Towards Unified Prompt Tuning for Few-shot Learning

Anonymous ACL submission

Abstract

Prompt-based fine-tuning has boosted the performance of Pre-trained Language Models (PLMs) on few-shot learning by employing task-specific prompts. However, PLMs are unfamiliar with the prompt-style expressions during pre-training, which limits the few-shot learning performance on downstream tasks. It would be desirable if models can acquire some prompting knowledge before task adaptation. We present the *Unified Prompt Tuning* (*UPT*) framework, leading to better few-shot learning for BERT-style models by explicitly capturing prompting semantics from non-target NLP datasets. In *UPT*, a novel paradigm *Prompt-Options-Verbalizer* is proposed for joint prompt learning across different NLP tasks, forcing PLMs to capture task-invariant prompting knowledge. We further design a self-supervised task named *Knowledge-enhanced Selective Masked Language Modeling* to improve the PLM's generalization abilities for accurate adaptation to previously unseen tasks. After multi-task learning, the PLM can be fine-tuned for any target few-shot NLP tasks using the same prompting paradigm. Experiments over a variety of NLP tasks show that *UPT* consistently outperforms state-of-the-arts for prompt-based fine-tuning. [1]

1 Introduction

The emergence of Pre-trained Language Models (PLMs) has boosted the performance of a variety of NLP tasks (Qiu et al., 2020; Han et al., 2021a). However, during fine-tuning, PLMs can perform poorly with few training samples due to model over-fitting (Gao et al., 2021).

To alleviate this problem for low-resourced scenarios, natural language prompts have been applied to enable few-shot or zero-shot learning with PLMs (Liu et al., 2021a). To make prompts more flexible and task-adaptive, *prompt tuning* freezes the PLM backbone and adjusts the representations of prompts (Lester et al., 2021). This type of methods is especially suitable for ultra-large PLMs that are difficult to tune. For BERT-style PLMs, *prompt-based fine-tuning* has been proposed, transforming most NLP tasks into cloze-style problems (Schick and Schütze, 2021a,b; Gao et al., 2021). To specify, task-specific prompt templates, together with token masks, are added to input texts. The result tokens of the masked positions predicted by the Masked Language Modeling (MLM) head of the model are used for class label prediction. [2] Therefore, the pre-trained knowledge acquired by PLMs can be better utilized by "re-using" the MLM training objective. Witnessing the successful usage of prompts for few-shot learning, various following-up works have been conducted, such as continuous prompt encoding (Liu et al., 2021c), knowledgeable prompt learning (Hu et al., 2021) and prompt generation (Shin et al., 2020).

Recently, a few works (Wei et al., 2021; Zhong et al., 2021a; Mishra et al., 2021) focus on multi-task prompt tuning on ultra-large PLMs. Specifically, they tune PLMs on full training samples from different tasks to force PLMs learn more prompting knowledge, and directly make predictions over the target task by zero-shot learning. Yet, we observe that for BERT-style small PLMs, the performance is not satisfactory for two reasons. 1) These PLMs are sensitive to different designs of prompt templates and verbalizers (Liu et al., 2021c), which fail to adapt to target tasks with new prompts and verbalizers. 2) There are word distribution differences between prompt-style texts and sentences in pre-training corpora. It would be better if the BERT-style PLMs can acquire some prompting

[1] All the datasets are publicly available. Source codes are provided in attachments and will be released upon acceptance.

[2] For example, in the review analysis task, given an input "It is a wonderful movie.", one can add the prompt template "Based on the review, it is [MASK]." to the input. The output of the masked token "great" and "terrible" can be mapped to the positive and negative class, respectively.

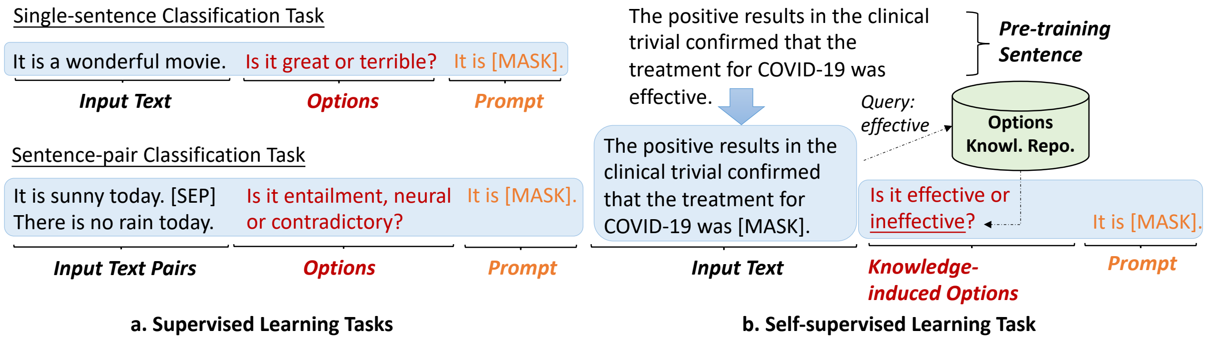

Figure 1: *UPT* is a unified framework that learns prompting knowledge from non-target NLP datasets to improve the performance on target tasks, in the format of *Prompt-Options-Verbalizer* (Sect. 2.2). Figures a) and b) show examples of supervised and the self-supervised learning task (i.e., *Knowledge-enhanced Selective MLM*, Sect. 2.3). (Best viewed in color.)

knowledge before they are adapted to downstream tasks. Therefore, a natural question arises: *how can we make BERT-style PLMs to adapt to target NLP tasks accurately with more prompting knowledge?*

To address these issues, we introduce a novel framework named *Unified Prompt Tuning* (*UPT*), facilitating better few-shot learning for BERT-style models by explicitly capturing general prompting semantics from non-target datasets. Specially, we propose a unified paradigm named *Prompt-Options-Verbalizer* (*POV*), which enables the mixture training of PLMs over a series of *non-target NLP tasks* of varied types. To further improve the model's generalization abilities on previously unseen tasks, the PLM is also jointly trained over a self-supervised task named *Knowledge-enhanced Selective MLM* (*KSMLM*), which mimics the behavior of MLM with explicit usage of prompts and background knowledge mined from massive corpora. After the multi-task training process is completed, the underlying PLM can be fine-tuned to fit any target few-shot NLP tasks using the same prompting paradigm.

