Adversarial Knowledge Stimulated Contrastive Prompting for Few-shot Language Learners

Kai Zheng Qingfeng Sun Yaming Yang†
Tengchao Lv Yeyong Pi Changlin Zhao Fei Xu Qi Zhang
Microsoft, Beijing, China
{zhengkai,qins,yayaming,tengchaolv,yepi,neilz,fexu,qizhang}@microsoft.com

Abstract
Prompt-based fine-tuning has boosted the performance of Pre-trained Language Models (PLMs) on few-shot Natural Language Understanding (NLU) tasks by employing task-specific prompts. Yet, PLMs are unfamiliar with prompt-style expressions during pre-training, which limits the few-shot learning performance on downstream tasks. It would be desirable if the models can stimulate prompting knowledge while adaptation to specific NLU tasks. We present the Adversarial Knowledge Stimulated Contrastive Prompting (AKSCP) framework, leading to better few-shot NLU tasks for language models by implicitly stimulate knowledge from pretrained language model. In AKSCP, a novel paradigm Cloze-driven prompt is proposed for joint prompt tuning across word cloze task and prompt-based learning, forcing PLMs to stimulate prompting knowledge. We further design an Adversarial Contrastive learning method to improve the generalization ability of PLM for different downstream tasks. Experiments over a variety of NLU tasks show that AKSCP consistently outperforms state-of-the-arts for prompt-based fine-tuning.

1 Introduction
In recent years, pretrained language models (PLMs) have improved performance on various Natural Language Understanding (NLU) tasks such as general language understanding evaluation (GLUE) (Wang et al., 2018; Radford et al., 2018; Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019; Raffel et al., 2020). However, during fine-tuning, PLMs would perform poorly with few training samples due to model over-fitting (Gao et al., 2020).

To alleviate the above dilemma for low-resource scenarios, natural language prompts have been applied to enable few-shot or zero-shot learning with PLMs (Brown et al., 2020b; Li and Liang, 2021; Liu et al., 2021b; Lester et al., 2021; Liu et al., 2021a). To make prompts more flexible and task-adaptive, prompt tuning freezes the PLM backbone and only adjusts the representations of prompts (Lester et al., 2021). This type of method is especially suitable for ultra-large PLMs that are difficult to tune. In addition, prompt-based fine-tuning has been proposed, transforming text classification tasks into cloze-style problems (Gao et al., 2020; Schick and Schütze, 2020). To specify, task-specific discrete templates with masked language tokens are added to input texts. The result tokens of the masked positions predicted by the Masked Language Modeling (MLM) head are used for class label prediction. Therefore, the pre-trained knowledge acquired by PLMs can be better utilized by “re-using” the MLM training objective. However, prompt construction is usually handcrafted or searched by gradient descent, which may lack coverage and bring considerable bias and high variances to the results (Hu et al., 2021a). A recent work (Hu et al., 2021a) attempts to tackle the above challenge using external training knowledge data. Yet, these external knowledge data could be expensive to obtain and not transferable. It would be better if PLMs can stimulate more internal knowledge while they are adapted to downstream tasks without any external knowledge data.

A major limitation of task-based prompts is that they are too coarse-grained and fail to capture the fine-grained information in the input data. Existing methods use the same prompt for all input data within a tuning task. However, the input data also contains context-specific information that can help the PLM retrieve more relevant knowledge, such as the particular entity being discussed. Such knowledge embedded in the input data should be fully exploited to unleash the potential of prompts. The key challenge is that there is a mismatch between prompt and pre-training, because the template used in the prompt may not
be present during pre-training (Gao et al., 2022; Su et al., 2022; Zheng et al., 2022). To address this issue, we propose a unified paradigm named Cloze Driven Prompt (CDP). CDP uses a word cloze task that is more compatible with PLM, that is, it only masks and reconstructs the original input (see Table 1) to activate the knowledge learned by the original PLMs. To enable the model to better understand the NLU task, we further propose a novel adversarial contrastive learning objective to encourage the PLM to discriminate between different classes. Specifically, we propose a supervised contrastive framework that clusters inputs from the same class under different augmented "views" and pushes away the ones from different classes. We create different "views" of an example by appending it with various language prompts and contextual demonstrations. Furthermore, we design a prompt-based adversarial training method to improve the generalization abilities of PLMs. As our training method does not actually generate adversarial samples, it can be applied to large-scale training sets efficiently.

