Making Pre-trained Language Models End-to-end Few-shot Learners with Contrastive Prompt Tuning

Ziyun Xu[1,2*], Chengyu Wang[1*], Minghui Qiu[1#], Fuli Luo[1], Runxin Xu[1,3], Songfang Huang[1], Jun Huang[1]

[1] Alibaba Group [2] School of Computer Science, Carnegie Mellon University [3] Key Laboratory of Computational Linguistics, Peking University

ziyunx@andrew.cmu.edu,{chengyu.wcy,minghui.qmh,lfl259702,songfang.hsf,huangjun.hj}@alibaba-inc.com,runxinxu@gmail.com

ABSTRACT

Pre-trained Language Models (PLMs) have achieved remarkable performance for various language understanding tasks in IR systems, which require the fine-tuning process based on labeled training data. For low-resource scenarios, prompt-based learning for PLMs exploits prompts as task guidance and turns downstream tasks into masked language problems for effective few-shot fine-tuning. In most existing approaches, the high performance of prompt-based learning heavily relies on handcrafted prompts and verbalizers, which may limit the application of such approaches in real-world scenarios. To solve this issue, we present *CP-Tuning*, the first end-to-end *Contrastive Prompt Tuning* framework for fine-tuning PLMs *without any manual engineering of task-specific prompts and verbalizers*. It is integrated with the task-invariant continuous prompt encoding technique with fully trainable prompt parameters. We further propose the pair-wise cost-sensitive contrastive learning procedure to optimize the model in order to achieve verbalizer-free class mapping and enhance the task-invariance of prompts. It explicitly learns to distinguish different classes and makes the decision boundary smoother by assigning different costs to easy and hard cases. Experiments over a variety of language understanding tasks used in IR systems and different PLMs show that *CP-Tuning* outperforms state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**.

KEYWORDS

few-shot learning, prompt-based fine-tuning, Pre-trained Language Models, deep contrastive learning

ACM Reference Format:

Ziyun Xu[1,2*], Chengyu Wang[1*], Minghui Qiu[1#], Fuli Luo[1], Runxin Xu[1,3], Songfang Huang[1], Jun Huang[1]. 2018. Making Pre-trained Language Models End-to-end Few-shot Learners with Contrastive Prompt Tuning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

[*] Z. Xu and C. Wang contributed equally. Corresponding author: M. Qiu.

1 INTRODUCTION

Starting from BERT [4], fine-tuning large-scale Pre-trained Language Models (PLMs) has become the *de facto* standard practice for solving a majority of Natural Language Processing (NLP) tasks [18, 42, 49], which have been extensively used in Information Retrieval (IR) systems for tasks such as content analysis, question matching and question answering [24]. To guarantee high accuracy, it is necessary to obtain a sufficient amount of training data for downstream tasks, which is the bottleneck in low-resource scenarios.

The successful application of the ultra-large GPT-3 model [2] shows that with a sufficiently large memory capacity and massive pre-training computation, large PLMs can learn to solve a task with very few training samples. However, the large model size and the long inference time make it infeasible to deploy such PLMs online with limited computational resources. Inspired by these works, Gao et al. [6] propose a prompt-based approach to fine-tune BERT-style PLMs in a few-shot learning setting. It converts text classification and regression problems into masked language problems where the knowledge captured during pre-training can be better utilized during the few-shot learning process. Similar usage of prompts for fine-tuning PLMs has also been shown in [33, 34] and many others. Scao and Rush [32] conduct a rigorous test to show that prompting is highly beneficial in low-data regimes.

In most prompt-based approaches, there exist two types of model components that require careful manual engineering, namely *prompts* and *verbalizers*. Here, prompts are fixed templates or patterns that are employed to inject task-specific guidance to input texts, while verbalizers establish explicit mappings between output tokens and class labels. An example of prompts and verbalizers on review sentiment analysis is illustrated in the left part of Figure 1. As reported in [22], designing high-performing prompts and the corresponding verbalizers is challenging and requires a very large validation set. As for prompts, even a slight change of expressions can lead to big variance in the performance of downstream tasks. To alleviate this issue, Liu et al. [21, 22] propose P-tuning, which uses *continuous prompt embeddings* to avoid the manual prompt engineering process. However, this method still requires the design of verbalizers, with a strong hypothesis of token-to-label mappings. Therefore, the drawbacks of prompt and verbalizer engineering potentially hinder the wide application of these approaches.

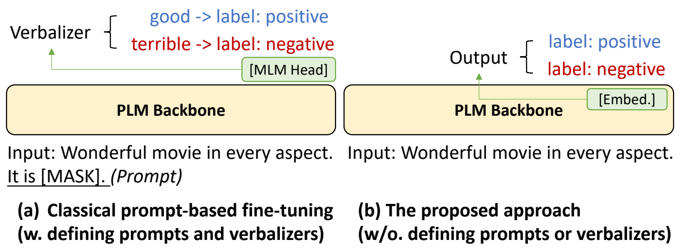

Verbalizer
good -> label: positive
terrible -> label: negative
[MLM Head]
PLM Backbone
Input: Wonderful movie in every aspect. It is [MASK]. (Prompt)
(a) Classical prompt-based fine-tuning (w. defining prompts and verbalizers)
Output
label: positive
label: negative
[Embed.]
PLM Backbone
Input: Wonderful movie in every aspect.
(b) The proposed approach (w/o. defining prompts or verbalizers)

Figure 1: A simple comparison between classical prompt-based fine-tuning and *CP-Tuning* w.r.t. the inputs and outputs. The underlying task is review sentiment analysis.

To address these issues, we present *CP-Tuning*, an end-to-end *Contrastive Prompt Tuning* framework for PLMs *without the manual design of task-specific prompts and verbalizers*. To our knowledge, our work is the first to study contrastive learning for prompt-based fine-tuning without manual prompt and verbalizer engineering. Specifically, our approach consists of two major techniques:

- **Task-invariant Continuous Prompt Encoding.** We employ *continuous embeddings* as prompts and do not employ any prompt encoders to avoid learning additional parameters during few-shot learning (in contrast to [22]). Specially, we initialize continuous prompt embeddings as the pre-trained representations of a collection of *task-invariant* tokens, and enable prompt embeddings to be *task-adaptive* by back propagation. Hence, *CP-Tuning* does not require manual prompt engineering for specific tasks.
- **Verbalizer-free Class Mapping.** We further propose the novel *verbalizer-free* mechanism to ease the manual labor of designing verbalizers and to improve the generalization ability of our model, as well as the task-invariance of prompts. Specifically, the *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) is introduced to train our few-shot learner, together with an auxiliary Mask Language Modeling (MLM) task as the regularizer. *PCCL* explicitly learns to distinguish different classes and makes the decision boundary smoother by assigning different costs to easy and hard cases. In contrast to previous approaches, embeddings of instances before the MLM classifier are directly used for inference. We also theoretically prove that *PCCL* is an extension to various metric learning based loss functions.

For evaluation, we conduct extensive experiments to verify the effectiveness of *CP-Tuning* over eight public datasets, including various tasks used in IR and NLP systems (e.g., sentiment analysis, question matching, Natural Language Inference (NLI)). Experimental results show that *CP-Tuning* consistently outperforms state-of-the-arts for prompt-based few-shot learning.

In summary, we make the following contributions:

- We introduce the end-to-end *CP-Tuning* framework to enable prompt-based few-shot learning without designing task-specific prompts and verbalizers. To our knowledge, our work is the first to employ contrastive learning for end-to-end prompt-based learning that eases manual engineering.