In the experiments, we verify the effectiveness of *UPT* over 9 public NLP datasets of various tasks. Experimental results show that *UPT* consistently outperforms state-of-the-art approaches for prompt-based few-shot fine-tuning. Overall, the averaged accuracy is improved by over 4.56%.

In summary, we make the following major contributions:

- We introduce the novel *UPT* framework to improve prompt-based few-shot learning for BERT-style models. To our knowledge, our work is the first to leverage other NLP datasets to help these PLMs capture unified prompting semantics for few-shot learning on new tasks.
- In *UPT*, a new paradigm *POV* is proposed for joint prompt tuning across different NLP tasks. We further design the self-supervised *KSMLM* task to improve the PLM's generalization abilities for accurate task adaptation.
- Extensive experiments over 9 public NLP datasets show that *UPT* consistently outperforms state-of-the-arts for prompt-based few-shot fine-tuning by a relatively large margin.

2 *UPT*: The Proposed Framework

We start with a brief overview of the *UPT* framework, followed by its detailed techniques.

2.1 A Brief Overview of *UPT*

For clarity, we introduce some basic notations. Let $\mathcal{D}^*$ be the N-way-K-shot training set of a target NLP task $\mathcal{T}^*$. The underlying PLM is parameterized by Θ. The basic goal of few-shot learning is to obtain a high-performing model for $\mathcal{T}^*$ based on $\mathcal{D}^*$, with parameters initialized from Θ. As the size of $\mathcal{D}^*$ is only $N \times K$, the model performance would be highly limited. Here, we assume that there are M other NLP tasks that are *dissimilar* to $\mathcal{T}^*$, i.e., $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(M)}$, with their (usually non few-shot) training sets denoted as $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(M)}$, respectively. [3] The *UPT* framework seeks to explore how to employ $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(M)}$ to enhance the performance of the PLM on a new task (such as $\mathcal{T}^*$) based on its own few-shot training set $\mathcal{D}^*$.

Hence in *UPT*, the model is firstly trained over all the tasks $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(M)}$, aiming to learn the

[3]Note that we constrain that $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(M)}$ are dissimilar to $\mathcal{T}^*$ to deal with true low-resourced scenarios where no training sets of similar tasks are available. If $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(M)}$ are similar to $\mathcal{T}^*$, one can directly apply transfer learning techniques to train the model, which is considered a relatively trivial problem and not the major focus of this work.

semantics of prompts and the general methodology of solving downstream tasks by prompting. After that, it is prompt-tuned over a specific task $\mathcal{T}^*$. To unify the learning process, each training sample i in all different tasks (either $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(M)}$ or $\mathcal{T}^*$) is augmented in the same format, by means of the *Prompt-Options-Verbalizer* (*POV*) triple (P_i, O_i, V_i). Here, P_i is the prompt. O_i is the expression containing all possible options of the masked language token appearing in the prompt P_i (i.e., the collection of label words). V_i is the verbalizer that maps the target token predicted by the MLM head of the PLM to the class label. Readers can also refer to the examples of supervised learning tasks in Figure 1.

In addition, we observe that the diversity of label words in $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(M)}$ is limited. For previously unseen tasks, the optimization of these tasks alone often leads to a poorly generalized model that are biased towards these tasks. Therefore, we further introduce the self-supervised *Knowledge-enhanced Selective MLM* (*KSMLM*) task $\tilde{\mathcal{T}}$ as an auxiliary task, which takes pre-training sentences as inputs (with the self-supervised training set denoted as $\tilde{\mathcal{D}}$.). These sentences are selectively masked, with options generated by rich options knowledge mined from a massive corpus. An example is also shown in Figure 1. Hence, the model has a better generalization abilities and avoids catastrophic forgetting of the pre-training knowledge, before it is adapted to specific target tasks.

2.2 The Unified Prompting Paradigm

A fundamental challenge for prompt-based training across $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(M)}$ for BERT-style models is that different NLP tasks have diverse sets of label words w.r.t. masked language tokens. When dealing with a mixture of training samples, a naive solution is to build a unified output prediction space, consisting of candidate label words from all tasks. However, the enlarged output space makes it challenging for the PLM to optimize. Additionally, the output prediction space may not cover the label words of all possible unseen tasks.

Here, we propose a unified prompting paradigm that augments each sample i by a *Prompt-Options-Verbalizer* (*POV*) triple (P_i, O_i, V_i). P_i is the prompt that provides task guidance (in line with PET (Schick and Schütze, 2021a,b)). O_i is a fixed expression that explicitly provides selection for the model over all its candidate label words. To facilitate the fast adaptation of arbitrary tasks, the verbalizer V_i maps the output of the masked language token to the entire vocabulary $\mathcal{V}$.[4] We can see that the options are crucial as they give strong indications on the possible outputs of the PLM (i.e., the candidates). Overall, the output probability $q(v|i, P_i, O_i, \Theta)$ of the token $v \in \mathcal{V}$ w.r.t. the training sample i is computed as follows:

$$q(v|i, P_i, O_i, \Theta) = \frac{\exp(s(v|i, P_i, O_i, \Theta))}{\sum_{v' \in \mathcal{V}} \exp(s(v'|i, P_i, O_i, \Theta))}$$

where $s(v|i, P_i, O_i, \Theta)$ is the un-normalized score of the MLM head (before the softmax function) for generating token v at the position of the masked language token with i, P_i and O_i as inputs. Denote the entire prediction vector (of the length $|\mathcal{V}|$) as $Q(\mathcal{V}|i, P_i, O_i, \Theta)$. The *multi-task prompting loss* (denoted as $\mathcal{L}_{MP}$) can be written as follows:

$$\mathcal{L}_{MP} = -\sum_{i \in \mathcal{D}} P(\mathcal{V}|i, P_i, O_i, \Theta) \cdot \log Q(\mathcal{V}|i, P_i, O_i, \Theta)$$

where $\mathcal{D} = \bigcup_{k=1}^{M} \mathcal{D}^{(k)}$, and $P(\mathcal{V}|i, P_i, O_i, \Theta)$ is the one-hot ground-truth prediction vector.

