STT: Soft Template Tuning for Few-Shot Adaptation

Anonymous submission

***Abstract*—Prompt tuning has been an extremely effective tool to adapt a pre-trained model to downstream tasks. However, standard prompt-based methods mainly consider the case of sufficient data of downstream tasks. 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. Although some works have demonstrated the potential of prompt-tuning under the few-shot setting, the main stream methods via searching discrete prompts or tuning soft prompts with limited data are still very challenging. Through extensive empirical studies, we find that there is still a gap between prompt tuning and fully fine-tuning for few-shot learning. To bridge the gap, we propose a new prompt-tuning framework, called Soft Template Tuning (STT). STT combines manual and auto prompts, and treats downstream classification tasks as a masked language modeling task. Comprehensive evaluation on different settings suggests STT can close the gap between fine-tuning and prompt-based methods without introducing additional parameters. Significantly, it can even outperform the time- and resource-consuming fine-tuning method on sentiment classification tasks.**

***Index Terms*—NLP, few-shot learning, prompt-tuning, language model**

I. 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 [1] and BERT [2], **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 [3]–[5], it has gradually become more challenging to update all model parameters during fine-tuning.

One extreme case is the GPT-3 model [6], 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, ideas have been explored to update a small number of model parameters while keeping most parameters frozen. [7] proposed **prefix-tuning** which 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. [8] simplified this method by restricting the trainable parameters only to the input and output subnetworks of a masked language model, and showed reasonable results on classification tasks. [9] proposed a further simplification – **prompt-tuning** for adapting language models, which shows that prompt tuning alone (with no intermediate-layer prefixes) is competitive with model tuning. However, [9] only test prompt tuning in the case where each task has enough data for adaptation.

Why Prompt Tuning in the Few-Shot Regime: To the best of our knowledge, only tuning prompt tokens for few-shot adaptation is under-explored. Note that related works have different settings with ours. For example, [10] and [11] combines both discrete prompts and soft prompts, however, they fine-tune the soft prompts jointly with the entire pre-trained model; P-tuning v2 [12] further enhanced the universality of [10] on a broader selection of models and tasks at the cost of introducing soft prompts (more trainable parameters) to every layer of the pre-trained model, and is therefore only evaluated in the fully-supervised setting rather than few-shot setting. [13] proposed Pre-trained Prompt Tuning framework (PPT). They proved that after pre-training on soft prompts, prompt tuning could reach or even outperform full-model fine-tuning under few-shot settings. However, the improvement was achieved at the cost of introducing additional pre-training efforts, moreover different types of pre-training tasks need to be recruited to accommodate diverse downstream tasks (e.g., single sentence, sentence-pair, multiple-choice classifications). For a comprehensive review of popular prompt-tuning based methods, please refer to Section V-A in the Appendix. 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 over-fit the few training data, similar to the traditional machine-learning setting.

Our Proposed Method: We propose improved techniques to alleviate this problem in the practical scenario where a moderate-sized pre-trained language model (*e.g.*, RoBERTa large [14]) 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

* These authors contributed equally to this work.

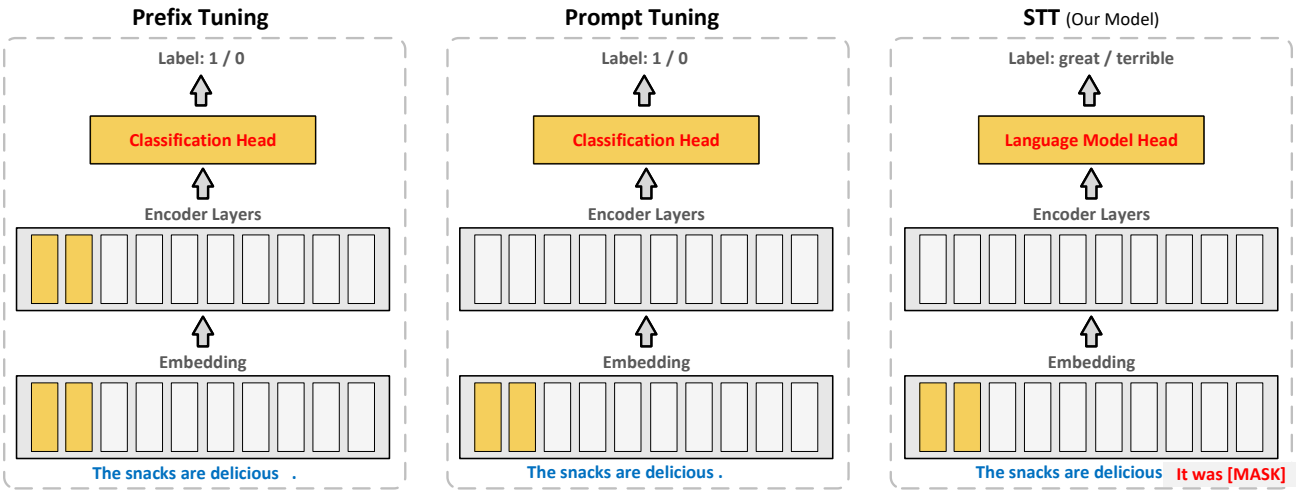

Fig. 1: **Model Comparison:** We compare STT with Prefix tuning [7] and Prompt tuning [9]. The yellow box means parameters need to be updated. The white box means parameters are fixed. STT, on the basis of the Prompt tuning method, adds a human-defined template to the input sentence, and replaces the classifier head with the language model head to predict whether the word in the mask position is great or terrible.

fine-tuning. Specifically, we propose to add manual prompts on the basis of prompt-tuning [9] 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.

