PPT: Pre-trained Prompt Tuning for Few-shot Learning

Anonymous ACL submission

Abstract

Prompts for pre-trained language models (PLMs) have shown remarkable performance by bridging the gap between pre-training tasks and various downstream tasks. Among these methods, prompt tuning, which freezes PLMs and only tunes soft prompts, provides an efficient and effective solution for adapting large-scale PLMs to downstream tasks. However, prompt tuning is yet to be fully explored. In our pilot experiments, we find that prompt tuning performs comparably with conventional full-model tuning when downstream data are sufficient, whereas it is much worse under few-shot learning settings, which may hinder the application of prompt tuning. We attribute this low performance to the manner of initializing soft prompts. Therefore, in this work, we propose to pre-train prompts by adding soft prompts into the pre-training stage to obtain a better initialization. We name this Pre-trained Prompt Tuning framework “PPT”. To ensure the generalization of PPT, we formulate similar classification tasks into a unified task form and pre-train soft prompts for this unified task. Extensive experiments show that tuning pre-trained prompts for downstream tasks can reach or even outperform full-model fine-tuning under both full-data and few-shot settings. Our approach is effective and efficient for using large-scale PLMs in practice.

1 Introduction

Fine-tuning pre-trained language models (PLMs) (Devlin et al., 2019; Radford et al., 2019; Raffel et al., 2020) has made great progress in recent years. By tuning the entire model parameters, the versatile knowledge acquired from large-scale unlabeled corpora can be adapted to handling various NLP tasks and outperform the approach of learning models from scratch (Han et al., 2021a). For simplicity, we name this full-model tuning as “FT”. As shown in Figure 1 (b) and (c), there are two mainstream FT approaches. The first one is task-oriented fine-tuning, where a task-specific head is added on top of PLMs, and the entire model is then fine-tuned by optimizing task-specific objectives on corresponding training data.

The second one is prompt-oriented fine-tuning (Schick and Schütze, 2021a), which is inspired by the recent works utilizing language prompts to probe the knowledge in PLMs (Petroni et al., 2019; Brown et al., 2020). In prompt-oriented fine-tuning, data samples are converted to sequences containing prompt tokens, and downstream tasks are formalized as language modeling problems. As shown in Figure 1 (c), by adding the prompt “It was ⟨X⟩.” to a sentence, we can determine its sentiment polarity with PLMs by predicting “great” or “terrible” at the mask position. As shown in Figure 1, compared to task-oriented fine-tuning, prompt-oriented fine-tuning is more similar to the pre-training objectives (masked language modeling), thereby helping to better use knowledge in PLMs and often obtaining better performance.

Although the FT methods have shown promising results, with the rapid growth of model scale, fine-tuning and storing the entire large model for each downstream task becomes more and more expensive. To address this challenge, Lester et al. (2021) propose prompt tuning (PT) to adapt large PLMs to downstream tasks cheaply, as shown in Figure 1 (d). Specifically, PT uses soft prompts composed of continuous embeddings instead of hard prompts (discrete language phrases). These continuous prompt embeddings are generally randomly initialized and learned end-to-end. To avoid storing the entire model for each downstream task, PT freezes all PLM parameters and merely tunes soft prompts, without adding any intermediate layers and task-specific components.

PT has two promising advantages. First, soft prompts can be learned end-to-end in comparison to hard prompts. Second, PT is an efficient and effective paradigm for the practical use of large-
I like eating apples.  <X>

Figure 1: Paradigms of pre-training (masked language modeling), full-model tuning (task-oriented fine-tuning and prompt-oriented fine-tuning), and prompt tuning. The verbalizer is a function to map task labels to concrete words.

Figure 2: Comparison between PT and FT. The tuned prompt is composed of 100 learnable embeddings whose dimensions are the same as the token embeddings of PLMs (4096 dimensions). All these results are based on 11B PLMs T5 and CPM-2. FT needs to optimize all 11B parameters, while PT only trains about 410K prompt parameters.

scale PLMs, which is comparable to FT when downstream data are sufficient (Figure 2(a)). However, as shown in Figure 2(b), we find that PT performs much worse than FT under few-shot settings, which may hinder the application of PT in various low-resource scenarios.

Hence, in this paper, we explore how to use PLMs for few-shot learning in an efficient and effective manner through PT. Specifically, we conduct pilot experiments to empirically analyze the effectiveness of PT on large-scale PLMs in Section 2, which is ignored by most existing works. Our discoveries are as follows: (1) the verbalizer choice has a large impact on the performance; (2) simply initializing soft prompts with concrete word embeddings fails to improve the performance, yet (3) combining soft and hard prompts is helpful; and (4) all these methods cannot handle few-shot prompt tuning problems well. The above observations reveal that prompt searching for PLMs is not trivial, and carefully initialized soft prompt tokens is crucial.