In addition, we notice that $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(M)}$ can be arbitrary labeled datasets with varied sizes. Optimizing $\mathcal{L}_{MP}$ directly on their original datasets would make the few-shot learner more likely to be biased towards larger datasets. In our work, we do stratified sampling to form a batch where a training sample i from $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(M)}$ is picked with the probability proportional to its own dataset size (denoted as w_i), i.e., $w_i = \frac{\log|\mathcal{D}^{(k)}|+\gamma}{M \cdot \gamma + \sum_{k'=1}^{M} \log|\mathcal{D}^{(k')}|}$ where $\gamma > 0$ is a smoothing factor and $i \in \mathcal{D}^{(k)}$. Hence, we re-formulate our loss function $\mathcal{L}_{PT}$ as the *weighted multi-task prompting* loss $\mathcal{L}_{WMP}$:

$$\mathcal{L}_{WMP} = -\sum_{i \in \mathcal{D}} w_i \cdot P(\mathcal{V}|i, P_i, O_i, \Theta) \cdot \log Q(\mathcal{V}|i, P_i, O_i, \Theta)$$

2.3 Extending Unified Prompting to Self-supervised Learning

One drawback of the above approach is that the diversity of label words in these supervised learning tasks is usually limited, covering a narrow spectrum of the vocabulary $\mathcal{V}$. The model would not be well generalized for tasks with new label words.

[4] The list of prompts, options and verbalizers of all the tasks are given in the appendix.

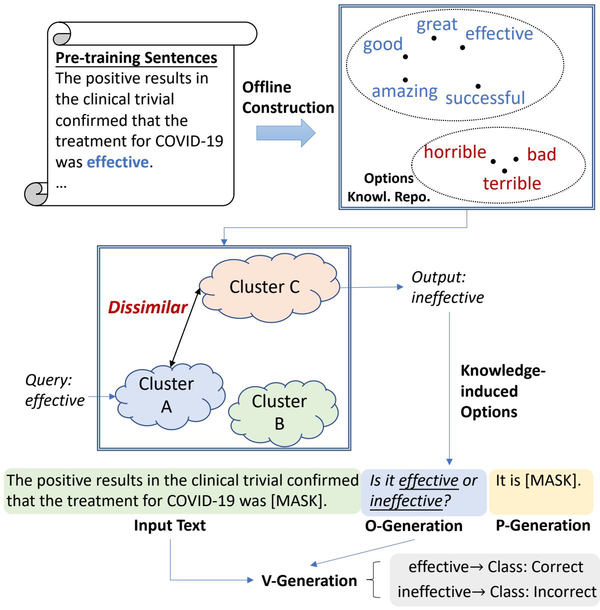

Figure 2: An illustrated example of the *POV* generation process for the *KSMLM* task.

Hence, we leverage the idea of MLM pre-training, formulated by the *POV* paradigm.

As a naive approach, given a sentence, we can randomly mask a word and generate the options of the correct and a randomly selected word, and then ask the model to make the prediction. Unfortunately, the seemingly feasible approach may ruin the training process, because not all words are suitable label words. For example, stop words and a large number of verbs and adverbs have not been used in any verbalizers in downstream tasks. The alternatives used in options should be reasonable, in order to make the model learn truly useful knowledge. To address the issue, we present the self-supervised *KSMLM* task, with an example shown in Figure 2. In the following, we describe the *POV* construction process for *KSMLM*. After that, the loss function of the task is given.

P-Generation. The prompt for all training samples for *KSMLM* is universal, which is fixed to be "It is `[MASK]`.". During training, the PLM is asked to predict the actual word of the masked position.

O-Generation. From Gao et al. (2021), we can see that most label words for language understanding tasks are adjectives (such as "great" and "terrible" for sentiment analysis). Thus in our work, we detect all adjectives in the pre-training corpus by part-of-speech tagging models and filter out low-frequency adjectives[5]. The adjectives are then clustered by K-Means, with their token representations generated from the underlying PLM as features. Formally, We construct a knowledge repository named *Options Knowledge Repository* (*OKR*), in the form of triples $\mathcal{R} = \{(v, \vec{v}, c_v)\}$, where v is a candidate label word. $\vec{v}$ and c_v denote the representation vector and the cluster membership of v, respectively. The cluster centroids are also stored. We do not use existing lexicons such as WordNet because they may have limited coverage of label words. Additionally, the automatic construction process allows us to extend our algorithm to arbitrary languages and domains.

With the availability of $\mathcal{R}$, we can generate knowledge-induced options. Given a pre-training sentence with the masked word as v, we query v against $\mathcal{R}$ for the most dissimilar cluster w.r.t. v, denoted as $\tilde{c}_v$, where the cosine similarity of the vector representation $\vec{v}$ and the cluster centroid is employed as the similarity measure. Finally, we randomly select one adjective from $\tilde{c}_v$ as the alternative label word to generate the *knowledge-induced options*. The text expressions of options is fixed, i.e., "Is it `[x1]` or `[x2]`?". Readers can further refer to the example in Figure 2.

