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

Ziyun Xu
School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA
ziyunx@andrew.cmu.edu

Chengyu Wang
Alibaba Group
Hangzhou, China
chengyu.wcy@alibaba-inc.com

Minghui Qiu
Alibaba Group
Hangzhou, China
minghui.qmh@alibaba-inc.com

Fuli Luo
Alibaba Group
Hangzhou, China
lf259702@alibaba-inc.com

Runxin Xu
Key Laboratory of Computational Linguistics, Peking University
Beijing, China
runxin.xu@gmail.com

Songfang Huang
Alibaba Group
Hangzhou, China
songfang.hsf@alibaba-inc.com

Jun Huang
Alibaba Group
Hangzhou, China
huangjun.hj@alibaba-inc.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, an 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 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, Chengyu Wang, Minghui Qiu, Fuli Luo, Runxin Xu, Songfang Huang, and Jun Huang. 2023. Making Pre-trained Language Models End-to-end Few-shot Learners with Contrastive Prompt Tuning. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM ’23), February 27-March 3, 2023, Singapore, Singapore. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570398

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 [19, 43, 52], which have been extensively used in Information Retrieval (IR) systems for tasks such as content analysis, question matching and question answering [25]. 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 [34, 35] and many others. Scao and Rush [33] 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 [23], 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. [22, 23] 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.

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. Specifically, our approach consists of two major techniques: i) 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 [23]). Specially, we initialize continuous 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. ii) 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.

![Figure 1: Comparison between classical prompt-based fine-tuning and CP-Tuning for review sentiment analysis.](image)

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. ²

2 CP-TUNING: PROPOSED APPROACH

2.1 Overview of CP-Tuning

Let \( D \) 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. Denote \( M \) 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 \( M \) based on \( D \) that can effectively generalize to previously unseen 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 [23], 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 [34, 35]. 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.

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.

²The implementations are released in the EasyNLP framework [47, 48]. URL: https://github.com/alibaba/EasyNLP

![Figure 2: An overview of the CP-Tuning framework.](image)
Table 1: Examples of inputs and processed token sequences for CP-Tuning (first and second lines of input sequence). [PRO] 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.

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

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.

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: i) [PRO] (Prompt): the placeholder for continuous prompt embeddings; ii) [TMSK] (Token Mask): the token mask of the input texts for optimizing the auxiliary MLM loss; iii) [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.

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.

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 [23, 34, 35]. 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.

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 \( \mathcal{B} \), i.e., \( P(i) = \{ j | j \neq i, y_j = y_i, j \in \mathcal{B} \} \), \( N(i) = \{ j | j \neq y_i, j \in \mathcal{B} \} \).

Let \( z_i \) 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 \( s_{i,n} = z_i^T \cdot z_n \) where \( p \in P(i), n \in N(i) \) and minimize the between-class similarity \( s_{i,n} = z_i^T \cdot z_n \) where \( n \in N(i) \). Following previous supervised contrastive learning models [7, 15], it is straightforward to derive the sample-wise contrastive loss:

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

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

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

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.
In this way, our few-shot learner will be less likely to over-fit to learning based loss functions. As and a brief theoretical analysis on \( \frac{\partial}{\partial \alpha} \) temperatures easy cases. Another empirical setting is that we use separate margin between to \( \alpha \) and propose a new loss function named Pair-wise Cost-sensitive Contrastive Loss (PCCL) as follows:

\[
L_{\text{PCCL}}(i) = -\sum_{p \in P(i)} \log \frac{\exp(\alpha_{i,p} \cdot s_{i,p} / \tau_p)}{Z(i)}
\]

where \( Z(i) \) is the normalization factor: \( Z(i) = \sum_{p \in P(i)} \exp(\alpha_{i,p} \cdot s_{i,p}) + \sum_{n \in N(i)} \exp(\alpha_{i,n} \cdot s_{i,n}) \).