We conduct experiments over 15 public NLU benchmarks. Evaluation results indicate that our model Adversarial Knowledge Stimulated Contrastive Prompting (AKSCP) not only outperforms the performance of the state-of-the-art models, but also exhibits a good generalization ability over extensive tasks. In addition, we find that with the decrease of training data, the performance of AKSCP with fine-tuned parameters consistently outperforms the standard prompt learning method which freezes LM parameters. We also analysis the sample efficiency and the improvement margin difference to further verify the correctness of our motivation of AKSCP.

In this paper, we make three main contributions: (1) introduce a knowledge stimulated method that leverages knowledge of pre-trained language models (PLMs) to enhance the performance of prompt tuning; (2) proposal of a unified cloze adversarial contrastive prompting learning framework that jointly optimizes the cloze prompts and the PLM parameters in an adversarial and contrastive manner; and (3) conduct extensive experiments on fifteen few-shot natural language understanding (NLU) datasets and demonstrate the effectiveness of our approach.

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Mask</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-training</td>
<td>15%</td>
<td>It just loved every minute</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>0</td>
<td>M just loved every minute M</td>
</tr>
<tr>
<td>Prompt tuning</td>
<td>1</td>
<td>It just loved every minute</td>
</tr>
<tr>
<td>CDP</td>
<td>sup-k</td>
<td>M every minute M</td>
</tr>
</tbody>
</table>

Table 1: Masked examples. M denotes [MASK] token. Different colors represent different mask strategies’ tokens replaced by M. Italic words represent prompt template. All models are large models trained with the efficient pre-training.

2 Related Work

Prompt tuning Many studies (Li and Liang, 2021; Liu et al., 2021b; Lester et al., 2021; Liu et al., 2021a) have focused on how to design prompts since good prompts can narrow the gap between pretrained language models and downstream tasks. Depending on the prompt types, existing researches can be divided into two main categories: manually designed ones (Li and Liang, 2021; Liu et al., 2021b; Lester et al., 2021; Liu et al., 2021a) and automatically created ones (discrete prompts (Gao et al., 2020; Schick and Schütze, 2020) or continuous prompts (Shin et al., 2020; Hambardzumyan et al., 2021)), where continuous prompts focus on utilizing learnable continuous embeddings as prompt templates rather than label words. However, these prompts construction still lack coverage and bring considerable bias and high variances to the results. Recently, Hu et al. (2021a) propose to utilize external knowledge data to solve this issue. However, these works can not stimulate knowledge without external data directly.

Contrastive learning Contrastive learning is a self-supervised learning technique that aims to learn representations that are semantically similar for samples from the same class (positive pair) and semantically dissimilar for samples from different classes (negative pairs). This technique achieves this by maximizing the lower bound of the mutual information between two augmented views of the samples (Bachman et al., 2019; Tian et al., 2020b,a). Various contrastive learning methods have been proposed (Wang et al., 2021; Logeswaran and Lee, 2018; Wang et al., 2020; Gao et al., 2021; Zhang et al., 2021). Among them, SupCon (Khosla et al., 2020) is a distinctive method that performs contrastive learning at the class level by clustering two augmented batches of samples in the feature space. This allows SupCon to generate more negative pairs, which enhances the efficiency of contrastive learning in practice.

Adversarial training Many approaches for im-
proving the model robustness against adversarial perturbations (Szegedy et al., 2013) have been advanced. Goodfellow et al. (2014) proposed a FGSM method based on linear perturbation of non-linear models. Later, Madry et al. (2017) presented PGD-based adversarial training through multiple projected gradient ascent steps to adversarially maximize the loss. In NLU, Belinkov and Bisk (2017); Iyyer et al. (2018) exploited structure invariant word manipulation and robust training on noisy texts for improved robustness. Adversarial training also plays a role in improving model’s generalization (Cheng et al., 2019). Dong et al. (2020) exploit FGM-based adversarial training in self-learning for improved NLU tasks. In our setting, we count on adversarial training in the word embedding space and show that PGD-based adversarial training remains effective when the adversarial perturbation is applied to noisy augmented examples.

3 Problem Formalization

In the low-resource NLU tasks, only a set of labeled training data \( \mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \). Following the few-shot setting of Gao et al. (2020), we assume to have access to a pre-trained language model \( \mathcal{M} \) and \( \mathcal{D} \), we aim to learn a model \( p_\theta(y|x) \) from \( \mathcal{D} \) without any external knowledge and related data. For all of the following experiments, there are only \( K \) examples per class in \( \mathcal{D} \).