To help the model find suitable prompts, we pretrain these tokens with self-supervised tasks on large-scale unlabeled corpora. To ensure the generalization of pre-trained prompts, we group typical classification tasks into three formats: sentence-pair classification, multiple-choice classification, and single-text classification, each format corresponding to one self-supervised pre-training task. In addition, we find multiple-choice classification more general among these formats and we can unify all classification tasks to this format. We name this Pre-trained Prompt Tuning framework “PPT”. We evaluate PPT on several datasets based on three 11B PLMs: T5-XXL (Raffel et al., 2020), mT5-XXL (Xue et al., 2021) and CPM-2 (Zhang et al., 2021b) in few-shot scenarios. Experiments show that PPT can not only improve PT by a large margin, reaching or even outperforming FT methods, but also reduce the variance of few-shot learning. Besides the effectiveness, PPT also retains the parameter efficiency of PT, which is valuable for future applications on large-scale PLMs.

2 Pilot Experiments

In this section, we present pilot experiments of PT for few-shot learning. We analyze three strategies including hybrid prompt tuning, verbalizer selection, and real word initialization. We follow Lester et al. (2021) to test PT with T5-XXL (11B parameters) and use 100 tunable soft prompt tokens1.

1Using 100 soft prompt tokens achieves the best performance in Lester et al. (2021).
Following Schick and Schütze (2021b), we randomly select 32 samples to construct the training set $D_{\text{train}}$ from the original training data. To tune the hyper-parameters, we compose a validation set $D_{\text{dev}}$ from the original training data and ensure $|D_{\text{train}}| = |D_{\text{dev}}|$ to simulate the few-shot learning setting (Perez et al., 2021). We follow Zhang et al. (2021a) and Gao et al. (2021) to use the original validation set as the test set $D_{\text{test}}$, which means $|D_{\text{test}}| \gg |D_{\text{train}}| = |D_{\text{dev}}|$. 

**Hybrid Prompt Tuning** In hybrid prompt tuning, both soft and hard prompts are used (Liu et al., 2021; Han et al., 2021b). However, previous works train soft prompts jointly with the entire model. In PT where only prompt tokens are tunable, the effectiveness of hybrid prompts is under-explored. In Table 1, we show the results of combining soft prompts $P$ with three manually designed hard prompts and two auto-generated hard prompts (Gao et al., 2021) on a sentiment classification task (Socher et al., 2013). We can see that hard prompts improve PT, but still under-perform FT. Furthermore, different hard prompts affect the performance remarkably, therefore much human labor for prompt design and selection is needed.

**Verbalizer Selection** Verbalizer maps task-specific labels to concrete tokens. For instance, in Figure 1 (c) and (d), the verbalizer maps the label “Positive” to “great”. From Table 1 we can see that the choices of verbalizers influence the performance remarkably. In general, common words that explain the meaning of corresponding labels work well. This also guides our verbalizer selection for PPT in Section 3.

**Real Word Initialization** In real word initialization, we use the embeddings of concrete words to initialize the soft prompt and test four initialization strategies. The effectiveness of this approach has been verified on small PLMs (fewer than 3B parameters) in previous works (Lester et al., 2021). However, from the experiments on SST-2 (Socher et al., 2013) and BoolQ (Clark et al., 2019) (Table 2), we find that for the 11B model, real word initialization has little or even negative impact on the performance in few-shot scenarios. This suggests that observations on small models can not be directly adapted to large models and finding a good initialization for soft prompts is yet to be explored.

To summarize, although the above enhancement strategies cannot help PT achieve comparable results with FT under few-shot settings, they are still the key factors that influence the PT performance. In the following sections, we describe our PPT framework and show in experiments that PPT not only provides a good prompt initialization, but also takes advantage of the good verbalizer, and is complementary to hybrid prompts.

### 3 Pre-trained Prompt Tuning (PPT)

In this section, we describe the whole framework of PPT, including how to pre-train prompts and use these pre-trained prompts for specific tasks.

#### 3.1 Overview

Following the approach of T5 (Raffel et al., 2020) and PT (Lester et al., 2021), we solve all down-stream learning tasks with good/bad and “Gen” means auto-generated hard prompts. The choices of verbalizers influence the performance remarkably. In general, common words that explain the meaning of corresponding labels work well. This also guides our verbalizer selection for PPT in Section 3.