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

\[
\alpha_{i,p} = \max\{0, 1 + m - s_{i,p}\}, \quad \alpha_{i,n} = \max\{0, s_{i,n} + m\}
\]

Comparing to the original \( L_{\text{CL}}(i) \), 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 \( \tau_p \) and \( \tau_n \) for within-class and between-class similarities, instead of a uniform temperature \( \tau \). 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: \( \alpha_{i,p} \cdot s_{i,p} = \alpha_{i,p} \cdot s_{i,p} \cdot \gamma_{i,p} = \alpha_{i,p} \cdot s_{i,p} \) where \( \alpha_{i,p} = \max\{0, \frac{1}{\xi} (1 + m - s_{i,p})\} \). 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 \( \frac{\partial L_{\text{PCCL}}(i)}{\partial \vec{z}_i} \) and updating the PLM. 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 \( \frac{\partial}{\partial \alpha} \) is directly extended from the supervised contrastive loss [7, 15] 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 [36]. Assume that there are only one positive and one negative samples for each anchor. We simplify \( L_{\text{PCCL}}(i) \) as follows:

\[
L_{\text{PCCL}}(i) = \log(1 + \exp(\frac{\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n}}{\tau_n})) = \log(1 + \exp(-\frac{1}{\tau_n} (\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n})))
\]

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

\[
L_{\text{PCCL}}(i) \approx \frac{1}{\tau_n} \frac{\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n}}{\exp(\frac{\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n}}{\tau_n}) - 1} = \frac{1}{\tau_n} \frac{\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n}}{\exp(\frac{\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n}}{\tau_n}) - 1}
\]

\[
\alpha = \frac{1}{2\tau_n} (\frac{\alpha_{i,p}}{\xi} ||\vec{z}_i - \vec{z}_p||^2 - \alpha_{i,n} ||\vec{z}_i - \vec{z}_n||^2)
\]

Approximately, the problem of minimizing \( L_{\text{PCCL}}(i) \) is equivalent to optimizing the loss function \( L_{\text{TL}}(i) \) (with the margin omitted) as follows:

\[
L_{\text{TL}}(i) = \alpha_{i,n} ||\vec{z}_i - \vec{z}_n||^2 - \alpha_{i,p} ||\vec{z}_i - \vec{z}_p||^2
\]

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 [41], we consider the situation where there is only one positive sample and multiple negative examples for each anchor. We re-write \( L_{\text{PCCL}}(i) \) as:

\[
L_{\text{PCCL}}(i) = \log(1 + \sum_{n \in N(i)} \exp(\frac{\alpha_{i,p} \cdot s_{i,p} - \alpha_{i,n} \cdot s_{i,n}}{\tau_n}))
\]

By setting \( \alpha_{i,p} = 1 \) and \( \alpha_{i,n} = 1 \), 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 \( L_{\text{PCCL}}(i) \) only may result in the catastrophic forgetting of the pre-training knowledge. Similar to Schick and Schütze [34, 35], 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 \( L_{\text{MLM}}(i) \). The sample-wise overall loss function \( L(i) \) can be written as follows:

\[
L(i) = \lambda \cdot L_{\text{PCCL}}(i) + (1 - \lambda) \cdot L_{\text{MLM}}(i)
\]

where \( \lambda \) 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 [44] and few-shot learning at the same time.

### 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: \( \hat{y}_i = \text{argmax}_{c \in C} z_c^T \tilde{z}_c \) where \( C \) is the collection of the class labels, and \( \tilde{z}_c \) 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, 39].

3 EXPERIMENTS

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 [40], MR [9] and CR [28]);
- **Sentence Matching**: predicting the semantic equivalence of sentences in Web corpora and questions in online forums, respectively (MRPC [5] and QQP [8]);
- **Natural Language Inference (NLI)**: predicting relations between two sentences (QNLI [32] and RTE [1]);
- **Subjectivity Classification**: predicting whether the contents of documents are subjective or objective (SUBJ [27]).

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.

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, namely RoBERTa [24] and ALBERT [19]. 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** [34, 35]: it employs manually-crafted, discrete prompt templates and verbalizers for few-shot learning.
- **LM-BFF** [6]: 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** [23]: it employs continuous prompt embeddings generated by light-weight neural nets and fixed verbalizers for few-shot learning.
- **PET + CL and LM-BFF + CL** [13]: to our knowledge, it is the only concurrent work that leverages constructive learning for few-shot learning based on [6, 34].

As the experimental settings of PET, LM-BFF, P-tuning and WARP are different, in order to conduct a rigorous comparison, we reproduce the results based on their open-source codes. Hence, 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 [17], 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 2. 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 outperform 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.

- By comparing our work against a concurrent work [13], we can also see that our work outperforms theirs in both settings (i.e., PET+CL and LM-BFF+CL).