4 Methodology

This section first formulates the knowledge stimulation method for low-resource NLU task. We then introduce the two important components in proposed Adversarial Knowledge Stimulation Contrastive Prompt (AKSCP), including i) Cloze Driven Prompt; ii) Adversarial Contrastive learning. Figure 1 shows the overall AKSCP.

4.1 Low-Resource Learning Framework

Figure 2 shows the graphical model of our approach. The model consists of four variables: dataset \( \mathcal{D} \), cloze-driven \( \bar{\mathcal{D}} \) construct by keyword extraction method \( \mathcal{P} \), label \( \mathbf{Y} \), latent knowledge \( Z^b_K \) stimulated by word cloze task, and latent knowledge \( Z^y_K \) stimulated by prompt tuning. The variable \( Z^b_K \) bridges \( \mathcal{D} \) and \( Z^y_K \) guided by \( \bar{\mathcal{D}} \) using self-supervised learning, while \( Z^b_K \) and \( Z^y_K \) stimulated each other simultaneously. Prompt tuning use prompt to stimulate task-related knowledge \( Z^b_K \), while word cloze task narrows the gap between pre-

Figure 1: An illustration of our proposed Adversarial Knowledge Stimulated Contrastive Prompting approach.

Figure 2: Abstract Logic of the proposed approach. Solid lines indicate the existence of stimulation links in both the probabilistic graph and the neural graph, while dotted lines indicate different levels of stimulation intensity, red lines mean backward optimization.
training and prompt tuning, and stimulates more enriched knowledge from PLMs. Moreover, to improve the generalization abilities of PLMs, adversarial attack $A$ is applied at the embedding level. In addition, $Z^u_K$ is an unsupervised knowledge that does not require labeled data and has a good extensibility. However, $Z^p_K$ is a supervised knowledge that requires labeled data. Finally, $Z^g_K$ injects its more enhanced knowledge into the $Z^p_K$ for downstream training. Therefore, the variables enable us to model the objectives of different knowledge levels with respect to keywords in a unified framework. Some advantages of joint learning include (1) the model and contrastive learning are more robust to the noise in $Z^g_K$, inferred in the training process due to the protection of each task; and (2) in terms of prediction, the model can automatically control the expression of knowledge, so it can easily adapt to different scenarios without too much extra effort. The overall objective of learning as

$$L_\theta = \mathbb{E}_{(X,D)}[\log_{\mathcal{M}_\theta}(D)]$$  \tag{1}

4.2 Cloze Driven Prompt

4.2.1 Prompt-based Learning

Fine-tuning is a common method to adapt PLM to specific downstream tasks (Devlin et al., 2019). However, for low-resource data augmentation, we want the stimulated synthetic knowledge $\mathcal{K}_{LM}$ to be different from $\mathcal{K}$, and provide new information for NLU model learning. Fine-tuning PLM may not be an optimal solution, as it may overfit to a small number of training examples. Inspired by the zero-shot instructions in GPT3 (Brown et al., 2020a), we adopt prompt learning, which keeps the whole PLM parameters frozen, and adds discrete natural language task instructions (e.g. "translate into English") before the task input. Freezing PLM parameters may help with generalization during training. However, finding a suitable discrete task instruction is not easy to optimize in an end-to-end manner, and requires additional human effort. Compared with the previous methods (Brown et al., 2020a; Gao et al., 2020) of generating prompts by manual or neural network methods, we design prompt mapping based on several heuristic rules: $G_p$ represents the mapping of NLU tasks. Let $X$ and $G_p = G_p(x)$ denote the input sentence and the instance prompt respectively. Then we get the input: $x_{input} = [CLS]X[C]G_p [SEP]$.

We use $[C]$ as a special token to separate the prompt from the input sentence. For example, the input of Figure 1 is "I love every minute of the movie, it was [MASK].", where the prompt $G_p$ is "it was [MASK]". If a sentence does not follow this format, we append multiple [MASK] tokens to the end of the sentence. The number of [MASK] tokens in the prompt is a predefined hyper-parameter $l_{mask}$. We use demonstrations of label words to construct our input as follows:

$$x_d = x_0, t_0([MASK]), x_1, t_1(word_t), \text{ where } t_0 \text{ and } word_t \text{ are the template and the label word for } s_i \text{ respectively, and } s_i \text{ is sampled from the training set. During training, we update the parameters using masked language modeling (MLM) loss:}$$

$$L = \text{MLM}(x_{input}, y)$$  \tag{2}

where $y$ denotes the label word that corresponds to $x_{input}$.