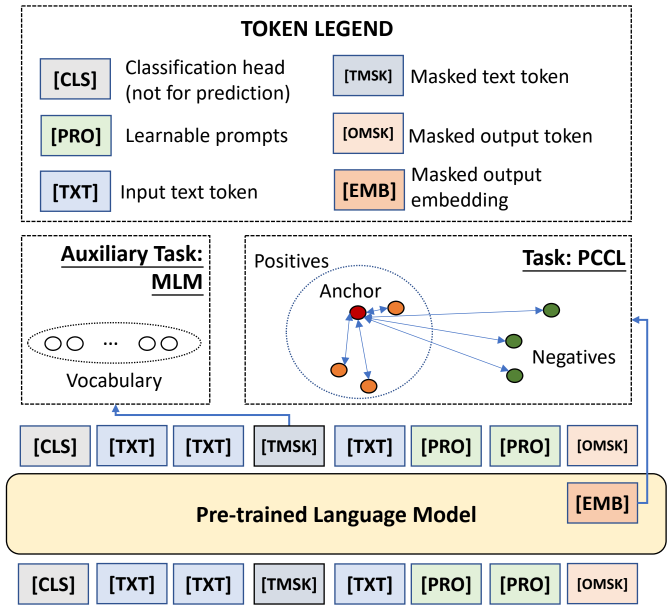

TOKEN LEGEND
[CLS] Classification head (not for prediction)
[TMSK] Masked text token
[PRO] Learnable prompts
[OMSK] Masked output token
[TXT] Input text token
[EMB] Masked output embedding
Auxiliary Task: MLM
Vocabulary
Positives
Anchor
Task: PCCL
Negatives
[CLS] [TXT] [TXT] [TMSK] [TXT] [PRO] [PRO] [OMSK]
Pre-trained Language Model
[EMB]
[CLS] [TXT] [TXT] [TMSK] [TXT] [PRO] [PRO] [OMSK]

Figure 2: An overview of the *CP-Tuning* framework. For simplicity, we only show input/output text sequences for single-sentence classification tasks.

- In *CP-Tuning*, the task-invariant continuous prompt encoding technique is presented. We further propose the *PCCL* technique to train the model without the usage of any verbalizers based on contrastive learning.
- Experiments over eight public datasets show that *CP-Tuning* consistently outperforms state-of-the-arts for prompt-based few-shot learning. We also theoretically derive the relations between *PCCL* and other losses.

The rest of this paper is organized as follows. Section 2 presents our *CP-Tuning* in detail. The experiments are shown in Section 3, with the related work summarized in Section 4. Finally, we draw the conclusion and discuss the future work in Section 5.

2 *CP-TUNING*: PROPOSED APPROACH

In this section, we begin with an overview of our approach. After that, the detailed techniques are elaborated.

2.1 Overview of *CP-Tuning*

We begin with a few basic notations. Let $\mathcal{D}$ be an N-way K-shot training set of a specific NLP task, where each of the N classes is associated with K training samples. [1] Denote $\mathcal{M}$ as the collection of parameters of the underlying PLM. The goal of our work is to generate a high-performance few-shot learner initialized from $\mathcal{M}$ based on $\mathcal{D}$ that can effectively generalize to previously unseen

[1]Our work can be easily extended to other fine-tuning scenarios without modification where each class is associated with different numbers of training samples. We also find that *CP-Tuning* is better at learning with unbalanced training sets than previous prompt-based methods. Readers can refer to experiments for details.

Category	Task	Example of Input Sequence
Single Sentence	Sentiment Analysis	Movie fans, get ready to take off... the other direction. [CLS] Movie [TMSK], get ready to take off... the other direction. $[\text{PRO}]_{\text{it}}$ $[\text{PRO}]_{\text{is}}$ [OMSK]
Single Sentence	Subjectivity Classification	Zero, who, like many Hong Kong youngsters, has a handful of unsteady jobs. [CLS] Zero, who, like [TMSK] Hong Kong youngsters, has a handful of unsteady jobs. $[\text{PRO}]_{\text{it}}$ $[\text{PRO}]_{\text{is}}$ [OMSK]
Sentence Pair	NLI	a. What was Telenet? b. Telenet was incorporated in 1973 and started operations in 1975. [CLS] What was Telenet? $[\text{PRO}]_{?}$[OMSK][SEP] Telenet was [TMSK] in 1973 and started operations in 1975.
Sentence Pair	Question Matching	a. How do I start trying to trace my family tree? b. How would I start tracing my family history? [CLS] How do I start trying to trace my family tree? $[\text{PRO}]_{?}$[OMSK][SEP] How would I [TMSK] tracing my family history?
Sentence Pair	Sentence Equivalence	a. Around 1,500 police are to be deployed at Niigata for the ferry's visit. b. About 1,500 police will be deployed for the visit. [CLS] Around 1,500 police are to be deployed at Niigata for the ferry's [TMSK]. $[\text{PRO}]_{?}$[OMSK][SEP] About 1,500 police will be deployed for the visit.

Table 1: Examples of inputs and processed token sequences for *CP-Tuning* (first and second lines of input sequence). $[\text{PRO}]_{\text{test}}$ refers to the prompt embedding token initialized by the representation of the token "text", which can be updated via back propagation. Note that the initializations of prompts w.r.t. all single-sentence (or sentence-pair) tasks are the same.

data samples of the same task. We present the overview of our approach in Figure 2, with major techniques summarized below.

As traditional prompt-based models require the cumbersome process of prompt engineering, we employ *continuous embeddings* as input prompts. Rather than employing sub-networks (*e.g.,* LSTMs) as prompt encoders [22], to avoid learning additional parameters during few-shot learning, we directly feed prompt embeddings to the PLM encoder, and enable the embeddings to be *task-adaptive* by back propagation.

Besides manually-designed patterns, previous methods also require handcrafted verbalizers, which map the output of the masked token to the class label [33, 34]. In our work, we propose the *verbalizer-free* mechanism to ease the manual labor and to improve the generalization ability of our few-shot learner. As prompts and verbalizers are semantically correlated, this technique also enhances the task-invariance of prompts. Inspired by the contrastive learning paradigm [11], we propose the *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) to train our few-shot learner. In the few-shot learning setting, the lack of training data may easily result in model over-fitting. Hence, an auxiliary MLM loss is also optimized during few-shot learning to alleviate the issue. In addition, we further show that *PCCL* is an extension to a variety of loss functions.

2.2 Task-invariant Prompt Encoding

The input format of our approach is significantly different from previous works to facilitate *task-invariant continuous prompt learning*. To be more specific, in contrast to [4], we have three additional types of special tokens used as the inputs to the PLM:

- [PRO] (Prompt): the placeholder for continuous prompt embeddings;
- [TMSK] (Token Mask): the token mask of the input texts for optimizing the auxiliary MLM loss;
- [OMSK] (Output Mask): the mask as a placeholder to generate the output result.

For a better understanding, please refer to an example for single-sentence classification in Figure 2. Here, "[TMSK]" is only applied to a small portion of the input texts for MLM. "[OMSK]" is used for generating outputs, rather than the "[CLS]" token. Hence, no additional parameters are introduced to our model for prompt learning.

