	0M	1.054M	**1.08M**

Table 2: Trainable parameters for the prompt-based methods (with prompt length 25). Note that, the total trainable parameters for fine-tuning is **355.36M**, which is much larger than all of the prompt-based methods.

where $\boldsymbol{h}_{[\text{MASK}]}$ is the hidden vector of [MASK] and $\boldsymbol{w}_{F(y)}$ denotes the linear weights before softmax function for class y. It is important to note that we have re-used the language model linear weights from the pre-trained language model. We also show in Appendix A.4 that tuning the language model head could gain significant improvement.

3 Experiments

3.1 Experimental Setup

Baselines We compared our proposed STT with the two recent prompt-based methods: prefix-tuning (Li and Liang, 2021) and prompt-tuning (Lester et al., 2021). We show structure differences between our model and these two prompt-based methods in Figure 1. To illustrate the gap between prompt tuning and fine-tuning more clearly, we also provide the fine-tuning results. All of the baselines in our paper are based on the RoBERTa-large (Liu et al., 2019b) model in the HuggingFace Transformers codebase (Wolf et al., 2020).

Evaluation Tasks To conduct a systematic evaluation, we consider a total of 9 tasks, covering different domains and difficulties. Broadly speaking, the evaluation set includes 6 single-sentence tasks and 3 sentence-pair tasks. Specifically, we select 3 tasks from the GLUE benchmark (Wang et al., 2018) (one from each category, including SST-2, QQP, and QNLI), and 6 from other popular NLU benchmarks (SST-5, MR, CR, MPQA, TREC, SNLI). All the details of selected tasks are provided in Appendix A.2. We follow Gao et al. (2020) for pre-processing and use the same way to sample the testing set.

Evaluation Settings Without loss of fairness, we evaluate our STT approach based on the pre-trained RoBERTa-large (Liu et al., 2019b) model as per the baseline methods. To realize the few-shot setting, for each task, the evaluation is conducted by training on K samples per class, where $K = 1, 2, 5$. We tested all the models with 5 seeds (13, 21, 42, 87, 100) and calculated the mean and variance. More details can be found in Appendix A.3.

3.2 Main Results

Table 1 (Top) shows the one-shot evaluation results. Regarding the number of trainable parameters, our model only tunes 1.13M parameters, which is almost the same as prompt-tuning (Lester et al., 2021), less than prefix-tuning (Li and Liang, 2021), and far less than fine-tuning. In this set of experiments, although prefix-tuning and prompt-tuning surpass fine-tuning in some tasks, it still lags behind fine-tuning on average. Comparing with the prompt-tuning method (Lester et al., 2021), prefix-tuning adds prefix parameters to each transformer layer. It gains 0.6% performance improvement but at the expense of introducing additional parameters. Our method, on the basis of prompt-tuning (Lester et al., 2021), improves the average accuracy by 2.2 points. With only 0.3% trainable parame-

[1]This column is related to the number of task labels, and is calculated based on bi-classification.

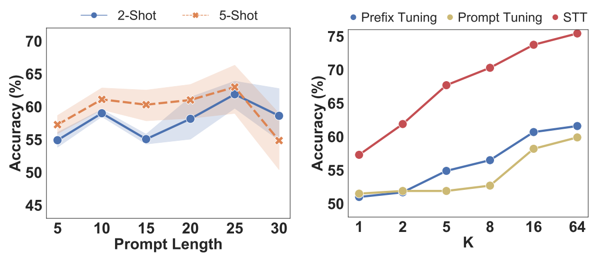

Figure 2: **Left:** Performance on SST-2 task over different prompt lengths. A prompt length of 25 generally gives better results. **Right:** Performance comparison over different K (# samples per class) size.

ters, it even surpasses the fine-tuning method in most tasks, resulting in a 0.7 points improvement on average.

Table 1 (Middle and Bottom) shows the evaluation results for two-shot and five-shot scenarios. Similarly, our STT approach surpasses prefix-tuning (Li and Liang, 2021) and prompt-tuning (Lester et al., 2021) in general. Surprisingly, when compared with fine-tuning, our method still achieves significantly better results on SST-2, MR, and MPQA.

Table 2 provides a comparison on model parameters. Although language model head used a linear transformation for mapping hidden vector to dictionary size, we only utilize the matrix with indices of the label words (which will be 2 for bi-classification task, 3 for Tri-classification task, 6 for TREC task).

3.3 Ablation Studies

Prompt Length Prompt length is a critical hyper-parameter for prompt-based methods. We, therefore, test the impact of prompt length on our STT. We initialize STT with various prompt lengths from 5 to 30, with an increment of 5. Intuitively, more prompt tokens indicate better expressive power but introduce slightly more trainable parameters. The results are plotted in Figure 2 (left). We observe that the best results are achieved with around a certain prompt length (in our case, 25 for both 2-shot and 5-shot settings), while inserting more prompt tokens beyond this point may yield a performance drop, especially in the few-shot setting.