4.2.2 Word Cloze Task

Table 1 illustrates that conventional prompt-based learning approaches rely on a single mask token to infer the label of an entire sentence. However, this method faces a challenge because pre-trained language models (PLMs) are not exposed to prompt-style expressions during pre-training, resulting in a gap between the prompt and the PLM’s knowledge. To address this issue, we propose a word cloze task that bridges the gap between pre-training, prompting, and fine-tuning in natural language understanding (NLU).

The word cloze task has a significant impact on knowledge stimulation, especially in low-resource setting. Hu et al. (2021a) propose to further train the full PLM parameters using external knowledge bases (KBs) to enhance the knowledge capability. However, this strategy (i.e., full PLM training) incurs high data collection costs and substantial computational overhead. In contrast, we propose to directly train the parameters using the word cloze task without any external training data. Assuming that knowledge stimulation updates the parameters based on partial information (such as keywords) through the MLM model, we propose the Significant Keywords to Sentence cloze task. Given a piece of text, we use the unsupervised keyword extraction algorithm to extract keywords. Given these keywords, the Cloze Sequence is trained to reconstruct the original text blocks. When the Cloze Task is applied to knowledge stimulation, we only
need to fine-tune the Cloze Sequence under unsupervised learning. This training process is conducted jointly with the prompt-based learning process. We only use the few-shot training data. Formally, the Significant Keywords to word cloze task creates a corrupted version \( x_c \) for an input \( x_{\text{input}} \): \( x_c = [\text{CLS}]X'[\text{SEP}] \), where \( X' \) is the corrupted version of \( X \) using Significant Keywords masking. After constructing this corrupted version of the sequence, the MLM model attempts to predict the masked tokens to restore the original tokens. The word cloze task loss is then defined as:

\[
L_C = \text{MLM}(x_c, x_{\text{input}}) \tag{3}
\]

4.3 Adversarial Contrastive Learning

4.3.1 Multi-View Contrastive Learning

Previous works often limit the encoder inputs to demonstration or view in a random strategy, such as random demonstration (Gao et al., 2020) and random view (Jian et al., 2022a). The relatively random sampling could mislead the model with counter target or event result in grouping together in the latent space. To enrich the positive pair construction, we propose Multi-View to generate positive pairs from the input view (conditional on keywords in the input statement) and the output view (conditional on labels). Figure 1 illustrates examples of these two views. As shown in Algorithm 1 (line 4 to 5), after fine-tuning the Word Cloze task in PLMs, AKSCP first generates \( \mathcal{H}_I \) and \( \mathcal{H}_O \) from the input view and output view respectively. AKSCP then extracts labels from \( \mathcal{H}_I \) and [MASK] statement from \( \mathcal{H}_O \). We select sentences with the same [MASK] tokens and the same label as positive instances, and negative instances otherwise. In order to reduce the inconsistency caused by masking between training and evaluation, we keep the probability \( \delta \) mentioned by a phrase unchanged in a direct alignment. In this way, the resulting output text should maintain a higher level of distinguishability and diversity in latent space and stimulate more task/keyword agnostic novel knowledge. We use SupCon (Khosla et al., 2020) to compute the contrastive learning loss. To apply SupCon on multiple views of input text, we need to first obtain two views of text: \( s_1 = x_0, G_{p_0}, x_i, G_{p_i} \) and \( s_2 = x_0, G_{p_j}, x_j, G_{p_j} \). We generate candidate demonstrations for each input instance based on different \( G_p \). Let \( s_{2b-1}, s_{2b} \) be two augmented views of input batch \( s_0 \), and \( r_{2b} \) and \( r_{2b-1} \) are the features of \( s_{2b-1} \) and \( s_{2b} \), \( y_b \) denote the label for \( x_b \), then we can calculate the SupCon loss as follows:

\[
L_S = \text{SupCon}(r_{2b-1}, r_{2b}, y_b) \tag{4}
\]