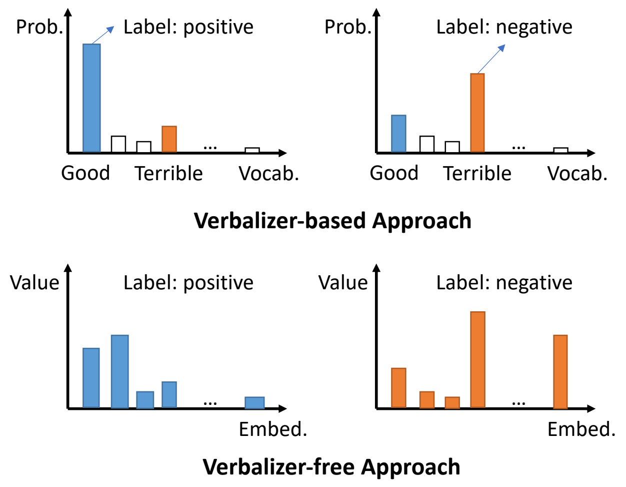


Figure 3: A simple comparison between *verbalizer-based* and *verbalizer-free* approaches w.r.t.the model outputs (which are the output token probability of the masked language token and the dense "[OMSK]" embeddings, respectively). The underlying task is review sentiment analysis.

As the parameters w.r.t. "[PRO]" tokens need to be learned for a given task, the lack of training data in few-shot learning still brings some burdens. We initialize prompt embeddings to be the pre-trained representations of *universal task-invariant prompts*. Here, the universal task-invariant prompt for single-sentence tasks is "it is"; and "?" for sentence-pair tasks. Note that the "[PRO]" and "[OMSK]" tokens are placed between the two pieces of texts to better capture the relations between the sentence pair. Refer to examples in Table 1. This setting can be viewed as the *knowledge prior* for prompt embeddings. During model training, the representations of prompts can be automatically adapted to specific tasks. In the experiments, we further show that it is unnecessary to design task-specific prompts for our approach. Despite the usage of such prompts, our method should be regarded as *task-invariant prompt-free* because no manual work is required for designing different prompts for specific tasks.

2.3 Verbalizer-free Class Mapping

A common property of existing prompt-based approaches is that they require handcrafted verbalizers to establish mappings between tokens and class labels [22, 33, 34]. We suggest that this practice might be sub-optimal. Consider the example on review analysis in Figure 3. *Verbalizer-based* approaches generate the distributions over the entire vocabulary (which may contain over 10 thousand words), and only pay attention to the probabilities of very few words (such as "good" and "terrible" in our case). The semantic association between words is also ignored to a large extent. For example, the probabilities of words such as "nice", "fantastic", "bad" and "horrible" are also strong indicators of class labels. If we replace the high-dimensional, sparse distributions with lower-dimensional, dense representations, the generalization ability and the flexibility of the underlying model can be largely increased.

In our work, we propose a novel *verbalizer-free* approach to generate model outputs based on *PCCL*. During training, denote $\mathcal{B}$ as the collection of instances in a batch ($\mathcal{B} \subset \mathcal{D}$). Each instance $i \in \mathcal{B}$ can be treated as an *anchor*, with the label denoted as y_i. We also have the *positive set* $P(i)$ and the *negative set* $N(i)$ w.r.t. the instance i and the batch $\mathcal{B}$:

$$\begin{aligned} P(i) &= \{j \mid j \neq i, y_j = y_i, j \in \mathcal{B}\} \\ N(i) &= \{j \mid y_j \neq y_i, j \in \mathcal{B}\} \end{aligned} \tag{1}$$

Let $\vec{z}_i$ be the l_2-normalized embedding of the "[OMSK]" token of the last layer of the underlying PLM (before the *softmax* function). In the context of contrastive learning, we aim to maximize the within-class similarity $s_{i,p} = \vec{z}_i^T \cdot \vec{z}_p$ where $p \in P(i)$, and also minimize the between-class similarity $s_{i,n} = \vec{z}_i^T \cdot \vec{z}_n$ where $n \in N(i)$. Following previous supervised contrastive learning models [7, 14], it is straightforward to derive the *sample-wise* contrastive loss:

$$\mathcal{L}_{CL}(i) = -\log \frac{\exp(s_{i,p}/\tau)}{\exp(s_{i,p}/\tau) + \exp(s_{i,n}/\tau)} \tag{2}$$

where τ is the temperature value. When multiple instances in $P(i)$ and $N(i)$ are considered, we re-write $\mathcal{L}_{CL}(i)$ as follows:

$$\mathcal{L}_{CL}(i) = -\log \sum_{p \in P(i)} \frac{\exp(s_{i,p}/\tau)}{\sum_{a \in A(i)} \exp(s_{i,a}/\tau)} \tag{3}$$

where the collection $A(i) = \mathcal{B} \setminus \{i\}$. This gives the model more generalization abilities in that multiple within-class and between-class similarity values are averaged, thus making the learned decision boundary smoother. The partial gradient of $\mathcal{L}_{CL}(i)$ is given by:

$$\begin{aligned} \frac{\partial \mathcal{L}_{CL}(i)}{\partial \vec{z}_i} = & \frac{\sum_{p \in P(i)} \exp(s_{i,p}/\tau)\vec{z}_p + \sum_{n \in N(i)} \exp(s_{i,n}/\tau)\vec{z}_n}{\tau \cdot \sum_{a \in A(i)} \exp(s_{i,a}/\tau)} \\ & - \frac{\sum_{p \in P(i)} \exp(s_{i,p}/\tau)\vec{z}_p}{\tau \cdot \sum_{p \in P(i)} \exp(s_{i,p}/\tau)} \end{aligned} \tag{4}$$

Minimizing $\mathcal{L}_{CL}(i)$ alone may be insufficient as it does not consider sample difficulty. For example, if $s_{i,p} = 0.2$ and $s_{i,p'} = 0.95$ where $p, p' \in P(i)$. The model should pay more attention to $s_{i,p}$ to reach the optima, and less attention to $s_{i,p'}$ to avoid model overfitting. Inspired by [41], we introduce *pair-wise relaxation factors* and propose a new loss function named *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) as follows:

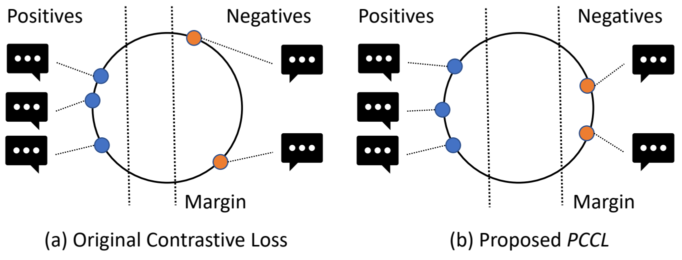


Figure 4: Illustration of how *PCCL* improves the learning process of "[OMSK]" embeddings of the last transformer encoder layer for review sentiment analysis. *PCCL* enlarges the margins between embeddings of instances from different classes, and the margins among embeddings of different negative samples w.r.t. the anchor.

$$\mathcal{L}_{PCCL}(i) = -\sum_{p \in P(i)} \log \frac{\exp(\alpha_{i,p} \cdot s_{i,p}/\tau_p)}{\mathcal{Z}(i)} \tag{5}$$

where $\mathcal{Z}(i)$ is the normalization factor:

$$\mathcal{Z}(i) = \sum_{p \in P(i)} \exp(\frac{\alpha_{i,p}}{\tau_p} s_{i,p}) + \sum_{n \in N(i)} \exp(\frac{\alpha_{i,n}}{\tau_n} s_{i,n}) \tag{6}$$