V-Generation. For verbalizers, we map the true and the generated label words in the options to two classes, namely *Class: Correct* and *Class: Incorrect*. For instance, the verbalizers of the sample sentence in Figure 2 are:

$$\text{It is ``effective''.} \rightarrow \text{``Class: Correct''}$$
$$\text{It is ``ineffective''.} \rightarrow \text{``Class: Incorrect''}$$

Loss Function. The *KSMLM* loss is significantly different from the auxiliary MLM loss used in Schick and Schütze (2021a,b). In $\tilde{\mathcal{D}}$, each sample i consists of a pre-training sentence with exactly one masked token, the *knowledge-induced options* O_i and the prompt P_i. The PLM is trained to predict the correct masked word in the sentence, with the loss function $\mathcal{L}_{KSMLM} = -\sum_{i\in\tilde{\mathcal{D}}} P(\mathcal{V}|i, P_i, O_i, \Theta) \log Q(\mathcal{V}|i, P_i, O_i, \Theta)$. Overall, the loss function of *UPT* $\mathcal{L}$ is defined as the summation of the WMP and KSMLM loss:

$$\mathcal{L} = \mathcal{L}_{WMP} + \lambda \cdot \mathcal{L}_{KSMLM}$$

where $\lambda > 0$ is the balancing hyper-parameter.

Discussion. To our knowledge, external knowledge has also been applied to other prompt-based methods, such as KPT (Hu et al., 2021). The major difference between KPT and ours is that *UPT*

[5] We use the *spacy* toolkit in our work. URL: https://spacy.io/.

Group	Category	Task	#Training	#Testing	N	Class Labels
G1: Sentiment Analysis	Single Sentence	SST-2	6,920	872	2	positive, negative
		MR	8,662	2,000	2	positive, negative
		CR	1,775	2,000	2	positive, negative
G2: NLI	Sentence Pair	MNLI	392,702	9,815	3	entailment, neutral, contradiction
		SNLI	549,367	9,842	3	entailment, neutral, contradiction
		QNLI	104,743	5,463	2	entailment, not entailment
		RTE	2,490	277	2	entailment, not entailment
G3: Paraphrase	Sentence Pair	MRPC	3,668	408	2	equivalent, not equivalent
		QQP	363,846	40,431	2	equivalent, not equivalent

Table 1: Dataset statistics. We only sample $N \times K$ instances from the original training sets to form the few-shot training and development sets. The testing sets used in the experiments are full datasets.

uses the knowledge for options creation of the self-supervised task *KSMLM* that we proposed, in order to improve the model generalization abilities for accurate adaptation on new tasks. In contrast, previous works consider the expansion of verbalizers for specific downstream NLP tasks.

2.4 Few-shot Fine-tuning

For a specific downstream task $\mathcal{T}^*$, the samples in the target few-shot training set $\mathcal{D}^*$ can be processed and computed in the same way as those supervised tasks used during *UPT*. The learning consistency in the two stages ensures that the underlying PLM has already acquired prompting knowledge for $\mathcal{T}^*$. In addition, one can prompt-tune a single PLM over various tasks and uses it to fine-tune over any target tasks, making it computationally efficient to produce models for these applications.

3 Experiments

We conduct extensive experiments on a variety of NLP tasks to evaluate the *UPT* framework.

3.1 Experimental Settings

In the experiments, we employ nine public datasets to evaluate the proposed *UPT* framework, which are divided into three groups: sentiment analysis (SST-2 (Socher et al., 2013), MR (Hu and Liu, 2004), CR (Pang and Lee, 2005)), Natural Language Inference (NLI) (MNLI (Williams et al., 2018), SNLI (Bowman et al., 2015), QNLI (Wang et al., 2019a), RTE (Dagan et al., 2005)) and paraphrase (MRPC (Dolan and Brockett, 2005), QQP[6]). Table 1 lists the statistics of each dataset. In default, $K = 16$ (the number of training samples per class).

As mentioned above, during *UPT*, we leverage full training data from all dissimilar task groups, and then prompt-tune the model on the target task in the few-shot learning setting. For example, when the target task is SST-2, the training data during *UPT* is from NLI and paraphrase. The underlying PLM is the RoBERTa-large model (with 335M parameters) (Liu et al., 2019), unless otherwise specified. The baselines include standard *fine-tuning*, and three recently proposed few-shot learning algorithms: PET (Schick and Schütze, 2021a) [7], LM-BFF (Gao et al., 2021) [8] and P-tuning (Liu et al., 2021c) [9]. A variant of our approach (denoted as *UPT*-Single) is also implemented, which is our few-shot *fine-tuning* method based on the *POV* paradigm without the usage of dissimilar supervised or self-supervised datasets.

As we use other dissimilar datasets to train our model, we include two multi-task methods that are *meta-tuned* using the same dissimilar datasets as strong baselines, namely MT (Zero-shot) and MT (Few-shot) (Zhong et al., 2021a).[10] In addition, given a supervised NLP task, multiple prompts can be manually crafted. By augmenting one training sample with these prompts, we can automatically realize *self-ensemble learning*. For the self-ensemble version of *UPT*, we employ five different prompts. For each input sample, we randomly select one expression of options and one set of verbalizers. We denote this method as *UPT*-SE. The designed prompts, options and verbalizers are listed in the Appendix A. All the results of these models are evaluated in terms of averaged accuracy,

[6] https://www.quora.com/q/quoradata/.

[7] https://github.com/timoschick/pet

[8] https://github.com/princeton-nlp/LM-BFF

[9] https://github.com/THUDM/P-tuning

[10] In Zhong et al. (2021a), the authors only conduct zero-shot learning using larger PLMs. To make their work comparable to ours, we re-implement their algorithm over the Roberta model on our datasets under two settings. MT (Zero-shot) refers to the model tuned only using dissimilar full datasets. MT (Few-shot) further tunes the entire model over the target few-shot training set based on the prompts. Note that a few contemporaneous works (such as Wei et al. (2021)) also consider multi-task zero-shot learning. Because the settings and model scales are significantly different from us, they are not directly comparable.