#### Table 2: Few-shot learning performance with different strategies for choosing concrete words for prompt initialization in PT. “Label Init.”: use the embeddings of the label words. “Vocab Sampling”: randomly sample words from the vocabulary. “Top-1000 Sampling”: randomly sample words from the most frequent 1000 words in the pre-training corpus. “Task-Related”: randomly sample words from the downstream data. We use the classification accuracy (%) for evaluation.
stream tasks in a text-to-text format. As shown in Figure 1 (c), to reduce the objective gap between pre-training and downstream tasks, prompt-oriented fine-tuning converts downstream tasks into cloze-style objectives. Taking classification for example, given an input sentence \( x \in \mathcal{V}^n \) and its label \( y \in \mathcal{Y} \), a pattern mapping \( f : \mathcal{V}^n \mapsto \mathcal{V}^n \) is first applied to convert \( x \) into a new sequence \( f(x) \), where \( \mathcal{V} \) is the vocabulary of PLMs. \( f(x) \) not only adds some prompt tokens as hints, but also preserves the mask token \( \langle X \rangle \) to let PLMs predict tokens at the masked positions. Then, a verbalizer \( v : \mathcal{Y} \mapsto \mathcal{V}_p \) is used to map \( y \) to some label tokens \( v(y) \). With \( f(\cdot) \) and \( v(\cdot) \), a classification task can be represented by a pattern-verbalizer pair \( (f, v) \):

\[
\arg \max_{\theta} \sum_x \log p(y|x; \theta) = \arg \max_{\theta} \sum_x \log p(\langle X \rangle = v(y)|f(x); \theta), \tag{1}
\]

where \( \theta \) indicates all tunable parameters, especially the parameters of PLMs. For convenience, we use “PVP” to denote this pattern-verbalizer pair (Schick and Schütze, 2021a).

In PT (Lester et al., 2021), a set of soft prompts \( P \) are concatenated to the beginning of the sequence and the model input becomes \( [P; f(x)] \), where \([\cdot; \cdot]\) is the concatenation operation. By tuning \( P \), Eq. (1) is replaced by

\[
\arg \max_{P} \sum_x \log p(\langle X \rangle = v(y) | [P; f(x)]; P). \tag{2}
\]

Owing to the power of large-scale PLMs, Eq. (2) is verified to be comparable to these FT methods under full-data settings. However, we find it hard to learn effective soft prompts, which may result in low performance in various few-shot scenarios. The parameter initialization usually has a large impact on the difficulty of the model training and optimization, and our pilot experiments have shown that existing initialization strategies have little or even negative impact on the PT performance of large-scale PLMs. We refer more details of these pilot experiments to Section 4.

Recently, pre-training has been proven to be an effective method to find a good model initialization. Inspired by this, we propose to pre-train soft prompts. We notice that some groups of downstream tasks are related to certain self-supervised tasks built on unlabeled pre-training corpora. For instance, some tasks in the form of sentence-pair classification, such as natural language inference and sentence similarity, are similar to the next sentence prediction (NSP) (Devlin et al., 2019) task used in the pre-training stage. As shown in Figure 3, these tasks all take two sentences as input and compare their semantic meanings. Therefore, soft prompts pre-trained by NSP can be a good initialization for these sentence-pair tasks.

Formally, suppose we can divide downstream tasks into \( m \) groups \( \{T_1, T_2, ..., T_m\} \), where \( T_i \) is the set containing \( n_i \) downstream tasks: \( \{PVP_{PVP}^{1}, PVP_{PVP}^{2}, ..., PVP_{PVP}^{n_i}\} \), where \( PVP_{PVP}^{i} = (f_{PVP}^{i}, v_{PVP}^{i}) \). For each group, we design a corresponding pre-training task \( PVP_{PVP}^{pre} = (f_{pre}^{i}, v_{pre}^{i}) \). After pre-training soft prompts on these tasks with all model parameters fixed, we get \( m \) pre-trained prompts \( \{P_{1}, P_{2}, ..., P_{m}\} \). Then, for each task \( PVP_{PVP}^{i} \) in \( T_i \), we continue to optimize Eq. (2) by using \( P_{i} \) as the soft prompts initialization.

### 3.2 Designing Pattern-Verbalizer Pairs for Pre-training

In this section, we take three typical classification tasks as examples to describe the design of pattern-verbalizer pairs \( PVP_{PVP}^{pre} \) for prompt pre-training.

#### 3.2.1 Sentence-Pair Classification

Sentence-pair classification tasks such as natural language inference and sentence similarity take two sentences \( x = (s_1, s_2) \) as the input. To design a PVP for these tasks, we extend the next sentence prediction in Devlin et al. (2019) to a 3-class classification with labels \( \mathcal{Y} = \{0, 1, 2\} \) as the pre-training task. These labels in \( \mathcal{Y} \) can respectively indicate that the semantic relation between two sentences is coherent (with label 2), similar (1) and irrelevant (0). To construct signal from unlabeled documents, we set the two sentences next to each other as label 2, those from the same doc-
284 ugment but not true next sentence as 1, and those from different document as 0. We consider the label set $|Y| \leq 3$ because this covers most sentence pair tasks. $\text{PVP}_i^{\text{pre}} = (f_i^{\text{pre}}, v_i^{\text{pre}})$ is given as

$$
\begin{align*}
    f_i^{\text{pre}}(x) &= "s_1(X), s_2", \\
    v_i^{\text{pre}}(Y) &= [\text{no}, \text{maybe}, \text{yes}].
\end{align*}
$$

Designing $\text{PVP}_i^k = (f_i^k, v_i^k)$ according to $\text{PVP}_i^{\text{pre}}$ is simple. $s_1$ and $s_2$ can be replaced by the input sentence pair. If a task outputs two labels, then we take $v_i^k(Y) = [\text{no}, \text{yes}]$. If a task outputs three labels, we set $v_i^k = v_i^{\text{pre}}$. If a task requires to measure the similarity between two sentences, the probability over $\{\text{no}, \text{yes}\}$ can serve for this task.