4.3.2 Adversarial Training

After completing the two kinds of Multi-View data augmentation, we obtain synthesized data that is substantially less noisy, denoted as \( \tilde{\mathcal{H}}_{LM} = \mathcal{H}_I \cup \mathcal{H}_O \), as shown in Algorithm 1 (line 6). We then proceed to train the model \( f(\cdot; \theta) \) for the final NLU tasks. As a special training regimen, we adopt adversarial training, which aims to minimize the maximal loss caused by label-preserving adversarial perturbations (Szegedy et al., 2013; Goodfellow et al., 2014), thereby making the model more robust. Specifically, adversarial training is especially effective in a Natural Language Inference (NLI) framework when used to exploit augmented data, as it encourages the model to be more resilient to the variation among similar words and word orders in different source sentences and to better adapt to the new moderately noisy data. We confirm this hypothesis in our experimental results (see SNLI in Table 3). Adversarial training is based on the idea of finding optimal parameters \( \theta \) to make the model robust against any perturbation \( r \) within a norm ball on a continuous (sub-)word embedding space. Hence, the loss function becomes:

\[
\mathcal{L}_{\text{Adv}}(x_i, y_i) = \mathcal{L}(f(x_i + r_{\text{Adv}}(x_i, y_i); \theta), y_i) \tag{5}
\]

\[
r_{\text{Adv}}(x_i, y_i) \approx \epsilon \frac{\nabla_{x_i} \mathcal{L}(f(x_i; \hat{\theta}), y_i)}{\|\nabla_{x_i} \mathcal{L}(f(x_i; \hat{\theta}), y_i)\|_2} \tag{6}
\]

Madry et al. (2017) demonstrated that projected gradient descent (PGD) allows us to find a better perturbation \( r_{\text{Adv}}(x_i, y_i) \). In particular, for the norm ball constraint \( ||r|| \leq \epsilon \), given a point \( r_0 \), \( \prod ||r|| \leq \epsilon \) aims to find \( r \) that is closest to \( r_0 \) as follows:

\[
\prod_{||r|| \leq \epsilon} (r_0) = \arg \min ||r - r_0|| \tag{7}
\]

In order to explore more optimal points in the latent space, one needs to perform \( K \)-step PGD during the training process, which entails \( K \) forward-backward passes through the network. Under a linear approximation and an \( L_2 \) norm constraint, each iteration of PGD takes the following form:

\[
r_{t+1} = \prod_{||r|| \leq \epsilon} (r_t + \alpha \frac{\nabla_{r_t} \mathcal{L}(f(x_i + r_t; \hat{\theta}), y_i)}{||\nabla_{r_t} \mathcal{L}(f(x_i + r_t; \hat{\theta}), y_i)||_2}) \tag{8}
\]

Here, \( \alpha \) is the step size and \( t \) is the step index.
Algorithm 1 Optimization Algorithm

Require: \( s \) : number of training iterations
1: \( D \) : few-shot labeled dataset
2: \( M \) : model
3: \( N \leftarrow 1 \)
4: \( \mathcal{H}_I \leftarrow \) GEN\((D, I)\) \hspace{1cm} \triangleright \text{input view} \\
5: \( \mathcal{H}_O \leftarrow \) GEN\((H, O)\) \hspace{1cm} \triangleright \text{output view} \\
6: \( \mathcal{H}_{LM} \leftarrow \mathcal{H}_I \cup \mathcal{H}_O \)
7: \text{while } N \neq s \text{ do}
8: \( \mathcal{L}_{S} = \text{SupCon}(\mathcal{M}^N, \mathcal{H}_{LM}) \)
9: \( \mathcal{L}_{P} = \text{CE}(\mathcal{M}^N, \mathcal{H}_{LM}) \)
10: \( \mathcal{L}_C = \text{CE}(\mathcal{M}^N, \mathcal{H}_{LM}) \)
11: \# Build \( \mathcal{H}_{LM}^\text{Adv} \) with PGD
12: \( \mathcal{L} = \mathcal{L}_P + \gamma \mathcal{L}_C + \beta \mathcal{L}_S + \alpha \mathcal{L}_{Adv} \)
13: \( M^N \leftarrow \text{TRAIN}(M, \mathcal{H}_{LM}) \)
14: \( N \leftarrow N + 1 \)
15: \text{end while}
16: \text{return } M

4.4 Joint Learning

To enable the integration of CDP and Adversarial Contrastive learning, we propose a joint training method:

\[
\mathcal{L} = \mathcal{L}_P + \gamma \mathcal{L}_C + \beta \mathcal{L}_S + \alpha \mathcal{L}_{Adv} \tag{9}
\]

where \( \beta, \alpha \) and \( \gamma \) are loss balance weights, and \( \alpha, \gamma, \beta \in (0.0, 1.0) \). We note that \( \gamma > 0.0 \) is required to ensure that the parameters of the word cloze task can be optimized through back propagation. \( \gamma < 1.0 \) is necessary to prevent the cloze task loss from reducing the performance of prompt tuning (Zhang et al., 2019).