Paradigm	Method	G1: Sentiment Analysis SST-2	G1: MR	G1: CR	G2: NLI MNLI	G2: SNLI	G2: QNLI	G2: RTE	G3: Paraphrase MRPC	G3: QQP	Avg.
Single-task methods w/o. the usage of dissimilar datasets ($K = 16$)											
FT	Fine-tuning	81.42	76.90	75.80	45.80	48.40	60.21	54.50	76.72	60.70	64.49
PT	PET	91.86	86.45	90.50	58.46	59.43	61.30	65.70	74.51	67.65	72.87
	LM-BFF	91.62	87.25	91.80	64.25	71.21	69.19	69.51	74.23	60.59	75.52
	P-Tuning	91.85	86.60	91.75	62.41	70.28	68.79	70.81	66.42	60.57	74.39
	***UPT*-Single**	92.89	87.65	91.15	64.47	70.20	68.33	68.23	71.57	69.36	75.98
Multi-task methods w. the usage of dissimilar datasets ($K = 16$)											
PT	MT (Zero-shot)	58.72	59.00	58.90	36.33	39.20	40.93	54.87	70.59	42.86	51.27
	MT (Few-shot)	92.09	86.55	91.00	69.60	67.12	68.94	68.59	71.08	77.83	76.98
	UPT	**93.46**	88.15	92.05	70.17	68.26	**71.87**	72.56	**76.96**	78.79	79.14
	***UPT*-SE**	93.12	**88.45**	**92.10**	**71.39**	**73.58**	70.51	**75.81**	76.23	**79.57**	**80.08**

Table 2: Comparison between *UPT* and baselines over all testing sets in terms of accuracy (%). "FT" and "PT" refer to the *fine-tuning* and *prompt-based fine-tuning* paradigm, respectively. The methods in bold refer to our approach and its variants. The scores of baselines are re-produced using their open-source codes.

over 5 random seeds.

Our *UPT* framework is implemented in PyTorch and run with NVIDIA V100 GPUs. Specifically, we train our model by the Adam optimizer. The learning rate for all training stages is fixed to be 1e-5. We set the default hyper-parameters as $\gamma = 0.001$ and $\lambda = 0.1$, which are also tuned over the development sets. The parameter regularizers are the same as in Gao et al. (2021).

3.2 Main Results

In Table 2, we report the general experimental results of *UPT* and all the baselines. The results show that: 1) Prompt-based methods (such as PET (Schick and Schütze, 2021a), LM-BFF (Gao et al., 2021) and P-tuning (Liu et al., 2021c)) have large improvements over standard *fine-tuning*. Additionally, *UPT*-Single slightly outperforms previous few-shot learning models in average. 2) *UPT* (both the original and the ensemble versions) consistently outperforms all baselines on all tasks and improves by over 4.56%, which demonstrates that our framework possesses better generalization by learning from dissimilar groups of tasks.[11] 3) MT (Zero-shot) (Zhong et al., 2021a) does not yields satisfactory results on BERT-style models. Different from ultra-large models, we suggest that few-shot prompt tuning is necessary for BERT-style models to produce good results over these tasks. By comparing *UPT* against MT (Few-shot), we can see that the proposed *POV* paradigm and the self-supervised *KSMLM* task are more effective for few-shot learning. 4) Generally, *UPT*-SE improves the averaged accuracy on all tasks by 0.94% than *UPT*. It means that self-ensemble learning can enhance model generalization, but the improvement is not consistent across all tasks. A possible cause is that some prompts and options are not optimal for the target task.

[11] We also conduct the single-tail paired t-test to compare our approach against few-shot baselines across tasks. The result is $p < 0.05$, indicating the statistical significance.

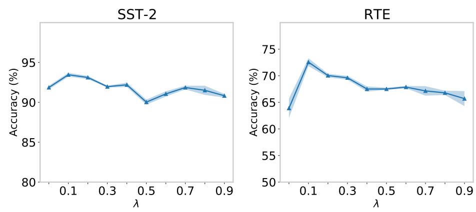

Figure 3: Parameter analysis w.r.t. hyper-parameter λ.

3.3 Model Analysis

Parameter Analysis. We conduct parameter analysis to investigate the best choice of the balance coefficient λ. Results over SST-2 and RTE are shown in Figure 3. We have the best performance when $\lambda = 0.1$, which indicates that our proposed *UPT* possess generalization when it is jointly trained with over the self-supervised *KSMLM* task. We also observe that the performance decreases when λ becomes larger. This means *KSMLM* is a suitable regularization task, but also may introduce a lot of prompts and options that are irrelevant to downstream tasks. This opens up new opportunities for model improvement.

Ablation Study. To clearly verify the contributions of each component in *UPT*, we conduct an ablation study. As shown in Table 4, w/o. *POV* denotes the method with manually designed prompts without the usage of any options. w/o. *KSMLM* equals the setting with $\lambda = 0$. w/o. *OKR* means that we randomly choose the alternative label words in the options without knowledge guidance, when we optimize the *KSMLM* task. w/o. *POV* & *KSMLM* denotes the method without any options and the auxiliary *KSMLM* task. The results show that no

Backbone	#Layer	#Dimension	#Param.	SST-2	MR	CR	Avg.
BERT-base	12	768	110M	82.57 (+3.79)	71.10 (+9.25)	78.05 (+8.85)	77.24 (+7.30)
BERT-medium	8	512	70M	68.00 (+2.98)	63.35 (+4.15)	70.15 (+6.10)	67.17 (+4.41)
BERT-small	4	512	31M	66.28 (+3.67)	58.10 (+4.55)	68.15 (+5.50)	64.18 (+4.57)
BERT-mini	4	256	22M	58.83 (+3.09)	59.40 (+7.60)	65.75 (+7.45)	61.33 (+6.05)
BERT-tiny	2	128	14M	54.13 (+3.79)	53.95 (+1.30)	54.40 (+5.20)	54.16 (+3.43)

Table 3: Results of model scale analysis. We report the accuracy (%) of *UPT* based on BERT with other scales, and relative improvements, compared to the models w/o. prompt learning over dissimilar datasets.

Method/Task	SST-2	MR	RTE	QQP
UPT	**93.46**	**88.15**	**72.56**	**78.79**
w/o. *POV*	91.51	86.55	66.43	78.64
w/o. *KSMLM*	91.86	87.15	63.90	78.34
w/o. *POV&KSMLM*	91.17	86.25	60.65	77.99
w/o. *OKR*	93.00	87.75	65.35	78.43

Table 4: Ablation study in terms of accuracy (%).