### 3.2.2 Multiple-Choice Classification

Many tasks can be formulated as multiple-choice classification, which takes a query and several answer candidates as the input. We design a next sentence selection task to pre-train the prompt. Given a sentence as the query $s_q$, the model is trained to select the adjacent sentence from six candidates, denoted as $s_1 \sim s_6$ and thus the label set is $Y = \{1, 2, 3, 4, 5, 6\}$. These candidates consist of the right answer, one sentence from the same document but not adjacent to the query, and four sentences from other documents. For $x = (s_q, s_1, s_2, \cdots, s_6)$, $(f_i^{\text{pre}}, v_i^{\text{pre}})$ is given as

$$
\begin{align*}
    f_i^{\text{pre}}(x) &= "s_i? A.s_1 \cdots F.s_6, Answer is (X). ", \\
    v_i^{\text{pre}}(Y) &= [A, B, C, D, E, F].
\end{align*}
$$

Most multiple-choice tasks can use $(f_i^{\text{pre}}, v_i^{\text{pre}})$ directly as their PVPs. For tasks like reading comprehension, the input may contain a passage and a question. We concatenate them to form the query.

### 3.2.3 Single-Sentence Classification

For single-sentence classification, we create pseudo labels for prompt pre-training. Taking sentiment classification as an example, we use another small model to annotate sentiment labels for the sentences from the pre-training corpus and filter out those with low classification probability. In practice, we use a RoBERTa\textsubscript{BASE} (Liu et al., 2019) model fine-tuned on a 5-class sentiment classification dataset other than the few-shot datasets we evaluate on. Then with a sentence $s$ from the corpus, we have the input $x = (s)$ and the label set $Y = \{1, 2, 3, 4, 5\}$. $(f_i^{\text{pre}}, v_i^{\text{pre}})$ is given as

$$
\begin{align*}
    f_i^{\text{pre}}(x) &= "s. , ", \\
    v_i^{\text{pre}}(Y) &= [\text{terrible}, \text{bad}, \text{maybe}, \text{good}, \text{great}].
\end{align*}
$$

For sentiment classification tasks with 5 labels, we can use $\text{PVP}_i^k = \text{PVP}_i^{\text{pre}}$. For those with fewer than 5 labels, we choose a subset from $v_i^{\text{pre}}(Y)$ as labels.

Although the above method improves the model performance, we have to point out that it is still limited to generalize to other single-text classifications in different domains and with different numbers of labels. Therefore, the method described in the following section is proposed to solve this problem.

### 3.3 Unifying Task Formats

The above-mentioned PVPs for pre-training can be unified to a single format: multiple-choice classification. Specifically, for sentence-pair classification, the query is the concatenation of the two sentences and there are three options: no, maybe, and yes. For single-sentence classification, the query is the input sentence and the options are the concrete labels. Note that in this way, the pre-trained PVPs can be used in single text classification tasks from arbitrary domains and with much more labels.

Constructing a unified PVP is similar to the idea of MultiQA (Talmor and Berant, 2019) and UnifiedQA (Khashabi et al., 2020). Recently, Zhong et al. (2021a) use some hard prompts to unify several tasks as a meta question answering task. They tune the entire model with this meta task on a collection of QA datasets and then transfer to other classification tasks under low-resource settings. However, our PPT focuses on tuning soft prompts with the main body of PLMs fixed and our pre-training is conducted on fully unsupervised data, rather than the collection of supervised datasets.

Since different tasks may have different candidate numbers and lengths, we construct pre-training samples with option numbers varying from 2 to 16\(^2\) and option lengths from 50 to 20. We use the PVP in Section 3.2.2 for pre-training, and then apply pre-trained soft prompts to cover the above mentioned three classification tasks.

### 4 Experiments

#### 4.1 Setup

We conduct experiments on both Chinese and English tasks (see Table 3). As described in Section 2, for tasks with fewer than 5 labels, we construct $D_{\text{train}}$ and $D_{\text{dev}}$ with 32 samples from the original training data and ensure the number of labels is balanced. For tasks with more than 5 labels like

\(^2\)We set 16 labels in this paper as they can cover most benchmarks, but more labels are applicable for other tasks.
TNews and YahooAnswer, it is hard to compose a dataset with label-balanced samples. Therefore, we randomly select 8 samples for each label.