K **Size** We also evaluate how well STT works by increasing K in comparison with Prefix Tuning (Li and Liang, 2021) and Prompt Tuning (Lester et al., 2021). As shown in Figure 2 (Right), with K increasing all the way up to 64, the performance of STT keeps increasing, which consistently outperforms the other prompt-based methods, demonstrating a great potential when more training data become available.

3.4 Discussion

We find that existing prompt-based methods do not work well in this setting, whereas our STT can dramatically improve over the prompt-based methods without introducing additional parameters. Although our trainable parameters are much less than fine-tuning, STT can still outperform the fine-tuning method on SST-2, MR, MPQA. Most of these tasks belong to sentiment classification tasks. The performance improvement over the fine-tuning method may be attributed to the benefit of the manually-designed templates. Therefore, the performance of STT depends on the quality of manual templates and label words choice, which can be one limitation of our method.

In our experiments, we choose a public pre-trained model (RoBERTa large) in the huggingface transformer library with 355.36M parameters in total, which is a moderate-sized pre-trained language model. As the model size increases, the advantage of the prompt-based models over fine-tuning will be more obvious. (Lester et al., 2021) concludes that the gap between fine-tuning and prompt-based methods is closing when increasing the pre-trained language model size. Prompt-tuning method (Lester et al., 2021) could achieve competitive results when the model size reaches 100B. We will validate our STT on a larger pre-trained language model for future work.

4 Conclusion

To adapt prompt-based methods in the few-shot regime, we propose STT, a simple method that combines both manual templates and soft prompts, and treats the downstream classification tasks as a language modeling task. We conduct experiments on 9 downstream classification tasks. Experiment results reveal that STT outperforms both fine-tuning and prompt-based methods under the one-shot setting. In the two-shot and five-shot settings, our approach closes the gap between the fine-tuning method and prompted-based methods, which can even outperform fine-tuning on classification tasks. Our research indicates that prompt-tuning is an effective tool for adapting large pre-trained models, yet still there is room for further improvement, especially in the few-shot regime.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. arXiv preprint arXiv:2101.11038.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

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.

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

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: adapt language models to domains and tasks. arXiv preprint arXiv:2004.10964.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. arXiv preprint arXiv:2101.00121.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

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.

Shankar Iyer, Nikhil Dandekar, Kornél Csernai, et al. 2017. First quora dataset release: Question pairs. data. quora. com.

Teven Le Scao and Alexander M 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.

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, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190.

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. 2019a. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

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. arXiv preprint arXiv:1907.11692.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. arXiv preprint cs/0506075.

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.

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 models for natural language understanding: When and why does it work? arXiv preprint arXiv:2005.00628.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. arXiv preprint arXiv:2104.06599.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. arXiv preprint arXiv:2001.07676.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing, pages 1631–1642.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, pages 200–207.

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.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. Language resources and evaluation, 39(2):165–210.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. arXiv preprint arXiv:2006.05987.

A Appendix

A.1 Related Works

Language Model Fine-tuning A prevalent idea of adapting pre-trained language models for a wide range of downstream tasks is to fine-tune the pre-trained models with task specific heads. Since it became a standard practise of transfer learning in NLP domain, a number of efforts have been made seeking better ways of language model fine-tuning (Howard and Ruder, 2018; Gururangan et al., 2020; Phang et al., 2018; Aghajanyan et al., 2021; Dodge et al., 2020; Pruksachatkun et al., 2020; Zhang et al., 2020). However, fine-tuning usually requires the updating of all the model parameters, and therefore, a copy of the whole model needs to be stored for each specific downstream task. It is hence expensive for both computational and storage resources. Another significant problem lies in the different objective formats between the pre-training stage and fine-tuning stage. This not only leaves a gap between the language model and downstream tasks, but also introduces new parameters, making the fine-tuning less effective in the few-shot setting.

Prompting Prompting was recently proposed aiming at the above issues, especially for giant language models like GPT-3 (Brown et al., 2020). A prompt is usually referred to as some extra text information added to the input. By concatenating the input with a sequence of tokens on which the language model could condition, prompting enables the downstream tasks to adopt the same type of objective as the pre-training stage, and therefore, closes the gap between the 2 stages. Taking sentiment analysis as an example, the input sentence is concatenated with a prompt (e.g., "it was [MASK]"), where the label word is masked and to be predicted by the language model. Then the sentiment class could be inferred based on which of the selected label word is predicted (e.g., "great" as positive and "terrible" as negative). In addition, since prompting introduces no new parameters and requires no training (all parameters are fixed), it was demonstrated to be effective in the few-shot setting (Le Scao and Rush, 2021), and designing or searching for optimal prompts becomes a crucial issue (Schick and Schütze, 2020; Shin et al., 2020).