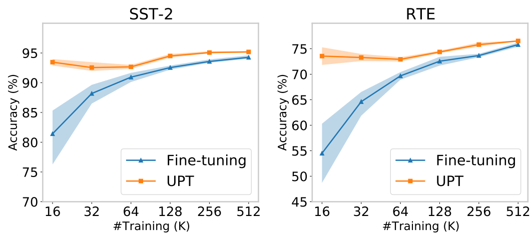

Figure 4: Results of sample efficiency analysis. We compare *UPT* with standard *fine-tuning* with different numbers of training samples K over two tasks.

matter which module is removed, the model performance is affected. Particularly, when we remove both *POV* and *KSMLM*, the performance is decreased by 2.29%, 1.6%, 11.91% and 0.8%, respectively. The accuracy values of this setting are lower than w/o. *POV* and w/o. *KSMLM*. It suggests that both of two components highly contribute to the high performance of our framework. In addition, the performance is further improved over MR, RTE and QQP by using the self-ensemble learning technique, which verifies the success of the combination of each components. Additionally, we find that if we use *KSMLM* but remove *OKR*, the results decreases over all these tasks, but are still higher than w/o. *KSMLM*. It means that the options knowledge that we mine from the corpus is suitable for the self-supervised learning task.

Sample Efficiency. We further explore the model effects with different numbers of training samples per class (K) from 32 to 512. We also use standard *fine-tuning* as the reference. As shown in Figure 4, each point refers the averaged score across 5 randomly sampled datasets. We observe that our *UPT* consistently achieves higher scores regardless of the number of training samples. In addition, the variance of *UPT* is lower than *fine-tuning*, meaning that the stability of our method is better. This is different from other prompt-based methods (Schick and Schütze, 2021a,b; Gao et al., 2021).

Model Scale Analysis. To further show that *UPT* can improve the model performance regardless of the scales, we regard multiple small-scale BERT as model backbones[12]. Due to space limitation, we only illustrate the results in Table 3 over SST-2, MR, and CR. To make a fair comparison, we also test the performance without the usage of dissimilar NLP datasets, and show the relative improvements. The results demonstrate that the model scale plays an important role in the ability of model generalization. We also find that *UPT* that uses dissimilar datasets can highly improve the effectiveness, especially on small-scale PLMs. Therefore, our method is better suitable for producing high-performing small PLMs for online applications.

Adaptation Efficiency of Task Groups. As for aforementioned, our framework focuses on multi-task *prompt-based fine-tuning* before *fine-tuning* on the few-shot target task. Therefore, it is worth exploring which group of tasks has a better effect on the adaptation improvement over the given target task. In this part, we re-design the multi-task training process. Specifically, when given a target task (e.g., MNLI), we only choose one group of tasks (e.g., MRPC and QQP of Group 3 (Paraphrase)) for multi-task prompt-tuning, and then fine-tune the model on the target task. As shown in Figure 5, the cell in the i-th row and j-th column denotes the relative improvement from single-task learning over the j-th task to the setting where the i-th group is added for multi-task prompt learning. We normalize the values of each column to show the percentage of influence of each group.

The results show that the performance of a target task improves the most when we add data samples from other datasets within the same task group. However, in low-resourced scenarios, sim-

[12] https://github.com/google-research/bert

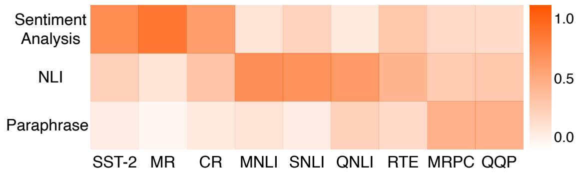

Figure 5: Adaptation efficiency between task groups. The shade of color indicates the degree of adaptation.

ilar datasets are not available. By using *UPT*, we can even transfer the knowledge from the datasets from dissimilar tasks to the target task.

4 Related Work

In this section, we summarize the related work on PLMs and prompt-based learning for PLMs.

4.1 Pre-trained Language Models

Recently, benefited from the powerful modeling abilities of PLMs and computational resources, we have witnessed the qualitative improvement of multiple NLP tasks (Qiu et al., 2020; Han et al., 2021a). For examples, the large GPT model series (Radford et al., 2019; Brown et al., 2020) utilizes multi-layer transformer decoders to capture left-to-right semantics of natural languages. BERT (Devlin et al., 2019) focuses on the learning of bidirectional contextual representations based on transformer encoders. Other notable PLMs include Transformer-XL (Dai et al., 2019), ELMo (Peters et al., 2018), RoBERTa (Liu et al., 2019), AlBERT (Lan et al., 2020), ERNIE (Zhang et al., 2019), XLNet (Yang et al., 2019), StructBERT (Wang et al., 2019b), Big Bird (Zaheer et al., 2020), SpanBERT (Joshi et al., 2020), T5 (Raffel et al., 2020), etc. As the model architecture of PLMs is not the focus of our work, we do not further elaborate.