For English datasets, we conduct PT based on T5-XXL with 11B parameters because previous works (Lester et al., 2021; Zhang et al., 2021b) have shown that, T5-XXL is comparable with FT under the full-data setting. We also evaluate PT on various sizes of T5 to verify that larger models perform better and thus improving PT based on T5-XXL is meaningful. For Chinese datasets, we do PT based on a 11B model CPM-2. Since CPM-2 does not provide other size models, we compare it with mT5 (Xue et al., 2021) of various sizes.

Consistently, we use 100 soft tokens for PT. As a result, the tunable parameters is only $100 \times 4096 = 4.1 \times 10^5 = 410K$. Compared with the 11B ($1.1 \times 10^{10}$) parameters of FT, PT only needs to store 3000 times smaller parameters for each task.

### 4.2 Main Results

The main results of English and Chinese datasets are shown in Table 4. In the block FT, we present the FT results of the T5 model from the size small to XXL. In the block PT, we show the results of PPT and other baselines. The first baseline is Vanilla PT, where the soft prompts are randomly initialized from a normal distribution. The second is the hybrid strategy in Section 2. We also consider LM Adaption used in Lester et al. (2021) in which the T5 model is further pre-trained for 10K steps with language modeling to reduce the gap between the pre-training and PT. We test two variants of PPT: Hybrid PPT, in which carefully designed hard prompts are combined with pre-trained soft prompt, and Unified PPT, in which all tasks are unified in the multiple-choice classification format.

**Effectiveness** From the Table 4 we have four observations. First, larger models achieve better overall performance, which means increasing the model size still help under the few-shot setting. Therefore, we study PT on the large-scale pre-trained model. Note that for Chinese experiments, CPM-2 and mT5-XXL share the same parameter scale. Since CPM-2 outperforms mT5-XXL across all tasks, we use CPM-2 as the base model.

Second, PPT outperforms Vanilla PT and LM Adaption on most datasets significantly. Although PPT is worse than Hybrid PT on BoolQ, combining PPT and hard prompts (Hybrid PPT) outperforms all baselines. This means pre-training soft prompts and using hybrid prompts are complementary. Similar phenomenons are observed on other datasets like RACE-m, LCQMC, and C$^3$, where adding hard prompts to PPT continues to improve results.

Third, PPT outperforms FT on all Chinese datasets and most English datasets. This indicates that there still remains a gap between masked language modeling and downstream tasks. Prompt pre-training bridges this gap to some extend. Based on this observation, an intuitive extension of our method is to further pre-train the entire model with PVP$^{\text{pre}}$ and fine-tune the model to the corresponding downstream tasks. However, since we focus on PT in this paper, we leave this as future work.

Fourth, PPT results in lower variances on most of the datasets. Few-shot learning is notorious for its instability, which becomes very obvious in Vanilla PT. For some datasets like SST-2, the variance reaches 15.5 which means the model does not perform better than random guesses under some random seeds. Combining with hard prompt or further pre-training with language modeling can alleviate this problem to some extent. But on some datasets like CCPM, Hybrid PT increases the variance and LM Adaption does not guarantee the average performance. With the help of pre-training, the variance remains at a low level across all datasets.

**Unified PPT** Unifying all formats to multiple-choice classification format is another variant of PPT. In Table 4, we can see that Unified PPT reaches comparable performance as PPT and Hybrid PPT, still outperforming other PT baselines. However, the datasets we have considered so far have no more than 5 labels. For tasks with more labels, especially single-text classification where