5 Experiments

This section is organized as follows. Section 5.1 introduces the experimental settings. Main experimental results were reported in Section 5.2. In Section 5.3, we perform ablation studies. And Section 5.4 compares AKSCP and standard prompting under different settings and analyzes the sample efficiency.

5.1 Experimental Setup

Following the few-shot setting in LM-BFF (Gao et al., 2020), we conduct experiments on 15 tasks. For each benchmark, we perform shot-16 experiments following (Gao et al., 2020). We repeat the experiments 5 times and report the average results according to the previous works (Gao et al., 2020; Jian et al., 2022b). The Baseline model is\( \text{Roberta-BASE} \) model, which only uses a few-shot training data \( D \) for training. We use the same hyper-parameter settings to train the same \( \text{Roberta-BASE} \) model. We compare our method with LM-BFF (Gao et al., 2020) (a method with demonstrations) and PET (Schick and Schütze, 2020) (a method without demonstrations). We use the state-of-the-art method LM-SupCon (Jian et al., 2022a) as a prompt tuning method for all tasks.

Implementation Details AKSCP is based on the RoBERTa-base (Liu et al., 2019). Our method uses a single prompt/template (primary prompt) for the prediction of each task, and a set of prompts (auxiliary prompts) for generating multi-view inputs for contrastive learning. We use the Adam optimizer with a learning rate 1e-5, warm-up rate of 0.1 and weight decay of 1e-3 in training process. The number of [MASK] tokens in word cloze task is \( l_{mask} = 2 \). The batch size is set to 16. We conduct the training on 8 Nvidia Tesla V100 32G GPU cards. The \( \gamma \) in Eq.9 is set to 0.3. Early stopping on validation is adopted as a regularization strategy. We determine all the hyperparameters by grid search.

5.2 Main Results

In this subsection, we introduce the specific results and provide possible insights of AKSCP. 

Main results on 15 tasks Table 2 summarizes the experiment results in shot-16. In all tasks, our

<table>
<thead>
<tr>
<th>Task</th>
<th>LM-BFF</th>
<th>PET</th>
<th>LM-SupCon</th>
<th>AKSCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2 (acc)</td>
<td>89.0±0.7</td>
<td>88.4±1.0</td>
<td>90.4±0.1</td>
<td>91.5±0.2</td>
</tr>
<tr>
<td>Subj (acc)</td>
<td>90.2±0.5</td>
<td>89.2±1.5</td>
<td>90.4±1.1</td>
<td>90.8±0.8</td>
</tr>
<tr>
<td>SST-5 (acc)</td>
<td>47.9±2.8</td>
<td>46.0±0.9</td>
<td>45.1±1.1</td>
<td>49.8±1.6</td>
</tr>
<tr>
<td>CoLa (Mat.)</td>
<td>6.1±5.3</td>
<td>3.5±3.4</td>
<td>3.2±5.8</td>
<td>10.7±4.3</td>
</tr>
<tr>
<td>TREC (acc)</td>
<td>82.3±3.1</td>
<td>77.8±9.1</td>
<td>83.3±1.5</td>
<td>86.6±0.6</td>
</tr>
<tr>
<td>MNLI (acc)</td>
<td>61.0±2.1</td>
<td>58.2±1.1</td>
<td>64.0±2.0</td>
<td>64.1±2.4</td>
</tr>
<tr>
<td>MNLI-mm (acc)</td>
<td>62.5±2.1</td>
<td>59.8±1.2</td>
<td>65.5±2.7</td>
<td>65.6±2.3</td>
</tr>
<tr>
<td>SNLI (acc)</td>
<td>66.9±2.4</td>
<td>63.1±2.6</td>
<td>69.9±2.4</td>
<td>72.6±1.9</td>
</tr>
<tr>
<td>QNLI (acc)</td>
<td>60.7±1.7</td>
<td>51.8±3.3</td>
<td>66.4±3.5</td>
<td>66.5±4.3</td>
</tr>
<tr>
<td>QQP (acc)</td>
<td>62.5±2.6</td>
<td>61.9±3.5</td>
<td>68.8±3.8</td>
<td>68.9±3.0</td>
</tr>
<tr>
<td>RTE (acc)</td>
<td>64.3±2.7</td>
<td>60.9±4.7</td>
<td>65.1±3.5</td>
<td>66.0±3.0</td>
</tr>
<tr>
<td>MRP (Fl)</td>
<td>75.5±5.2</td>
<td>70.6±6.0</td>
<td>78.2±3.1</td>
<td>78.3±2.9</td>
</tr>
<tr>
<td>MR (acc)</td>
<td>83.3±1.4</td>
<td>85.0±0.6</td>
<td>85.8±0.6</td>
<td>86.2±0.8</td>
</tr>
<tr>
<td>MPQ (acc)</td>
<td>83.6±1.8</td>
<td>81.3±2.6</td>
<td>84.6±1.5</td>
<td>85.3±1.2</td>
</tr>
<tr>
<td>CR (acc)</td>
<td>88.9±1.0</td>
<td>89.3±1.0</td>
<td>89.4±1.0</td>
<td>89.5±0.9</td>
</tr>
</tbody>
</table>