II. 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 evaluation. 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.

a) Manual Template Design: We first construct prompted input $\hat{x}_i$ by adding manual prompts to the input data x_i. Inspired by [15], we adapt manual prompts for different tasks, which are defined in Appendix Table I. 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.

b) 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

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

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 Section IV-B Ablation Study that tuning the language model head could gain significant improvement.

III. Experiments

A. Experimental Setup

a) Baselines: We compared our proposed STT with the two recent prompt-based methods: prefix-tuning [7] and prompt-tuning [9]. 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 [14] model in the HuggingFace Transformers codebase [16].

b) Evaluation Settings: Without loss of fairness, we evaluate our STT approach based on the pre-trained RoBERTa-large [14] model as per the baseline methods. To realize the few-shot setting, for each task, evaluation is conducted by training on K samples per class ($K = 1, 2, 5$). We tested all the models with 5 seeds (13, 21, 42, 87, 100) and calculated mean and variance.

c) Hyper-parameters: 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 [15], 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.

B. 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, the single-sentence tasks, including SST-2, SST-5 [17], MR [18], CR [19], MPQA [20], and TREC [21], are used to test the model's performance on predicting the label word for each input sentence. The sentence-pair tasks, including SNLI [22], QNLI [23], QQP [24], 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 [25].

As summarized in Table I, 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 into 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 I).

We follow [15] for pre-processing and use the same way to sample the testing set.

IV. Results

A. Main Results

Table II (Top) shows the one-shot evaluation results. Regarding the number of trainable parameters, our model only tunes 1.08M parameters, which is almost the same as prompt-tuning [9], less than prefix-tuning [7], and far less than fine-tuning. In this set of experiments, although prefix-tuning and prompt-tuning surpass fine-tuning in some tasks, they still lag behind fine-tuning on average. Comparing with the prompt-tuning method [9], prefix-tuning gains 0.6% performance improvement, but at the expense of introducing additional prefix parameters to each transformer layer. Our method, on the basis of prompt-tuning [9], improves the average accuracy by 2.2 points. With only 0.3% trainable parameters, it even surpasses the fine-tuning method in most tasks, resulting in a 0.7 points improvement on average.

Table II (Middle and Bottom) shows the evaluation results for two-shot and five-shot scenarios. Similarly, our STT approach surpasses prefix-tuning [7] and prompt-tuning [9] in general. Surprisingly, when compared with fine-tuning, our method still achieves significantly better results on SST-2, MR, and MPQA.

Table III provides a comparison on model parameters. Although language model head uses 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).

Discussion could be found in Section V.

B. Ablation Studies

a) 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.

b) K Size: We also evaluate how well STT works by increasing K in comparison with Prefix Tuning [7] and Prompt Tuning [9]. 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.

c) 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 or 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 IV. 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.

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

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

	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
	Adapter	51.0 ±2.9	18.9±5.9	50.2±0.5	49.7±0.8	50.0±0.1	23.1±4.4	33.3±0.3	50.3±1.9	40.2±7.9	39.5
	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
	Adapter	50.2 ±0.8	17.9±6.5	50.0±0.0	50.0±0.0	50.0±0.1	20.0±7.3	33.4±0.4	50.4±0.5	40.2±11.8	40.2
	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
	Adapter	49.8 ±0.9	17.9±6.5	50.0±0.0	50.0±0.0	50.0±0.1	16.2±9.8	33.2±0.4	49.5±0.1	40.5±11.2	40.5
	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 II: 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 III: 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.

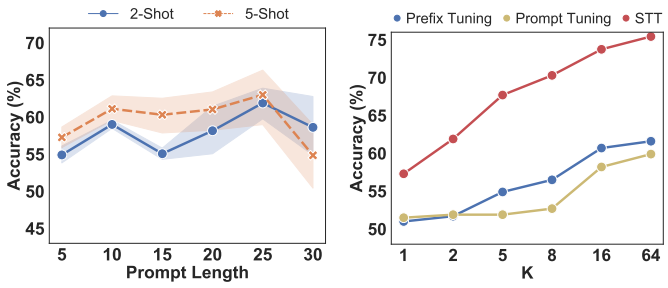

Fig. 2: **Left:** Performance on SST-2 task over different prompt lengths. A prompt length of 25 generally gives better results. **Right:** Performance comparison over different K (# samples per class) size.