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Format</th>
<th>$n_{\text{class}}$</th>
<th>Dataset</th>
<th>Format</th>
<th>$n_{\text{class}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2</td>
<td>SSC</td>
<td>2</td>
<td>ChnSent</td>
<td>SC</td>
<td>2</td>
</tr>
<tr>
<td>SST-5</td>
<td>SSC</td>
<td>5</td>
<td>Amazon</td>
<td>SC</td>
<td>5</td>
</tr>
<tr>
<td>YahooAns</td>
<td>SSC</td>
<td>10</td>
<td>TNews</td>
<td>SC</td>
<td>14</td>
</tr>
<tr>
<td>RACE-m</td>
<td>MCC</td>
<td>4</td>
<td>CPM</td>
<td>MC</td>
<td>4</td>
</tr>
<tr>
<td>RACE-h</td>
<td>MCC</td>
<td>4</td>
<td>C$^3$</td>
<td>MC</td>
<td>4</td>
</tr>
<tr>
<td>BoolQ</td>
<td>SPC</td>
<td>3</td>
<td>LCQMC</td>
<td>SPC</td>
<td>3</td>
</tr>
<tr>
<td>RTE</td>
<td>SPC</td>
<td>3</td>
<td>CMNLI</td>
<td>SPC</td>
<td>3</td>
</tr>
<tr>
<td>CB</td>
<td>SPC</td>
<td>3</td>
<td>OCNILI</td>
<td>SPC</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3: The datasets we evaluate. The “Format” column means the task category. SSC stands for single-sentence classification, MCC for multiple-choice classification, and SPC for sentence-pair classification. $n_{\text{class}}$ means the label number of each dataset.
Table 4: Classification results. The experiments are conducted with 32 training samples and 32 validation samples on each dataset. FT means full-model tuning, where the entire model (with about 11B parameters) should be tuned on each dataset. PT means prompt tuning, where only 410K parameters are trained. We report the mean and the standard deviation over 5 random seeds. The score marked as bold means the best performance among all the methods. The score marked with an underline means the best one among prompt tuning (PT) methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>SST-2 Acc.</th>
<th>SST-5 Acc.</th>
<th>RACE-m Acc.</th>
<th>RACE-h Acc.</th>
<th>BoolQ Acc.</th>
<th>RTE Acc.</th>
<th>CB F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT (11B)</td>
<td>mT5-Small</td>
<td>-</td>
<td>72.8 ± 1</td>
<td>31.1 ± 4</td>
<td>26.4 ± 6</td>
<td>26.3 ± 5</td>
<td>59.2 ± 6</td>
<td>54.0 ± 7</td>
</tr>
<tr>
<td></td>
<td>mT5-Base</td>
<td>-</td>
<td>74.6 ± 7</td>
<td>28.5 ± 8</td>
<td>27.2 ± 5</td>
<td>26.7 ± 2</td>
<td>61.9 ± 1</td>
<td>56.1 ± 3</td>
</tr>
<tr>
<td></td>
<td>mT5-Large</td>
<td>-</td>
<td>89.1 ± 2</td>
<td>42.4 ± 2</td>
<td>48.2 ± 1</td>
<td>43.2 ± 1</td>
<td>74.6 ± 9</td>
<td>64.4 ± 3</td>
</tr>
<tr>
<td></td>
<td>T5-XL</td>
<td>-</td>
<td>89.6 ± 3</td>
<td>38.4 ± 1</td>
<td>55.0 ± 8</td>
<td>50.9 ± 2</td>
<td>77.2 ± 1</td>
<td>62.3 ± 8</td>
</tr>
<tr>
<td></td>
<td>T5-XXL</td>
<td>-</td>
<td>91.4 ± 8</td>
<td>40.9 ± 0</td>
<td>62.9 ± 9</td>
<td>54.8 ± 0</td>
<td>80.8 ± 4</td>
<td>64.1 ± 2</td>
</tr>
<tr>
<td>PT (410K)</td>
<td>T5-XXL</td>
<td>Vanilla PT</td>
<td>70.5 ± 5</td>
<td>32.3 ± 8</td>
<td>34.7 ± 2</td>
<td>31.6 ± 3</td>
<td>61.0 ± 3</td>
<td>53.5 ± 5</td>
</tr>
<tr>
<td></td>
<td>Hybrid PT</td>
<td>87.6 ± 4</td>
<td>40.9 ± 7</td>
<td>53.8 ± 2</td>
<td>44.2 ± 4</td>
<td>79.8 ± 1</td>
<td>56.8 ± 6</td>
<td>66.5 ± 2</td>
</tr>
<tr>
<td></td>
<td>LM Adaptation</td>
<td>77.6 ± 5</td>
<td>36.2 ± 6</td>
<td>27.3 ± 2</td>
<td>26.5 ± 4</td>
<td>62.0 ± 3</td>
<td>55.3 ± 0</td>
<td>61.2 ± 7</td>
</tr>
<tr>
<td></td>
<td>PPT</td>
<td>93.5 ± 3</td>
<td>50.2 ± 7</td>
<td>60.0 ± 1</td>
<td>53.0 ± 0</td>
<td>66.4 ± 5</td>
<td>58.9 ± 6</td>
<td>71.2 ± 2</td>
</tr>
<tr>
<td></td>
<td>Hybrid PPT</td>
<td>93.8 ± 1</td>
<td>50.1 ± 5</td>
<td>62.5 ± 9</td>
<td>52.2 ± 7</td>
<td><strong>82.0 ± 0</strong></td>
<td>59.8 ± 3</td>
<td>73.7 ± 0</td>
</tr>
<tr>
<td></td>
<td>Unified PPT</td>
<td>94.4 ± 3</td>
<td>46.0 ± 1</td>
<td>58.0 ± 9</td>
<td>49.9 ± 1</td>
<td>76.0 ± 7</td>
<td><strong>65.8 ± 2</strong></td>
<td>82.2 ± 4</td>
</tr>
</tbody>
</table>

Table 5: The experiments on single-text classification tasks with more than 5 labels. Different from previous experiments, we randomly select 8 samples for each label. PT (MC) means doing PT in a multiple-choice format without prompt pre-training. We do not use PPT for single-sentence classification discussed in Section 3.2.3 because it is hard to find other suitable datasets to train the pseudo label annotator. However, we can see that Unified PPT still achieves the best performance, even exceeding FT by a large margin.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>TNews Acc.</th>
<th>YahooAns Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT (11B)</td>
<td>mT5-Small</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>mT5-Base</td>
<td>43.2 ± 6</td>
<td>64.1 ± 9</td>
</tr>
<tr>
<td></td>
<td>mT5-Large</td>
<td>41.2 ± 2</td>
<td>62.0 ± 4</td>
</tr>
<tr>
<td></td>
<td>T5-XL</td>
<td>11.8 ± 2</td>
<td>60.8 ± 9</td>
</tr>
<tr>
<td></td>
<td>T5-XXL</td>
<td><strong>50.6 ± 7</strong></td>
<td><strong>70.5 ± 9</strong></td>
</tr>
</tbody>
</table>