4.2 Prompt-based Learning for PLMs

Fine-tuning PLMs directly by learning the `[CLS]` head may perform poorly with few training samples due to model overfitting (Liu et al., 2021a). Recently, the huge GPT-3 model (Brown et al., 2020) has been proposed to enable in-context learning, which introduces handcrafted prompts and demonstrations. Schick and Schütze (2021a) apply handcrafted prompts to prompt-based fine-tuning for BERT-style models. To facilitate automatic prompt generation, Gao et al. (2021) present LM-BFF to generate discrete templates (Raffel et al., 2020). Other works (Shin et al., 2020; Han et al., 2021b; Scao and Rush, 2021; Utama et al., 2021) mine prompts from the training corpus based on heuristic rules or semantic relations. However, these methods are time-consuming for mining optimized prompts for target tasks. To deal with this problem, a series of methods are proposed to learn continuous/soft prompt embeddings with differentiable parameters, such as P-tuning (Liu et al., 2021c), P-tuning-V2 (Liu et al., 2021b), OptiPrompt (Zhong et al., 2021b), Prefix-tuning (Li and Liang, 2021). Zhao and Schütze (2021); Gu et al. (2021) focus on the hybrid training with both discrete and continuous prompts. Hu et al. (2021) consider the automatic expansion of label words, and presents Knowledgeable Prompt-tuning (KPT) to utilize knowledge from knowledge bases for the construction of verbalizers. Sun et al. (2021) and Wang et al. (2021b) prompt the PLMs to make language inference in zero-shot learning. In addition, Wang et al. (2021a); Vu et al. (2021) consider transfer learning on continuous prompt-tuning, and achieve better performance on cross-task training. Li et al. (2021); Chen et al. (2021); Ma et al. (2021) focus on prompts for specific NLP tasks, such as sentiment analysis, information extraction and question answering.

Recently, Wei et al. (2021); Zhong et al. (2021a); Min et al. (2021); Mishra et al. (2021) tune PLMs on mixed data samples drawn from different NLP tasks with manually designed task-specific prompts. The resulting PLMs are then utilized to solve unseen tasks by zero-shot learning. These methods successfully work for large PLMs such as GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2020), but consume a large amount of computational resources. Our work further leverages data from non-target tasks for BERT-style PLMs, which includes a unified prompting paradigm that makes the prompt-tuned PLMs have better capacities of adapting to previously unseen NLP tasks.

5 Conclusion

In this paper, we present the *Unified Prompt Tuning* framework (*UPT*) that enables better few-shot learning for BERT-style models by explicitly capturing prompting semantics from non-target NLP datasets. Extensive experiments are conducted on a variety of tasks, showing that *UPT* consistently outperforms state-of-the-arts for prompt-based fine-tuning. As for future work, we seek to extend *UPT* to other tasks such as named entity recognition, text generation, and machine translation.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. *CoRR*, abs/2104.07650.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *MLCW*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP 2005*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*, pages 3816–3830.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. PPT: pre-trained prompt tuning for few-shot learning. *CoRR*, abs/2109.04332.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021a. Pre-trained models: Past, present and future. *CoRR*, abs/2106.07139.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021b. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD 2004*, pages 168–177.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *CoRR*, abs/2108.02035.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 64–77.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *CoRR*, abs/2104.08691.

Chengxi Li, Feiyu Gao, Jiajun Bu, Lu Xu, Xiang Chen, Yu Gu, Zirui Shao, Qi Zheng, Ningyu Zhang, Yongpan Wang, and Zhi Yu. 2021. Sentiprompt: Sentiment knowledge enhanced prompt-tuning for aspect-based sentiment analysis. *CoRR*, abs/2109.08306.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. GPT understands, too. *CoRR*, abs/2103.10385.

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

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Qi Zhang, and Xuanjing Huang. 2021. Template-free prompt tuning for few-shot NER. *CoRR*, abs/2109.13532.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Metaicl: Learning to learn in context. *CoRR*, abs/2110.15943.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2021. Reframing instructional prompts to gptk's language. *CoRR*, abs/2109.07830.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Teven Le Scao and Alexander M. Rush. 2021. How many data points is a prompt worth? In *NAACL*, pages 2627–2636.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, pages 255–269.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*, pages 2339–2352.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, pages 4222–4235.

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 *EMNLP*, pages 1631–1642.

Yi Sun, Yu Zheng, Chao Hao, and Hangping Qiu. 2021. NSP-BERT: A prompt-based zero-shot learner through an original pre-training task-next sentence prediction. *CoRR*, abs/2109.03564.

Prasetya Ajie Utama, Nafise Sadat Moosavi, Victor Sanh, and Iryna Gurevych. 2021. Avoiding inference heuristics in few-shot prompt-based finetuning. *CoRR*, abs/2109.04144.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *CoRR*, abs/2110.07904.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Chengyu Wang, Jianing Wang, Minghui Qiu, Jun Huang, and Ming Gao. 2021a. Transprompt: Towards an automatic transferable prompting framework for few-shot text classification. In *EMNLP*, pages 2792–2802.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021b. Entailment as few-shot learner. *CoRR*, abs/2104.14690.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019b. Structbert: incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, pages 1112–1122.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451.

Mengjie Zhao and Hinrich Schütze. 2021. Discrete and soft prompting for multilingual models. In *EMNLP*, pages 8547–8555.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021a. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *EMNLP*, pages 2856–2878.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021b. Factual probing is [MASK]: learning vs. learning to recall. In *NAACL*, pages 5017–5033.