$\alpha_{i,p}$ and $\alpha_{i,n}$ are *pair-wise relaxation factors* with the definitions formulated as follows:

$$\begin{aligned} \alpha_{i,p} &= \max\{0, 1 + m - s_{i,p}\} \\ \alpha_{i,n} &= \max\{0, s_{i,n} + m\} \end{aligned} \tag{7}$$

Comparing to the original $\mathcal{L}_{CL}(i)$, two new features are added to *PCCL*. Inside $\alpha_{i,p}$ and $\alpha_{i,n}$, a margin factor m is employed to expect that $s_{i,p} > 1 - m$ and $s_{i,n} < m$. Hence, there is a relaxed margin between $s_{i,p}$ and $s_{i,n}$. The usage of $\alpha_{i,p}$ and $\alpha_{i,n}$ also makes the model focus on learning hard cases and avoid over-fitting on easy cases. Another empirical setting is that we use separate temperatures τ_p and τ_n for within-class and between-class similarities, instead of a uniform temperature τ. We further set $\tau_p = \xi \cdot \tau_n$ ($\xi > 1$) to give more relaxations on positive samples in order to make the within-class similarities not too large, as it is easy to see:

$$\frac{\alpha_{i,p}}{\tau_p} s_{i,p} = \frac{\alpha_{i,p}}{\xi \cdot \tau_n} s_{i,p} = \frac{\tilde{\alpha}_{i,p}}{\tau_n} s_{i,p} \tag{8}$$

where $\tilde{\alpha}_{i,p} = \max\{0, \frac{1}{\xi}(1 + m - s_{i,p})\}$. In this way, our few-shot learner will be less likely to over-fit to training instances. During the learning process, the $\vec{z}_i$ embeddings are optimized by computing the gradients $\frac{\partial \mathcal{L}_{PCCL}(i)}{\partial \vec{z}_i}$ and updating the PLM where:

$$\begin{aligned} \frac{\partial \mathcal{L}_{PCCL}(i)}{\partial \vec{z}_i} = & \frac{\sum_{p \in P(i)} \exp(\alpha_{i,p} s_{i,p}/\tau_p)\vec{z}_p + \sum_{n \in N(i)} \exp(\alpha_{i,n} s_{i,n}/\tau_n)\vec{z}_n}{\mathcal{Z}_p(i)} \\ & - \frac{\sum_{p \in P(i)} \exp(\alpha_{i,p} s_{i,p}/\tau_p)\vec{z}_p}{\tau_p \cdot \sum_{p \in P(i)} \exp(\alpha_{i,p} s_{i,p}/\tau_p)} \end{aligned} \tag{9}$$

with $\mathcal{Z}_p(i) = \tau_p \sum_{p \in P(i)} \exp(\alpha_{i,p} s_{i,p}/\tau_p) + \tau_n \sum_{n \in N(i)} \exp(\alpha_{i,n} s_{i,n}/\tau_n)$. We further provide an illustrative example in Figure 4 and a brief theoretical analysis on *PCCL*.

2.4 Theoretical Analysis of *PCCL*

We theoretically show that *PCCL* is an extension to various metric learning based loss functions. As *PCCL* is directly extended from the supervised contrastive loss [7, 14] by adding *pair-wise relaxation factors*, it is trivial to see that the supervised contrastive loss is a special case of *PCCL* with $\alpha_{i,p} = \alpha_{i,n} = 1$ and $\tau_p = \tau_n$.

Next, we consider the triplet loss [35]. Assume that there are only one positive and one negative samples for each anchor. We simplify $\mathcal{L}_{PCCL}(i)$ as follows:

$$\begin{aligned}\mathcal{L}_{PCCL}(i)' &= \log(1+\exp(\frac{\alpha_{i,p}}{\tau_p}s_{i,p} - \frac{\alpha_{i,n}}{\tau_n}s_{i,n})) \\ &= \log(1+\exp(\frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}s_{i,p} - \alpha_{i,n}s_{i,n}))\end{aligned} \tag{10}$$

If we set a small value for τ_n (close to 0, which is the case as shown in the experiments), then the value of $\frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}s_{i,p} - \alpha_{i,n}s_{i,n})$ is large. We have:

$$\begin{aligned}\mathcal{L}_{PCCL}(i)' &\approx \frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}s_{i,p} - \alpha_{i,n}s_{i,n}) \\ &= \frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}\vec{z}_i^T\vec{z}_p - \alpha_{i,n}\vec{z}_i^T\vec{z}_n) \\ &\propto -\frac{1}{2\tau_n}(\frac{\alpha_{i,p}}{\xi}\|\vec{z}_i - \vec{z}_p\|^2 - \alpha_{i,n}\|\vec{z}_i - \vec{z}_n\|^2)\end{aligned} \tag{11}$$

Approximately, the problem of minimizing $\mathcal{L}_{CCL}(i)'$ is equivalent of optimizing the loss function $\mathcal{L}_{TL}(i)$ (with the margin omitted) as follows:

$$\mathcal{L}_{TL}(i) = \alpha_{i,n}\|\vec{z}_i - \vec{z}_n\|^2 - \frac{\alpha_{i,p}}{\xi}\|\vec{z}_i - \vec{z}_p\|^2 \tag{12}$$

which is the triplet loss with the positive and negative pair-wise weights to be $\frac{\alpha_{i,p}}{\xi}$ and $\alpha_{i,n}$, respectively. Therefore, the triplet loss has a close connection to *PCCL*.

As for the N-pair loss [40], we consider the situation where there is only one positive sample and multiple negative samples for each anchor. We re-write $\mathcal{L}_{PCCL}(i)$ as:

$$\mathcal{L}_{PCCL}(i)'' = \log(1 + \sum_{n\in N(i)} \exp(\frac{\alpha_{i,p}}{\tau_p}s_{i,p} - \frac{\alpha_{i,n}}{\tau_n}s_{i,n})) \tag{13}$$

By setting $\frac{\alpha_{i,p}}{\tau_p} = 1$ and $\frac{\alpha_{i,n}}{\tau_n} = 1$, we simplify *PCCL* into the N-pair loss. We can see that *PCCL* combines the advantages of both supervised learning and metric learning, specifically assigning different costs to easy and hard cases.

2.5 Auxiliary Masked Language Modeling

As the learning objective of *PCCL* is significantly different from the MLM task, minimizing $\mathcal{L}_{PCCL}(i)$ only may result in the *catastrophic forgetting* of the pre-training knowledge. Similar to Schick and Schütze [33, 34], we treat MLM as an auxiliary task during few-shot learning to improve the model performance on previously unseen instances. Denote the *sample-wise* MLM loss as $\mathcal{L}_{MLM}(i)$. The *sample-wise* overall loss function $\mathcal{L}(i)$ can be written as follows:

$$\mathcal{L}(i) = \lambda \cdot \mathcal{L}_{PCCL}(i) + (1-\lambda)\cdot \mathcal{L}_{MLM}(i) \tag{14}$$

where λ is a pre-defined balancing hyper-parameter. In Figure 2, we apply the auxiliary MLM task to "[TMSK]" tokens and *PCCL* to "[OMSK]" tokens, separately. This practice can be viewed as performing *task-specific continual pre-training* [43] and few-shot learning at the same time. The overall training algorithm of *CP-Tuning* is summarized in Algorithm 1.