4.3 Sample Efficiency
We discuss how the performance of FT, PT, and PPT varies when the number of training samples increases. In Figure 4, we show the trend of these methods on the RACE-m and CB datasets. For 32 to 128 samples, PPT is consistently better than PT, and the performances of the three methods gradually converge when the number grows to 256.

5 Related Works
PLMs and Task-oriented Fine-tuning Recently, various powerful PLMs have been proposed,
Figure 4: Comparison between FT, Vanilla PT, and PPT when different numbers of training samples are available. For the small number of samples, PPT is consistently better than Vanilla PT. When the number grows, the performance of these methods becomes closer.

works still restrict auto-generated prompts to discrete spaces which are usually sub-optimal.

To overcome the shortcomings of discrete spaces, Li and Liang (2021); Liu et al. (2021); Han et al. (2021b); Hambardzumyan et al. (2021); Zhong et al. (2021b) explore to combine hard prompts and soft prompts. Different from hard prompts using concrete and discrete tokens, soft prompts are composed of several continuous learnable embeddings, and these embeddings are randomly initialized. To step forward, some works (Li and Liang, 2021; Qin and Eisner, 2021; Lester et al., 2021) propose to only tune soft prompts and fix the entire PLM parameters. When models are large enough, this method can be comparable to full-model tuning.

**Few-shot Learning with PLMs** Since long-tail distribution is common in real-world applications, few-shot learning is quite meaningful for the stable and effective use of PLMs, thereby attracts much attention recently. Apart from GPT-3 (Brown et al., 2020) and PET (Schick and Schütze, 2021a) which demonstrates the superiority of PLMs in few-shot scenarios, some later works Perez et al. (2021); Bragg et al. (2021) also discuss reasonable few-shot settings by restricting the size of validation set and proposing a unified framework to evaluate few-shot performance. There is also work (IV et al., 2021) pointing out the low performance of PT for few-shot learning. But they mostly focus on PLMs with fewer than 400M parameters. In this paper, we study few-shot learning on large-scale 11B PLMs.

6 Conclusion

In this paper, we present PPT, a framework that improves prompt tuning for few-shot learning. We propose to firstly unify downstream tasks to several formats. Then, we design self-supervised pre-training tasks for each format and pre-train prompts on these tasks. Finally, we do prompt tuning on downstream tasks based on the initialization of the corresponding pre-trained prompts. Extensive experiments show that our method significantly outperforms other prompt tuning baselines, performing comparable or even better than full-model tuning. There are two important directions for future work: (1) Designing unified task formats and the corresponding pre-training objectives for other kinds of tasks such as language generation and relation extraction. (2) Beyond the soft prompt, whether unified task pre-training helps the pre-trained language models itself.
References


Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? Transaction of TACL.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single qa system. In Findings of EMNLP.


Timo Schick and Hinrich Schütze. 2021b. It’s not just size that matters: Small language models are also few-shot learners. In Proceedings of NAACL-HLT.


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 EMNLP.


Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021b. Factual probing is [mask]: Learning vs. learning to recall. In Proceedings of NAACL-HTL.
Appendices

A Dataset Information

Since some of the test sets of the datasets we used is not publicly available, we follow Zhang et al. (2021) and Gao et al. (2021) to use original validation sets for testing. For English experiments, we use a dataset from GLUE (Wang et al., 2019b) (SST-2 (Socher et al., 2013)), datasets from SuperGLUE (Wang et al., 2019a), (BoolQ (Clark et al., 2019), CB (De Marneffe et al., 2019), and RTE (Dagan et al., 2006)), two extra single-text classification datasets (SST-5 (Socher et al., 2013) and YahooAnswers (Zhang et al., 2015)), and two standard question answering datasets (RACE-middle and RACE-high) (Lai et al., 2017) for multiple-choice classification. For Chinese experiments, we use four datasets from CLUE (Xu et al., 2020) (CMNLI1, OCNLI (Hu et al., 2020), TNews1, C3 (Sun et al., 2020)), two sentiment analysis datasets (ChnSent2 and Amazon Reviews3), and one extra natural language inference dataset LCQMC (Liu et al., 2018).

B PVPs for Chinese Tasks

We describe the PVP\textsuperscript{pre} for Chinese datasets in this section. Just like English scenarios, all these PVPs are simple and intuitive.