- We further conduct single-tailed, paired t-tests to compare the accuracy scores on all tasks produced by CP-Tuning against baselines. Experimental results show that the improvement of CP-Tuning is statistically significant (with the p-value \( p < 0.05 \)).

---

3https://www.quora.com/q/quoradata/
4https://huggingface.co/models
5https://github.com/timoschick/pet
6https://github.com/princeton-nlp/LM-BFF
7https://github.com/YerevaNN/WARP
Table 3: Ablation study of CP-Tuning on four tasks in terms of accuracy (%). "Full Implement." refers to the full implementation of our method.

| Method | Task | SST-2 | MR | MRPC | QQP |
|--------|------|-------|-----|------|-----|
| Full Implement. | | 93.35 | 99.43 | 72.60 | 73.56 |
| w/o. auxiliary MLM | | 91.35 | 86.67 | 71.96 | 72.47 |
| Ablations for PCCL | | | | | |
| w/o. \alpha_p and \alpha_n | | 92.50 | 88.59 | 69.28 | 69.32 |
| w/o. similarity avg. | | 92.04 | 86.37 | 67.11 | 69.14 |
| w/o. PCCL (entirely) | | 91.92 | 85.87 | 67.01 | 68.45 |
| Ablations for prompts | | | | | |
| w/ fixed prompts in PET | | 91.52 | 85.41 | 66.42 | 65.67 |
| w/ fixed prompts in LM-BFF | | 91.32 | 85.62 | 66.57 | 65.17 |

Table 2: Comparison between CP-Tuning and baseline methods over the testing sets in terms of accuracy (%). * denotes the results of the concurrent work [13] in their paper, using the same datasets and backbone.

| Backbone | Method | Sentiment Analysis | Sentence Matching | NLI | Subj |
|----------|--------|-------------------|------------------|-----|------|
|          | SST-2  | MR               | MRPC            | QNP | NLI  | RTE | Subjectivity | Average |
| 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 |
|          | L-BFF | 58.80 | 55.25 | 55.55 | 65.74 | 65.80 | 52.29 | 60.07 | 65.59 | 59.89 |
|          | CP-Tuning | 89.49 | 89.03 | 90.30 | 75.43 | 67.67 | 63.53 | 68.10 | 90.96 | 78.27 |
| ALBERT   | Standard Fine-tuning | 63.98 | 64.90 | 71.30 | 56.78 | 59.32 | 53.48 | 52.14 | 80.54 | 62.83 |
|          | PET    | 87.11 | 81.47 | 88.32 | 57.21 | 66.16 | 53.52 | 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 |
|          | L-BFF | 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 | 71.03 | 62.02 | 61.92 | 89.02 | 76.52 |

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.

3.4.1 Ablation Study. The ablation results of CP-Tuning are shown in Table 3. Here, "w/o. auxiliary MLM" refers to the variant of CP-Tuning without the auxiliary MLM task; "w/o. \alpha_p and \alpha_n" refers to CP-Tuning without the pair-wise relaxation factors; "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); "w/o. PCCL (entirely)" refers to the fixed verbalizer setting (same as PET). To be more specific, the contrastive loss for "w/o. \alpha_p and \alpha_n" is \mathcal{L}_{CL}(1), while the sample-wise loss function for "w/o. similarity averaging" is:

\-
\log \exp(\alpha_{tp} \cdot s_p / \tau_p) / \exp(\alpha_{tn} \cdot s_n / \tau_n).

For the ablation of our prompt design. We also implement two variants that use the fixed, discrete prompts in PET and LM-BFF, respectively, instead of our proposed prompting technique. From the results we can see that all the techniques contribute to the overall accuracy improvement. Specifically, the auxiliary MLM task has the most influence over SST-2, while PCCL 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.

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 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 \(\tau\) is around 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 \(\alpha_p \cdot s_p / \tau_p + \alpha_n \cdot s_n / \tau_n\), 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 \(\xi\). As seen in the figure, for sentence-pair tasks, the optimal \(\xi\) is between 2 to 5, while easier single sentence tasks are not sensitive to this hyper-parameter. We also try using the prototype embeddings \(z_c\) for model inference, of which the results are similar. We suggest that when CP-Tuning is
Table 4: Analysis of the batch size. The results show that our approach does not need a large batch size.