Algorithm 1 Training Algorithm of *CP-Tuning*

```
1: while not converge do
2:     Sample a batch of training data $\mathcal{B}$ from $\mathcal{D}$;
3:     for each training sample $i \in \mathcal{B}$ do
4:         Augment sample $i$ by the task-invariant prompt;
5:     end for
6:     Feed the batch $\mathcal{B}$ (with prompts) to the PLM to generate $\vec{z}_i$
       for each $i \in \mathcal{B}$;
7:     for each training sample $i \in \mathcal{B}$ do
8:         Retrieve the positive set $P(i)$ and the negative set $N(i)$;
9:     end for
10:    Compute the batch loss function $\mathcal{L} = \sum_{i\in\mathcal{B}} \mathcal{L}(i)$;
11:    Update the model parameters $\mathcal{M}$ and the representations of
       prompts by minimizing the loss function $\mathcal{L}$;
12: end while
13: return the fine-tuned few-shot learner.
```

2.6 Model Inference

During the model inference time, because we do not tune the "[CLS]" prediction head, we directly take the embedding $\vec{z}_i$ of a testing instance i to generate the class label $\hat{y}_i$ by comparing $\vec{z}_i$ against the k-nearest neighbors in the few-shot training set. When *CP-Tuning* is applied to larger training sets, for better scalability, the label $\hat{y}_i$ is predicted by:

$$\hat{y}_i = \text{argmax}_{c\in C}\vec{z}_i^T \cdot \vec{z}_c \tag{15}$$

where C is the collection of the class labels, and $\vec{z}_c$ is the prototype embedding of the class $c \in C$ (*i.e.,* the averaged embedding of all training instances with the class label as c). Hence, this practice is closely in line with *prototypical networks* [12, 38].

3 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate *CP-Tuning* and compare it against state-of-the-arts. We also analyze our approach in various aspects to show its superiority.

3.1 Evaluation Datasets

In the experiments, we evaluate *CP-Tuning* over various NLP tasks and datasets. Specifically, we focus on two NLP tasks frequently used in modern IR systems, namely sentiment analysis and sentence matching. To prove that our method can be applied to other NLP tasks as well, we also consider the tasks of natural language inference and subjectivity classification. The goals of these tasks and the corresponding datasets are listed as follows:

- **Sentiment Analysis**: predicting the sentiment polarity of review comments (SST-2 [39], MR [9] and CR [27]);
- **Sentence Matching**: predicting the semantic equivalence of sentences in Web corpora and questions in online forums, respectively (MRPC [5] and QQP [2]);

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

Task	Dataset	#Training	#Testing
Sentiment Analysis	SST-2	6,920	872
	MR	8,662	2,000
	CR	1,775	2,000
Sentence Matching	MRPC	3,668	408
	QQP	363,846	40,431
Natural Language Inference	QNLI	104,743	5,463
	RTE	2,490	277
Subjectivity Classification	SUBJ	8,000	2,000

Table 2: Dataset statistics. We only sample $K \times |C|$ instances from the original training sets to form the few-shot training and development sets.

- **Natural Language Inference (NLI)**: predicting the semantic relations between two sentences (QNLI [31] and RTE [1]);
- **Subjectivity Classification**: predicting whether the contents of documents are subjective or objective (SUBJ [26]).

The dataset statistics are summarized in Table 2. For few-shot learning, we follow the evaluation protocols in Gao et al. [6] to sample few-shot training and development sets from the original full training sets. In default, we set $K = 16$ and measure the average performance in terms of accuracy across 5 different randomly sampled training and development splits. Hence, the performance of *CP-Tuning* can be rigorously evaluated with a minimal influence of random seeds or datasets.

In addition, we consider the situation where the few-shot training sets are unbalanced w.r.t. the number of training samples for each class. The settings and results are elaborated in Section 3.5.

3.2 Experimental Settings

To verify that *CP-Tuning* is effective across different PLMs, we test the large version of two popular PLMs from Hugging Face Models[3], namely RoBERTa [23] and ALBERT [18]. We consider the following methods as strong baselines:

- **Fine-tuning** [4]: it is the standard fine-tuning approach by utilizing the "[CLS]" head of the PLM.
- **PET** [33, 34] [4]: it employs manually-crafted, discrete prompt templates and verbalizers for few-shot learning.
- **LM-BFF** [6] [5]: it generates templates and label words automatically. In our work, three settings of LM-BFF are used for comparison, where "Auto T", "Auto L" and "Auto T+L" refer to the model with automatically generated templates, label words and both, respectively.
- **P-tuning** [22] [6]: it employs continuous prompt embeddings generated by light-weight neural nets and fixed verbalizers for few-shot learning.
- **WARP** [8] [7]: it leverages continuous prompts to improve the model performance in fine-tuning scenarios. Specifically, it learns task-specific word embeddings concatenated to input texts, which guide the PLM to solve the specific task.

[3]https://huggingface.co/models
[4]https://github.com/timoschick/pet
[5]https://github.com/princeton-nlp/LM-BFF
[6]https://github.com/THUDM/P-tuning
[7]https://github.com/YerevaNN/WARP

Specifically, PET, LM-BFF, P-tuning and WARP are recent state-of-the-art approaches for prompt-based few-shot learning. As the experimental settings of PET, LM-BFF, P-tuning and WARP are different, in order to conduct a rigorous comparison, we re-produce the results based on their open-source codes. Hence, it is noted that the results reported in our work are slightly different from their original papers.

Our own *CP-Tuning* algorithm is implemented in PyTorch and run with NVIDIA V100 GPUs. In default, we set $\tau_p = 2$, $\tau_n = 1$ (with $\xi = 2$), $\lambda = 0.5$, $m = 0.3$ and $k = 3$, and also tune the parameters over the few-shot development sets. The model is trained with the Adam optimizer [16], with the learning rate and the batch size tuned around $\{1e-5, 3e-5, 5e-5\}$ and $\{4, 8, 16, 32\}$, respectively. The optimization process of the auxiliary MLM task is the same as in PET. We also study how the change of some important hyper-parameters affect the overall performance.

3.3 Overall Performance Comparison

The experimental results of *CP-Tuning* and all baselines on eight testing sets for few-shot learning are presented in Table 3. From the experimental results, we can draw the following conclusions:

- Prompt-based methods (such as PET, LM-BFF and P-tuning) outperform standard fine-tuning by a large margin. This shows that prompts are highly useful for few-shot learning over PLMs. Based on our re-production results, LM-BFF (with different settings) and P-tuning have similar performance, while PET produces slightly lower performance. As for WARP, it does not outperforms other three prompt-based methods. The most possible cause is that it does not leverage the MLM head of the PLM for prediction in the few-shot learning setting.
- In the experiments, we employ two PLMs to evaluate the effectiveness of our approach, namely RoBERTa and ALBERT. We observe that RoBERTa outperforms ALBERT, regardless of which learning algorithm is chosen. This is expected as the language modeling abilities of RoBERTa are better than ALBERT due to the larger pre-training data and parameter size. We can also see the our approach is highly general and can be effectively applied to any BERT-style PLMs.
- The performance gains of *CP-Tuning* over all the testing sets are consistent, compared to all the state-of-the-art methods. Overall, the average improvement is around 3% over the two PLMs. It can be seen that even without task-specific prompts and verbalizers, *CP-Tuning* is capable of producing high-accuracy models with few training instances.
- We further conduct *single-tailed, paired t-tests* to compare the accuracy scores on all tasks produced by *CP-Tuning* against LM-BFF and P-tuning. Experimental results show that the improvement of *CP-Tuning* is statistically significant (with the p-value $p < 0.05$).