Table 2: Few-shot experiments of baseline methods and ours (mean ± std). LM-BFF is a prompt-based method with demonstrations of label words, PET is one without demonstrations, LM-SupCon is SOTA approach. The experimental results show the means and standard deviations from 5 different train-test splits. ✗ results taken from (Jian et al., 2022a).
method can consistently boost the performance of baseline prompt tuning method LM-SupCon. There has a maximum improvement of 3.3% in TREC. The reason is obvious since the selection of label words among the vocabulary becomes inaccurate when labeled data is limited. Prompt empowered AKSCP successfully avoids this problem and stimulate PLM’s inner ability distribution among neurons to support model training on downstream tasks. In terms of variance, we can see that AKSCP enjoys smaller variances than baseline methods in most cases, demonstrating that the better coverage of label words stabilizes the training. Table 2 also shows that our method works outperform well than prompt-based methods without demonstrations. PET, which is a method without demonstrations, works consistently worse than AKSCP. In some tasks, e.g., SST-2, SST-5, QNLI, QQP, RTE MRPC, MR, and CR, the contribution of AKSCP can be even larger than the sole use of the demonstrations for label words. Figure 3(a), (b) and (c) shows the performance in shot-{16~2048} settings. AKSCP is always superior to other models in all settings. Compared with TREC and SNLI, the improvement of SST-2 is smaller. This may be due to the relatively high performance of SST-2 baseline (90.6%).

LM-SupCon performs consistently worse than AKSCP (e.g., more than 2.7% score gap in the SNLI and TREC experiment). This is because LM-SupCon tuned full PLMs, which can easily memorize the limited labeled training data and overfitting. In contrast, the adversarial contrastive learning allows AKSCP to maintain high generalization ability and CDP provide additional stimulated signals to the NLU models. The results from AKSCP are all statistical significant, compared to the Baseline model (paired student’s t-test, p < 0.05).

5.3 Ablation Study

We conduct ablation study to evaluate the effects of word cloze task, multi-view contrastive learning and adversarial training on the SST, TREC and SNLI datasets under the 16-shot setting.

**Word Cloze Task** To verify the contributions of the proposed prompt module, we replace the cloze-driven prompt with standard prompt. Table 3 shows the result: Cloze-driven Prompt (83.5%) outperforms the standard prompt tuning (82.2%) by up to 1.3% on average. The results verify the correctness of our motivation and the effectiveness of the word knowledge stimulation. This is because the key words in the sentence play a hint role, which makes the model ignore the overall semantic representation in the context, thus leading to representation collapse and generalization issue. Masking the phrase mentions forces the model to learn representations from context which prevents overfitting and representation collapse (Gao et al., 2020).

**Multi-View Contrastive Learning** We then examine the effect of Multi-View Contrastive Learning in AKSCP. We generate positive and negative data pairs from the input view and output view, respectively. As shown in Table 3, the data pairs from these two single views improve the model performance compared to the baseline. However, their performance is still inferior to that of AKSCP. This suggests that data from different views provide complementary training signals for NLU tasks. Interestingly, models trained with the output view outperform those trained with the input view, which indicates that the output pair provides more useful positive and negative examples for the task, and can guide the model to better learn from the contrastive learning objective.

**Adversarial Training** Finally, we examine the effect of Adversarial Training in AKSCP. In Table 3, we show the NLU model performance without adversarial training. The training has an important effect on the NLU performance. Without adversarial training, the performance gap almost disappears. The result show that adversarial training has a positive effect on the NLU performance. In particular, in SNLI Benchmark, the NLU model performance improves 1.3% acc score with adversarial training.

### Table 3: The ablation Acc scores over SST, TREC and SNLI of AKSCP for few-shot learning setting. w/o. denotes that we only remove one component from AKSCP. †refers to standard prompt tuning. • results taken from (Jian et al., 2022a).