Sentence-Pair Classification Given the input \(x = (s_1, s_2)\), the label list \(\mathcal{Y} = \{0, 1, 2\}\), we have:
\[
\begin{align*}
 f_1^{\text{pre}}(x) &= \langle s_1 \rangle \cdot \langle s_2 \rangle, \\
 v_1^{\text{pre}}(\mathcal{Y}) &= [\text{矛盾, 中立, 相似}].
\end{align*}
\]

Multiple-Choice Classification Given a input \(x\) consisting of a query and six candidates: \(x = (s_q, s_1, s_2, \cdots, s_6)\), we convert \(x\) to a language sequence by defining the PVP\textsuperscript{pre} as follows:
\[
\begin{align*}
 f_1^{\text{pre}}(x) &= \langle s_q \rangle \cdot \langle s_1, s_2, \cdots, s_6 \rangle, \\
 v_1^{\text{pre}}(\mathcal{Y}) &= [-, +, +, +, +, +].
\end{align*}
\]

Single-Sentence Classification Similar to the English scenario, we take sentiment classification as an example. Given the input \(x = (s)\), we have:
\[
\begin{align*}
 f_1^{\text{pre}}(x) &= \langle s \rangle \cdot \langle \rangle, \\
 v_1^{\text{pre}}(\mathcal{Y}) &= [\text{差, 不好, 一般, 好, 赞}].
\end{align*}
\]

Based on the PVP\textsuperscript{pre}, the design of PVP\textsuperscript{\(i\)} is similar to that of English tasks.

---

**Table 1:** The hard prompts for Hybrid PT and Hybrid PPT. SSC stands for single-sentence classification, MCC stands for multiple-choice classification, and SPC stands for sentence-pair classification.

![Table 1](https://www.cluebenchmarks.com/)

![Table 2](https://github.com/SophonPlus/ChineseNlpCorpus/)

C Training Details

Considering the instability of the few-shot learning, we run each experiment 5 times on the random seed [10, 20, 30, 40, 50] and report the averaged performance as well as the standard deviation. Due to the resource limit, for 11B models, we adopt model parallelism (Shoeybi et al., 2019) and store a model with 4 GPU devices. We also use mixed-precision training (Micikevicius et al., 2018) and ZeRO (Rajbhandari et al., 2020) stage-1 provided in DeepSpeed (Rasley et al., 2020) to reduce GPU memory usage. For models in other sizes, we all use full-precision training. We describe the details of the training hyper-parameters in the following sections.

C.1 Full-Model Tuning

For Full-Model Tuning (FT), we tune the entire parameters of the model without concatenating soft prompts. For all models, we fix the batch size as 16. In this way, we train the largest 11B model with 16 NVIDIA V100 32G GPUs. We find that different sized models prefer significantly different learning rates. Therefore, we search for the learning rates in varied intervals and show each model size and its corresponding searching interval in Table 2. We train the model for 50 epochs and do evaluation every 6 optimization steps. We choose the model performing the best on the validation set and evaluate it on the test set.

C.2 Prompt Tuning

For Prompt Tuning (PT), we add a set of soft prompts before the input text. When adapting the model to downstream tasks, we only tune the soft prompts with the entire model fixed. Similar to FT, we fix the batch size as 16 and train the model for 50 epochs, while evaluating the model every 6
<table>
<thead>
<tr>
<th>Model Size</th>
<th>Searching Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>2e-4, 5e-4, 1e-3</td>
</tr>
<tr>
<td>Base</td>
<td>2e-4, 5e-4, 1e-3</td>
</tr>
<tr>
<td>Large</td>
<td>5e-5, 1e-4, 2e-4</td>
</tr>
<tr>
<td>XL</td>
<td>3e-5, 5e-5, 1e-4</td>
</tr>
<tr>
<td>XXL</td>
<td>3e-6, 5e-6, 1e-5</td>
</tr>
</tbody>
</table>

Table 2: The searching intervals of learning rates for the models with different sizes. Generally, small models prefer large learning rates.

steps. Since the tunable parameters are much less in PT, 8 NVIDIA V100 32G GPUs are enough for the training. We find PT requires a much larger learning rate than FT. Therefore, we search for the learning rate in [5e-3, 1e-2, 2e-2, 5e-2] and choose the model with the best performance on the validation set. This observation also implies that PT is much harder to train than FT, which is consistent with the experiment results in the main paper.

D Hard Prompts

In this section, we describe the hard prompts we use in Hybrid PT and Hybrid PPT. For simplicity, we choose the best hard prompts for each task format (e.g., sentence-pair classification, multiple-choice classification, and single-sentence classification) based on PT in pilot experiments and directly use them in Hybrid PPT. The prompts corresponding to each task format are shown in Table 1.

References


Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence Moss. 2020. OCNLI: Original Chinese Natural Language Inference. In Findings of EMNLP.

Guokun Lai, Qi zhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale Reading comprehension dataset from examinations. In Proceedings of EMNLP.


Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In Proceedings of KDD.


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 EMNLP.