3.4 Detailed Model Analysis

We further study how *CP-Tuning* improves the model performance in various aspects. Here, we treat SST-2, MR, MRPC and QQP as pilot tasks to explore our method. The underlying PLM is uniformly set to be RoBERTa-large.

Backbone	Method	Sentiment Analysis SST-2	MR	CR	Sentence Matching MRPC	QQP	NLI QNLI	RTE	Subjectivity SUBJ	Avg.
RoBERTa	Standard Fine-tuning	78.62	76.17	72.48	64.40	63.01	62.32	52.28	86.82	69.51
	PET	92.06	87.13	87.13	66.23	70.34	64.38	65.56	91.28	78.01
	LM-BFF (Auto T)	90.60	87.57	90.76	66.72	65.25	68.87	65.99	91.61	78.42
	LM-BFF (Auto L)	90.55	85.51	91.11	67.75	70.92	66.22	66.35	90.48	78.61
	LM-BFF (Auto T+L)	91.42	86.84	90.40	66.81	61.61	61.89	66.79	90.72	77.06
	P-tuning	91.42	87.41	90.90	71.23	66.77	63.42	67.15	89.10	78.43
	WARP	58.80	55.25	55.55	65.74	65.80	52.29	60.07	65.59	59.89
	CP-Tuning	**93.35**	**89.43**	**91.57**	**72.60**	**73.56**	**69.22**	**67.22**	**92.27**	**81.24**
ALBERT	Standard Fine-tuning	63.98	64.90	71.50	56.78	59.32	53.48	52.14	80.54	62.83
	PET	87.11	81.47	88.32	57.21	66.16	55.32	61.85	83.28	72.59
	LM-BFF (Auto T)	82.60	83.23	88.48	**64.04**	60.28	59.42	60.42	84.67	72.75
	LM-BFF (Auto L)	86.83	83.02	89.12	63.43	59.49	56.86	57.33	88.08	73.02
	LM-BFF (Auto T+L)	84.40	82.75	89.52	62.48	56.48	57.69	61.09	88.44	72.85
	P-tuning	85.42	84.32	82.35	58.76	57.46	58.97	55.07	84.32	70.83
	WARP	66.63	65.59	72.34	63.48	58.20	57.45	53.86	62.41	62.49
	CP-Tuning	**89.63**	**84.68**	**90.39**	63.52	**71.05**	**62.02**	**61.92**	**89.02**	**76.52**

Table 3: Comparison between *CP-Tuning* and baseline methods over the testing sets in terms of accuracy (%).

Method/Task	SST-2	MR	MRPC	QQP
Full Implement.	**93.35**	**89.43**	**72.60**	**73.56**
w/o. auxiliary MLM	91.35	86.67	71.96	72.47
w/o. $\alpha_{i,p}$ and $\alpha_{i,n}$	92.50	88.59	68.28	69.32
w/o. similarity avg.	92.04	86.37	67.11	69.14

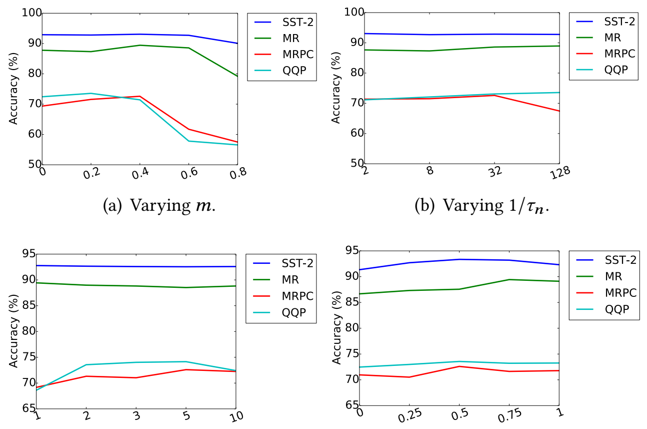
Table 4: Ablation study of *CP-Tuning* on four tasks in terms of accuracy (%). "Full Implement." refers to the full implementation of our method. The lowest accuracy scores over each dataset are printed underlined.

3.4.1 Ablation Study. The ablation results of *CP-Tuning* are shown in Table 4. Here, "w/o. auxiliary MLM" refers to the variant of *CP-Tuning* without the auxiliary MLM task; "w/o. $\alpha_{i,p}$ and $\alpha_{i,n}$" refers to *CP-Tuning* without the pair-wise relaxation factors; and "w/o. similarity averaging" refers to the model setting where we only consider one positive and one negative instance for each anchor (similar to the standard triplet loss). To specify, the contrastive loss for "w/o. $\alpha_{i,p}$ and $\alpha_{i,n}$" is $\mathcal{L}_{CL}(i)$, while the *sample-wise* loss function for "w/o. similarity averaging" is:

$$-\log \frac{\exp(\alpha_{i,p} \cdot s_{i,p}/\tau_p)}{\exp(\alpha_{i,p} \cdot \tau_p/s_{i,p}) + \exp(\alpha_{i,n} \cdot \tau_n/s_{i,n})} \quad (16)$$

From the results we can see that all three techniques contribute to the overall accuracy improvement. Specifically, the auxiliary MLM task has the most influence over SST-2, while similarity averaging contributes the most to the improvement on the remaining three datasets. It shows that all techniques proposed by this work positively contribute to the improvement.

In addition, we have a relatively surprising finding on the auxiliary MLM task. The performance drops by a large margin when we remove the MLM task for SST-2 and MR. This is because the few-shot learning ability of PLMs is largely based on the utilization of pre-trained knowledge learned by the MLM task. In *CP-Tuning*, the *PCCL* objective is significantly different from MLM, hence optimizing *PCCL* alone may lead to the catastrophic forgetting of the MLM knowledge acquired during the pre-training stage. We suggest that the auxiliary MLM task in *CP-Tuning* is vital for obtaining the high performance.



(a) Varying m. (b) Varying $1/\tau_n$.

(c) Varying ξ. (d) Varying λ.

Figure 5: Parameter analysis on hyper-parameters over four datasets in terms of accuracy.

3.4.2 Parameter Analysis. We also show how some of the important hyper-parameters in *CP-Tuning* affect the performance over the four datasets. The results are shown in Figure 5. We can see the the trends are almost consistent across all the datasets. The optimal setting of the margin m is around 0.2. As for the temperature, the optimal value of τ_n is around 1/8 to 1/32, which is different from other works where the default temperature is 1. This is probably due to the fact that we compute the total scores $\alpha_{i,p} \cdot s_{i,p}/\tau_p$ and

Batch Size/Task	SST-2	MR	MRPC	QQP
4	92.80	**89.43**	**72.60**	71.84
8	92.75	87.98	71.42	72.92
16	**93.35**	88.50	72.20	**73.56**
32	93.28	89.32	72.42	73.18

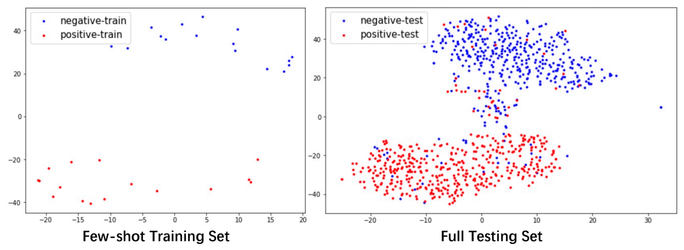
Table 5: Analysis of the batch size. The results show that our approach is highly effective even with a small batch size.