Prompt-tuning Instead of adding discrete prompt tokens from the vocabulary, prompt-tuning uses soft prompts (in the form of trainable continuous embeddings) and achieved significant improvements over prompting in many tasks (Lester et al., 2021). Specifically, Li and Liang (2021) proposed prefix-tuning for conditional generation tasks, where continuous embeddings were prepended to each layer of the encoder (and decoder if applicable) architecture as prefix. By tuning only the prefix parameters, comparable performance to fine-tuning was observed in full dataset setting. Soft prompts were also proved to be effective in knowledge probing tasks when inserting prompts into different positions of the input according to a manually determined pattern (Qin and Eisner, 2021; Liu et al., 2021). Some other tasks (Hambardzumyan et al., 2021; Lester et al., 2021) further simplified prefix-tuning by excluding intermediate-layer prefixes and only restricting the trainable parameters to the input. Despite prompt-tuning has demonstrated strength even comparable to fine-tuning when given enough data, we observed a considerable gap in performance between prompt-tuning and fine-tuning in the few-shot setting.

A.2 Datasets

The evaluation datasets are composed of 6 single-sentence tasks and 3 sentence-pair tasks. The single-sentence tasks, including SST-2, SST-5 (Socher et al., 2013), MR (Pang and Lee, 2005), CR (Hu and Liu, 2004), MPQA (Wiebe et al., 2005), and TREC (Voorhees and Tice, 2000), are used to test the model's performance on predicting the label word for each input sentence. The sentence-pair tasks, including SNLI (Bowman et al., 2015), QNLI (Rajpurkar et al., 2016), QQP (Iyer et al., 2017), are introduced to measure how well a model is by comparing the relationships between 2 input sentences. Among them, SST-2, QQP, and QNLI are each selected from one category of the GLUE benchmark (Wang et al., 2018).

As summarized in Table 3, we select the evaluation tasks of various types and cover diverse domains. SST-2, SST-5, MR, CR, and MPQA belongs to sentiment analysis. TREC classifies open-domain, fact-based questions in to different classes. SNLI and QNLI, respectively, are datasets for the inference of sentence relations and question answers. And QQP is a sentence similarity task from the social question-and-answer community Quora. In formulating them as masked language modeling task, we utilize manual templates and label words intuitively designed for each task (Table 3).

Task	Template	Label Words	Domain
Sentiment Analysis Tasks			
SST-2	<S1> it was [MASK] .	positive: great, negative: terrible	movie reviews
SST-5	<S1> it was [MASK] .	positive: great, neutral: okay, negative: terrible	movie reviews
MR	<S1> it was [MASK] .	positive: great, neutral: okay, negative: terrible	movie reviews
CR	<S1> it was [MASK] .	positive: great, neutral: okay, negative: terrible	customer reviews
MPQA	<S1> it was [MASK] .	positive: great, negative: terrible	news opinions
Question Classification Tasks			
TREC	[MASK] : <S1>	abbreviation: Expression, entity: Entity, description: Description, human: Human, location: Location, numeric: Number	misc.
Inference Tasks			
SNLI	<S1> ? [MASK] , <S2>	entailment: yes, neutral: maybe, contradiction: no	misc.
QNLI	<S1> ? [MASK] , <S2>	entailment: yes, not_entailment: no	Wikipedia
Similarity Tasks			
QQP	<S1> [MASK] , <S2>	equivalent: yes, not_equivalent: no	social QA questions

Table 3: Task types and domains in our experiments with manual templates and label words.

Seeds	13	21	42	87	100	Average
w/o updated lm_head	53.56	53.78	53.78	53.33	53.55	53.56±0.2
w/ updated lm_head	**62.73**	**63.07**	**68.92**	**55.96**	**64.33**	**63.00**±4.6

Table 4: Comparison between tuning with trainable lm_head and fixed lm_head. lm_head stands for language model head.

We follow Gao et al. (2020) for pre-processing and use the same way to sample the testing set.

A.3 Evaluation Settings

Hyper-parameters tuning is performed with a small development set, which is set to the same size as the training set. As pointed out by Gao et al. (2020), this can avoid the significant advantages of using large development set and complies better with the goal of few-shot setting. For simplicity, instead of tuning hyper-parameters for each task, we only tune and select the hyper-parameters on SST-2, and apply the tuned parameters to remaining tasks. For each task, we conduct training for 500 steps, with a batch size of 2, a learning rate of $2e^{-5}$, and a prompt length of 25. Furthermore, the ameliorate unstable performance induced by the randomness of sampling a small dataset, we repeat each task and average the performance with variance over 5 random trials.

A.4 The Role of Trainable Language Model Head

To demonstrate the importance of tuning the language model head, we conducted a comparative experiment between making the language model head fixed and trainable for tuning. The experiment was performed on SST-2 task over five random trials, and with a prompt length of 25. The results are concluded in Table 4. Tuning with language model head updated makes a marked improvement in all experiments, illustrating that it is essential to make the language model head trainable for our method.

A.5 Ethical Considerations

We need to point out, that bias might be introduced by training. This, however, is owing to the limitations of the dataset [2] and few-shot training samples at hand, and does not represent a flaw in the model.

[2]As pointed out in https://huggingface.co/roberta-large, the training data contains unfiltered and un-neutral content from the internet. Therefore, the model can have biased predictions.