Task	Prompt	Option	Label word
SST-2	**Template 1**: [<s1>]. It was [MASK]. **Template 2**: [<s1>]. I thought it was [MASK]. **Template 3**: [<s1>]. It is [MASK]. **Template 4**: [<s1>]. The review is [MASK]. **Template 5**: [<s1>]. A [MASK] one.	**Option 1**: Is <x1> or <x2>? **Option 2**: Does <x1> or <x2>? **Option 3**: <x1> or <x2>?	**Verbalizer 1**: Negative (Bad), Positive (Wonderful) **Verbalizer 2**: Negative (Silly), Positive (Solid) **Verbalizer 3**: Negative (Pathetic), Positive (Irresistible)
MR	**Template 1**: [<s1>]. It was [MASK]. **Template 2**: [<s1>]. A [MASK] piece of work. **Template 3**: [<s1>]. It is [MASK]. **Template 4**: [<s1>]. The film is [MASK]. **Template 5**: [<s1>]. A really [MASK] movie.	**Option 1**: Is <x1> or <x2>? **Option 2**: Does <x1> or <x2>? **Option 3**: <x1> or <x2>?	**Verbalizer 1**: Negative (Horrible), Positive (Exquisite) **Verbalizer 2**: Negative (Silly), Positive (Solid) **Verbalizer 3**: Negative (Bad), Positive (Wonderful)
CR	**Template 1**: [<s1>]. It was [MASK]. **Template 2**: [<s1>]. It looks [MASK]. **Template 3**: [<s1>]. It is [MASK]. **Template 4**: [<s1>]. The quality is [MASK]. **Template 5**: [<s1>]. I thought it was [MASK].	**Option 1**: Is <x1> or <x2>? **Option 2**: Does <x1> or <x2>? **Option 3**: <x1> or <x2>?	**Verbalizer 1**: Negative (Horrible), Positive (Fantastic) **Verbalizer 2**: Negative (Silly), Positive (Solid) **Verbalizer 3**: Negative (Bad), Positive (Wonderful) **Verbalizer 4**: Negative (Pointless), Positive (Neat)
MNLI	**Template 1**: [<s1>]. You are right, [MASK]. [<s2>]. **Template 2**: [<s1>]. It was [MASK]. [<s2>]. **Template 3**: [<s1>], [<s2>]. It is [MASK]. **Template 4**: [<s1>]. It is true that [MASK], [<s2>]. **Template 5**: [<s1>]. [MASK]. Then, [<s2>].	**Option 1**: Is <x1> or <x2> or<x3> ? **Option 2**: Based on the paragraph above, is the following <x1> or <x2> or <x3>?	**Verbalizer 1**: Contradiction (Next), Entailment (Exactly), Neutral (Indeed) **Verbalizer 2**: Contradiction (Wrong), Entailment (True), Neutral (Uncertain) **Verbalizer 3**: Contradiction (Otherwise), Entailment (Fine), Neutral (Plus) **Verbalizer 4**: Contradiction (Otherwise), Entailment (Exactly), Neutral (Naturally)
SNLI	**Template 1**: [<s1>]. [MASK], no, [<s2>]. **Template 2**: [<s1>]. [MASK], in this case, [<s2>]. **Template 3**: [<s1>]. [MASK], I think, [<s2>]. **Template 4**: [<s1>], [<s2>]. It was [MASK]. **Template 5**: [<s1>]. [MASK], [<s2>].	**Option 1**: Is <x1> or <x2> or<x3> ? **Option 2**: Based on the paragraph above, is the following <x1> or <x2> or <x3>?	**Verbalizer 1**: Contradiction (Next), Entailment (Exactly), Neutral (Indeed) **Verbalizer 2**: Contradiction (Wrong), Entailment (True), Neutral (Uncertain) **Verbalizer 3**: Contradiction (Instead), Entailment (Indeed), Neutral (Basically) **Verbalizer 4**: Contradiction (Except), Entailment (Alright), Neutral (Watch)
QNLI	**Template 1**: Question: [<s1>]? [<s2>]. The answer: [MASK]. **Template 2**: Question: [<s1>]? [<s2>]. [MASK]. **Template 3**: Question: [<s1>]? [MASK], Yes, [<s2>]. **Template 4**: [<s1>]? [MASK], it is known that [<s2>]. **Template 5**: [<s1>]? [MASK]. Then, [<s2>].	**Option 1**: Is <x1> or <x2> ? **Option 2**: Based on the question, is the following <x1> or <x2>? **Option 3**: Is the answer <x1> or <x2>?	**Verbalizer 1**: Entailment (Yes), Not Entailment (No) **Verbalizer 2**: Entailment (Okay), Not Entailment (Nonetheless) **Verbalizer 3**: Entailment (Notably), Not Entailment (Yet)
RTE	**Template 1**: [<s1>]. [<s2>]. The answer: [MASK]. **Template 2**: [<s1>]. [<s2>]. [MASK]. **Template 3**: [<s1>]. [MASK], I think, [<s2>]. **Template 4**: [<s1>]. The question: [<s2>]? It is [MASK]. **Template 5**: [<s1>]. [MASK]. I believe, [<s2>].	**Option 1**: Is <x1> or <x2> ? **Option 2**: Based on the question, the answer is <x1> or <x2>? **Option 3**: Is the answer <x1> or <x2>?	**Verbalizer 1**: Entailment (So), Not Entailment (Meanwhile) **Verbalizer 2**: Entailment (Yes), Not Entailment (No) **Verbalizer 3**: Entailment (Notably), Not Entailment (Yet)
MRPC	**Template 1**: [<s1>]. [<s2>]. The answer: [MASK]. **Template 2**: [<s1>]. [<s2>]. [MASK]. **Template 3**: [<s1>]. [MASK], however, [<s2>]. **Template 4**: [<s1>]. [<s2>]. In fact [MASK]. **Template 5**: [<s1>]. [MASK]. that's right, [<s2>].	**Option 1**: Is <x1> or <x2> ? **Option 2**: Are two question <x1> or <x2>? **Option 3**: <x1> or <x2>?	**Verbalizer 1**: 0 (Alas), 1 (Rather) **Verbalizer 2**: 0 (Different), 1 (Same) **Verbalizer 3**: 0 (Wrong), 1 (Right)
QQP	**Template 1**: [<s1>]. [<s2>]. The answer: [MASK]. **Template 2**: [<s1>]. [<s2>]. [MASK]. **Template 3**: [<s1>]. [MASK], however, [<s2>]. **Template 4**: [<s1>]. [<s2>]. In fact [MASK]. **Template 5**: [<s1>]. [MASK]. that's right, [<s2>].	**Option 1**: Is <x1> or <x2> ? **Option 2**: Are two question <x1> or <x2>? **Option 3**: <x1> or <x2>?	**Verbalizer 1**: 0 (Alas), 1 (Rather) **Verbalizer 2**: 0 (Different), 1 (Same) **Verbalizer 3**: 0 (Wrong), 1 (Right)

Table 5: The Prompts, Options and Verbalizers (POV) for each task. <s1> and <s2> denote the input sentences. <x1>, <x2> and <x3> denote the label words.

A The *POV* Settings of All Tasks

As shown in Table 5, we list all the designed *POV*s for each task. Note that for each task group, the options are the same, but verbalizers may be different. For example, SST-2, MR, and CR have the same schema of options, but with different verbalizers.