$\alpha_{i,n} \cdot s_{i,n}/\tau_n$, which are different from those in other works in contrastive learning. Nevertheless, the performance of *CP-Tuning* is not very sensitive to the choice of the temperature, proving that *CP-Tuning* is highly general for real-world applications.

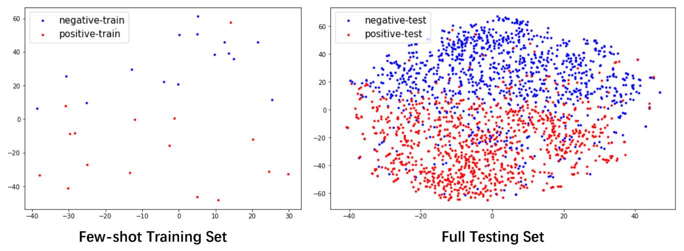
We further tune the value of ξ. As seen in the figure, for sentence-pair tasks, the optimal ξ is between 2 to 5, while easier single sentence tasks are not sensitive to this hyper-parameter. We also try using the prototype embeddings $\vec{z}_c$ for model inference, of which the results are similar. We suggest that when *CP-Tuning* is applied to large datsets, it is suitable to predict the class label $\hat{y}_i$ by $\text{argmax}_{c\in C}\vec{z}_i^T \cdot \vec{z}_c$ for better scalability. In PET [33, 34], the auxiliary MLM task is applied with $\lambda = 1e{-}4$, which is sufficiently small. In contrast to their work, we suggest that the optimal value of λ is in the range between 0.5 to 0.75.

3.4.3 Analysis of Batch Size for Contrastive Learning. Previous studies have shown that the contrastive learning process usually require a large batch size for effective representation learning [11]. For few-shot prompt-based learning, as we use large-scale PLMs as model backbones, the GPU memory consumption can be a significant challenge when the batch size is large. Here, we show the performance of our approach when the batch size varies, illustrated in Table 5. The results show that our approach does not need a large batch size for contrastive training. We suggest that our contrastive learning setting is *fully supervised*, with strong label signals provided. Hence, the data in a small batch can already guide the model to learn the differences between positive and negative classes.

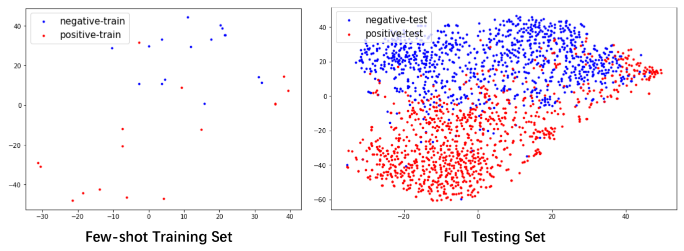
3.4.4 Visualizations. To show that the generated "[OMSK]" embeddings are separable for text classification, we plot the embeddings of the few-shot training and testing data in SST-2, CR and SUBJ. The results are illustrated in Figure 6. The underlying dimension reduction and visualization algorithm is t-SNE [45]. As seen, even reduced in two dimensions, most of the embeddings in the testing set are clearly separated, with the only $N \times K$ training samples available. Additionally, the embeddings in the few-shot training set are widely spread, showing the generalization of our algorithm.



(a) Dataset: SST-2.



(b) Dataset: CR.



(c) Dataset: SUBJ.

Figure 6: Visualizations of "[OMSK]" embeddings of the few-shot traing sets and the full testing sets of SST-2, CR and SUBJ by t-SNE. (Best viewed in color.)

Method/Task	SST-2	MR	MRPC	QQP
PET	87.25	83.44	64.61	58.82
LM-BFF	88.10	83.51	65.98	59.19
P-tuning	87.92	83.20	66.64	61.27
CP-Tuning	**91.25**	**86.52**	**70.12**	**65.52**

Table 6: Testing results of *CP-Tuning* and baseline methods for unbalanced few-shot learning in terms of accuracy (%).

3.5 Learning with Unbalanced Datasets

In the literature, few-shot learning is usually formulated as an *N-way-K-shot* problem. However, it may not be the case in real-world applications. In this set of experiments, we consider the situation where the few-shot training set is unbalanced w.r.t the numbers of training instances for each class. Following previous experiments, four binary classification tasks are used for experimental evaluation, namely SST-2, MR, MRPC and QQP. In each few-shot training set, we assume there are 8 and 24 training instances of the two classes, instead of setting $K = 16$ for all the classes. The few-shot development sets are of the same size as the training sets.

We compare *CP-Tuning* against three strong baselines for few-shot learning (*i.e.,* PET, LM-BFF and P-tuning). The results are shown in Table 6. As seen, *CP-Tuning* consistently outperforms these baselines by a large margin. The improvement rates are also larger than those in standard few-shot learning scenarios (as reported in Table 3). This is because the contrastive learning technique in *CP-Tuning* focuses on learning the distinctions between positive and negative samples, instead of tuning the MLM head only (as in previous approaches). Therefore, it is better at dealing with unbalanced few-shot learning scenarios.

Task/Method	*CP-Tuning*		PET	
	Acc.	Std.	Acc.	Std.
SST-2	92.91*	0.56*	91.28	1.38
MR	88.38*	1.46	86.28	1.70
MRPC	71.80*	2.20*	65.73	5.08
QQP	73.84*	2.16*	66.61	5.22

Table 7: Method comparison with five sets of prompts in terms of averaged accuracy (%) and standard deviation. * refers to statistical significance of higher accuracy and lower deviation at the 95% confidence interval.

3.6 Study on Task-invariance of Prompts

In *CP-Tuning*, we initialize prompt embeddings as the pre-trained representations of the universal task-invariant prompts and utilize the *verbalizer-free* mechanism to avoid the manual prompt engineering process. In the following experiments, we aim to study whether *CP-Tuning* is capable of generating more stable and accurate results using *universal task-invariant prompts*, compared to the non-contrastive baseline (*i.e.,* PET [33, 34]).

We consider two review sentiment analysis datasets: SST-2 and MR, as well as two paraphrase datasets: MRPC and QQP. Five prompt settings are employed: the *universal task-invariant prompts* used in *CP-Tuning* and the manually designed prompts used in PET [33, 34]. In Table 7, we present the averaged accuracy and its standard deviation of *CP-Tuning* and PET, under five different prompt settings. We can see that compared to PET, *CP-Tuning* has a higher accuracy and a lower deviation when the prompts change. Hence, our task-invariant prompts are highly effective.

This finding is different from previous works, showing that *CP-Tuning* is not sensitive to different prompts. Hence, we suggest learning with task-invariant prompts and no verbalizers are a desirable setting that reduces the amount of human labor. Additionally, during the learning process, prompt embeddings can be automatically adapted to fit specific tasks.

4 RELATED WORK

In this section, we summarize related work on PLMs, prompting PLMs for few-shot learning and contrastive learning. We also discuss how our work improves previous works from various aspects.