V. 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

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 IV: Comparison between tuning with trainable *lm_head* and fixed *lm_head*. *lm_head* stands for language model head.

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. [9] concludes that the gap between fine-tuning and prompt-based methods is closing when increasing the pre-trained language model size. Prompt-tuning method [9] 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.

For 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.

A. Related Works

a) 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 [26]–[32]. 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.

b) Prompting: Prompting was recently proposed aiming at the above issues, especially for giant language models like GPT-3 [6]. 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 [33], and designing or searching for optimal prompts becomes a crucial issue [34], [35].

c) 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 [9]. Specifically, [7] 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. Some other tasks [8], [9] further simplified prefix-tuning by excluding intermediate-layer prefixes and only restricting the trainable parameters to the input. 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 [10], [36]. **P-tuning** [10] demonstrated that by searching soft prompts in the continuous space, GPT-style models could achieve competitive performance with similar-size BERTs in NLU tasks. It also claims that adding task-related anchor tokens could bring further improvement. However, P-tuning introduces extra parameters by using bidirectional LSTM for prompt embedding, and hence fine-tunes the prompt embeddings together with the the pre-trained model for SuperGLUE [37] tasks. P-tuning v2 [12] further enhanced the universality of P-tuning on a broader selection of models and tasks at the cost of introducing soft prompts (more trainable parameters) to every layer of the pre-trained model, and is therefore only evaluated in the fully-supervised setting rather than few-shot setting. 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. [13] proposed Pre-trained Prompt Tuning framework (PPT) and proved that their approach could reach or even outperform full-model fine-tuning under few-shot settings. However, according to the format of task, they require that soft prompts need to be pre-trained on a designed self-supervised task before fine-tuning the prompts on the downstream task.

VI. Conclusion

To adapt prompt-based methods in the few-shot regime, we propose STT, a simple method that combines both manual templates and soft prompt, 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

[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.

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

[1] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[3] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," arXiv preprint arXiv:1910.13461, 2019.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," arXiv preprint arXiv:1910.10683, 2019.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," arXiv preprint arXiv:2005.14165, 2020.

[7] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," arXiv preprint arXiv:2101.00190, 2021.

[8] K. Hambardzumyan, H. Khachatrian, and J. May, "Warp: Word-level adversarial reprogramming," arXiv preprint arXiv:2101.00121, 2021.

[9] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," arXiv preprint arXiv:2104.08691, 2021.

[10] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," arXiv preprint arXiv:2103.10385, 2021.

[11] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification," arXiv preprint arXiv:2105.11259, 2021.

[12] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," arXiv preprint arXiv:2110.07602, 2021.

[13] Y. Gu, X. Han, Z. Liu, and M. Huang, "Ppt: Pre-trained prompt tuning for few-shot learning," arXiv preprint arXiv:2109.04332, 2021.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[15] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," arXiv preprint arXiv:2012.15723, 2020.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[17] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631–1642.

[18] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," arXiv preprint cs/0506075, 2005.

[19] M. Hu and B. Liu, "Mining and summarizing customer reviews," in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, pp. 168–177.

[20] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," Language resources and evaluation, vol. 39, no. 2, pp. 165–210, 2005.

[21] E. M. Voorhees and D. M. Tice, "Building a question answering test collection," in Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000, pp. 200–207.

[22] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," arXiv preprint arXiv:1508.05326, 2015.

[23] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," arXiv preprint arXiv:1606.05250, 2016.

[24] S. Iyer, N. Dandekar, K. Csernai et al., "First quora dataset release: Question pairs," data. quora. com, 2017.

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

[26] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," arXiv preprint arXiv:1801.06146, 2018.

[27] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: adapt language models to domains and tasks," arXiv preprint arXiv:2004.10964, 2020.

[28] J. Phang, T. Févry, and S. R. Bowman, "Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks," arXiv preprint arXiv:1811.01088, 2018.

[29] A. Aghajanyan, A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer, and S. Gupta, "Muppet: Massive multi-task representations with pre-finetuning," arXiv preprint arXiv:2101.11038, 2021.

[30] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," arXiv preprint arXiv:2002.06305, 2020.

[31] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. R. Bowman, "Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?" arXiv preprint arXiv:2005.00628, 2020.

[32] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi, "Revisiting few-sample bert fine-tuning," arXiv preprint arXiv:2006.05987, 2020.

[33] T. Le Scao and A. M. Rush, "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, 2021, pp. 2627–2636.

[34] T. Schick and H. Schütze, "Exploiting cloze questions for few shot text classification and natural language inference," arXiv preprint arXiv:2001.07676, 2020.

[35] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," arXiv preprint arXiv:2010.15980, 2020.

[36] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," arXiv preprint arXiv:2104.06599, 2021.

[37] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," Advances in neural information processing systems, vol. 32, 2019.