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

| Task/Method | CP-Tuning | PET |
|-------------|-----------|-----|
| SST-2       | 92.91     | 91.28 |
| MR          | 88.38     | 86.28 |
| MRPC        | 71.80     | 65.73 |
| QQP         | 73.84     | 66.61 |

Table 6: 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.

3.5 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 5. 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 2). 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.

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 [34, 35]).

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 [34, 35]. In Table 6, 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.

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 [36], the N-pair loss [41], InfoNCE [45] and the supervised contrastive loss [15]. Due to its effectiveness, contrastive learning has been applied to various NLP tasks, e.g., sentence representation [7, 16], text summarization [49], aspect detection [37], machine translation [51], commonsense reasoning [18].

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 [30]. Among these PLMs, ELMo [29] 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 [19], Transformer-XL [3], XLNet [52], StructBERT [50], Big Bird [53] and many others. Apart from the encoder-based PLMs, the encoder-decoder architecture is used in T5 [31] 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. [21]. To name a few, PET [34, 35] 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 [31]. In addition, Jiang et al. [14] also mine high-performing prompts from the training corpus. Auto-Prompt [38] 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 [23] learns continuous prompt embeddings with differentiable parameters for GPT-based models. The update version, P-tuning v2 [22] extends P-tuning to different scales of PLMs and NLP tasks. Prefix-tuning [20] extends the usage of continuous prompts for text generation tasks. Min et al. [26] 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 [19] optimizes the verbalizer construction process by integrating the knowledge from knowledge bases. The paper [13] presents a concurrent work that employs contrastive learning as an auxiliary loss. Our work further applies contrastive learning to making few-shot learners 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 Srerpekt. 2014. Benchmarking Applied Semantic Inference: The PASCAL Recognising Textual Entailment Challenges. In Language, Culture, Computation. Computing - Theory and Technology. Vol. 8001. 446–449.

[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, Alex Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models Are Few-Shot Learners. In NeURPS.

[3] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. 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 WPT@JCNLP.

[6] Tianyu Gao, Adam Fisher, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot learners. In ACL/JCNLP. 3816–3830.

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

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

[9] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In Proceedings of the 22nd Annual International Conference on Machine Learning. ACM. 492–499.

[10] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jianzhi Li, and Maosong Sun. 2021. Know What Language Models Know. In CoRR. abs/2106.02785 (2021).

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

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

[13] Yiren Jian, Chongyang Gao, and Soroush Vosoughi. 2022. Contrastive Learning with Contrastive Prompt Tuning. In WSDM ’23, February 27-March 3, 2023, Singapore, Singapore.

[14] 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. 446–449.

[15] Riadh Zbiri, Mohamed Amor, Nadia Soltani, and Mohamed El Hafidi. 2021. Contrastive Self-Supervised Learning for Deep Language Understanding. In ICLR.

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

[17] Menglong Zhao, Zhongzheng Lu, Minchen Li, and Mingda Chen. 2021. Improving Few-shot Language Understanding with Cross-Modal Contrastive Learning. In ICLR.

[18] Zonghan Yang, Yong Cheng, Yang Liu, and Maosong Sun. 2019. Reducing Word Embedding Size in Large-Scale Pre-trained Language Models. CoRR abs/1907.02035 (2019).

[19] Yifan Sun, Changmiao Cheng, Yuhang Zhi, Chi Zhang, Liangzhang Xiao, and Yichen Wei. 2020. Circle Loss: A Unified Perspective of Pair Similarity Optimization. In CVPR. 6979–6986.

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

[21] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Elliciting Knowledge from Language Models with Automatically Generated Prompts. In EMNLP. 4222–4235.

[22] 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 Sentence Treebank. In EMNLP. 1641–1652.

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

[24] Minqing Hu and Bing Liu. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In ACL. 115–124.

[25] Colin Raffel, Noam Shazeer, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In NeurIPS. 4077–4087.

[26] 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 Sentence Treebank. In EMNLP. 1641–1652.

[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] Colin Raffel, Noam Shazeer, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In NeurIPS. 4077–4087.

[29] 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.

[30] Tanyu Gao, Adam Fish, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot learners. In ACL/JCNLP. 3816–3830.

[31] Tanyu Gao, Xingchong Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. CoRR abs/2104.08821 (2021).

[32] William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In WPT@JCNLP.

[33] William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In WPT@JCNLP.

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

[35] Timo Schick and Hinrich Schütze. 2021. It’s Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In NAACL. 2359–2352.

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

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

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

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

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

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