4.1 Deep Contrastive Learning

Contrastive learning [11] aims to learn an embedding space in which similar instances have similar embeddings while dissimilar instances fall apart. Contrastive learning can be either supervised or unsupervised, and achieves good performance on computer vision tasks. In the literature, several contrastive learning objectives have been proposed, such as the triplet loss [35], the N-pair loss [40], InfoNCE [44] and the supervised contrastive loss [14]. Due to its effectiveness, contrastive learning has been applied to various NLP tasks, *e.g.,* sentence representation [7, 15], text summarization [46], aspect detection [36], machine translation [48], commonsense reasoning [17]. To our knowledge, *CP-Tuning* is the first to apply contrastive learning to prompt-based few-shot learning, to make the instances of different classes more separable.

4.2 Pre-trained Language Models

With the two-stage pre-training and fine-tuning paradigm, PLMs have achieved significant improvements on various NLP tasks, frequently applied in IR systems. Readers can refer to the survey for details [29]. Among these PLMs, ELMo [28] learns the contextual word representations by self-supervised pre-training using bidirectional LSTMs as encoders. BERT [4] is probably the most popular model, which learns the contextual representations of tokens by layers of transformer encoders. Other PLMs based on the transformer encoder architecture include ALBERT [18], Transformer-XL [3], XLNet [49], StructBERT [47], Big Bird [50] and many others. Apart from the encoder-based PLMs, the encoder-decoder architecture is used in T5 [30] and other PLMs for text generation. The GPT model series [2] employs the auto-regressive decoder architecture for zero-shot text generation. Our framework is highly general w.r.t. PLMs because it can be applied to any BERT-style PLMs with high accuracy. As the neural architectures are not our major focus, we do not elaborate.

4.3 Prompting PLMs for Few-shot Learning

With the prevalence of GPT-3 [2], prompting PLMs for few-shot learning has become a new, popular learning paradigm. A recent survey can be found in Liu et al. [20]. To name a few, PET [33, 34] turns text classification into cloze-style problems and use manually-defined prompts to provide additional task guidance. To facilitate automatic prompt discovery, Gao et al. [6] generate prompts and label words from the T5 model [30]. In addition, Jiang et al. [13] also mine high-performing prompts from the training corpus. AutoPrompt [37] employs gradient searching to detect prompts from the text corpus. However, these approaches focus on discrete prompts only and the detected prompts may not be human-understandable.

For continuous prompts, P-tuning [22] learns continuous prompt embeddings with differentiable parameters for GPT-based models. The update version, P-tuning v2 [21] extends P-tuning to different scales of PLMs and NLP tasks. Prefix-tuning [19] extends the usage of continuous prompts for text generation tasks. Min et al. [25] propose a noisy channel model for prompt learning over multiple prompts. WARP [8] leverages continuous prompts to improve the model performance in fine-tuning scenarios. Knowledgeable prompt-tuning [10] optimizes the verbalizer construction process by integrating the knowledge from knowledge bases. Our work further applies contrastive learning to making the few-shot learner fully verbalizer-free, without defining task-specific prompts.

5 CONCLUSION AND FUTURE WORK

In this work, we present an end-to-end *Contrastive Prompt Tuning* (*CP-Tuning*) framework that enables few-shot learning for PLMs without designing any task-specific prompts and verbalizers. In *CP-Tuning*, we employ task-invariant continuous prompt encoding and the *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) to train the model. Specifically, task-invariant prompt encoding eases the process of hand-crafting prompts, while *PCCL* learns to distinguish different classes and makes the decision boundary smoother by assigning different costs to easy and hard cases. We also give a theoretical analysis on *PCCL*. Experiments over eight public datasets show that *CP-Tuning* consistently outperforms state-of-the-art methods.

Future work of *CP-Tuning* includes: i) extending the *CP-Tuning* framework to other tasks such as named entity recognition, machine reading comprehension, text ranking and text generation; ii) combining *CP-Tuning* with transfer learning to improve the model performance in low-resource scenarios.

REFERENCES

[1] Roy Bar-Haim, Ido Dagan, and Idan Szpektor. 2014. Benchmarking Applied Semantic Inference: The PASCAL Recognising Textual Entailment Challenges. In *Language, Culture, Computation. Computing - Theory and Technology*, Vol. 8001. 409–424.

[2] 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*.

[3] 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*. 2978–2988.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[5] William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *IWP@IJCNLP*.

[6] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *ACL/IJCNLP*. 3816–3830.

[7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *CoRR* abs/2104.08821 (2021).

[8] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *ACL/IJCNLP*. 4921–4933.

[9] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*. 168–177.

[10] 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 (2021).

[11] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A Survey on Contrastive Self-supervised Learning. *CoRR* abs/2011.00362 (2020).

[12] Zhong Ji, Xingliang Chai, Yunlong Yu, Yanwei Pang, and Zhongfei Zhang. 2020. Improved prototypical networks for few-Shot learning. *Pattern Recognit. Lett.* 140 (2020), 81–87.

[13] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know. *Trans. Assoc. Comput. Linguistics* 8 (2020), 423–438.

[14] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. In *NeurIPS*.

[15] Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021. Self-Guided Contrastive Learning for BERT Sentence Representations. In *ACL/IJCNLP*. 2528–2540.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Tassilo Klein and Moin Nabi. 2020. Contrastive Self-Supervised Learning for Commonsense Reasoning. In *ACL*. 7517–7523.

[18] 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*.

[19] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL/IJCNLP*. 4582–4597.

[20] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *CoRR* abs/2107.13586 (2021).

[21] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *CoRR* abs/2110.07602 (2021).

[22] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *CoRR* abs/2103.10385 (2021).

[23] 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 (2019).

[24] Craig Macdonald, Nicola Tonellotto, and Sean MacAvaney. 2021. IR From Bag-of-words to BERT and Beyond through Practical Experiments. In *CIKM*. 4861.

[25] Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Noisy Channel Language Model Prompting for Few-Shot Text Classification. *CoRR* abs/2108.04106 (2021).

[26] Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *ACL*. 271–278.

[27] Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL*. 115–124.

[28] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*. 2227–2237.

[29] 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 (2020).

[30] 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 (2020), 140:1–140:67.

[31] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*. 2383–2392.

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

[33] Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *EACL*. 255–269.

[34] Timo Schick and Hinrich Schütze. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *NAACL*. 2339–2352.

[35] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. 815–823.

[36] Tian Shi, Liuqing Li, Ping Wang, and Chandan K. Reddy. 2021. A Simple and Effective Self-Supervised Contrastive Learning Framework for Aspect Detection. In *AAAI*. 13815–13824.

[37] 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*. 4222–4235.

[38] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *NIPS*. 4077–4087.

[39] 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*. 1631–1642.

[40] Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *NIPS*. 1849–1857.

[41] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. 2020. Circle Loss: A Unified Perspective of Pair Similarity Optimization. In *CVPR*. 6397–6406.

[42] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation. *CoRR* abs/2107.02137 (2021).

[43] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. In *AAAI*. 8968–8975.

[44] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018).

[45] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 2605 (2008), 2579–2605.

[46] Hong Wang, Xin Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Self-Supervised Learning for Contextualized Extractive Summarization. In *ACL*. 2221–2227.

[47] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding. In *ICLR*.

[48] Zonghan Yang, Yong Cheng, Yang Liu, and Maosong Sun. 2019. Reducing Word Omission Errors in Neural Machine Translation: A Contrastive Learning Approach. In *ACL*. 6191–6196.

[49] 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*. 5754–5764.

[50] 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*.