<table>
<thead>
<tr>
<th>Model</th>
<th>SST-2</th>
<th>TREC</th>
<th>SNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Few-shot Baseline♣</td>
<td>90.0</td>
<td>83.3</td>
<td>69.9</td>
</tr>
<tr>
<td>AKSCP</td>
<td>91.5</td>
<td>86.6</td>
<td>72.6</td>
</tr>
<tr>
<td>output</td>
<td>90.9</td>
<td>85.6</td>
<td>72.0</td>
</tr>
<tr>
<td>input</td>
<td>91.1</td>
<td>86.0</td>
<td>72.2</td>
</tr>
<tr>
<td>w/o. Multi-View Contrastive Learning</td>
<td>90.8</td>
<td>85.0</td>
<td>71.8</td>
</tr>
<tr>
<td>w/o. Adversarial Training</td>
<td>90.9</td>
<td>85.9</td>
<td>71.3</td>
</tr>
</tbody>
</table>

**Comparasion with standard prompting.** Figure 3(d) compare our AKSCP and baseline model (LM-SupCon) with fine-tuned parameters and stan-
standard prompting (frozen the parameters of the LM) (Brown et al., 2020b) on SST-2 in shot-{4–2048} settings. Basically, when there are enough training data (e.g., > 16 shot), fine-tuning prompt can further improve the model performance than fixed parameters. With the decrease of shot, the performance of our prompt is consistently outperform than the fixed method. However, the results of baseline are opposite. The performances of LM-SupCon drop a lot when shot<16. In particular, acc drops by 29.5% (from 89.2% to 59.7%) in shot-8, even underperform than fixed model (67.5%), and the smaller the training size is, the bigger the gap is between the model with fixed parameters and the model with fine-tuned parameters. This is because the parameters will be over-fitted when the training size is small. AKSCP without word cloze task have the similar result. The reasons are i) word cloze task is Self-Supervised Learning (SSL), which can trained along with supervision signals provided by itself; ii) the keywords of sentence may play a hint role, which makes model ignore the overall semantic representation in the context, thus leading to representation collapse and generalization issue (Li et al., 2022). Masking the phrase forces model to learn representations from context which prevents overfitting and representation collapse with limited data (Li et al., 2022; Gao et al., 2020). Other tasks have the same experiment result.

Sample Efficiency We discuss how the performance of AKSCP, LM-SupCon, PET and LM-BFF varies when the number of training samples increases. In Figure 3(a),(b) and (c), we show the trend of these methods on the SST-2, TREC and SNLI datasets. For 16-shot to 2048 samples, our model is consistently better than others. The gap enlarges as the shot becomes fewer. Specifically, on TREC benchmark, the model performance is improved from 86.6% to 91.0% acc score in average. Comparing the baseline methods, AKSCP generally wins over other methods by a large margin especially in a low-shot setting. The reason is obvious since the selection of label words among the vocabulary becomes inaccurate when labeled data is limited (Hu et al., 2021b). And cloze-driven can stimulate inner ability distribution among neurons of PLM to enrich the selection (Su et al., 2022). In terms of variance, we can see that AKSCP enjoys smaller variances than baseline methods in most cases, demonstrating that the adversarial training effectively improves the robustness of the model.

Improvement Margin Difference As shown in Table 2, the improvement margins in the classification tasks are generally larger than the ones in the similarity and paraphrase tasks. The reasons are two-folds: i) the similarity and paraphrase tasks are more fine-grained and knowledge-intensive task than the single sentence classification task; ii) the stimulated knowledge for the similarity and paraphrase tasks includes entity type and boundary, which is more difficult for PLMs to mining, in particular for low-resource settings, compared to the sentence classification task (Wang et al., 2022).

Joint Learning parameter We investigate the effect of Joint Learning in AKSCP. It can be observed that, in general, a lower weight loss balance weight leads to better performance in most cases. Specifically, in Eq.9, setting \( \gamma \) to 0.3 is always better than other values on the SST dataset. This is because the weight of the word cloze task should not be too large, so as to avoid interfering with the prompt tuning tasks.

6 Conclusion

In this paper, we propose the first prompt-based knowledge stimulation model AKSCP for low-resource NLU tasks. We conduct experiments on 15 tasks and demonstrate the effectiveness of our approach. For future work, we plan to expand our model to other NLP tasks such as QA and NLG.
Limitations

In this paper, we only evaluated our method on a limited number of NLU tasks and datasets. It is possible that our method may not generalize well to other tasks or domains that require different types of prompting knowledge or cloze-driven prompts. A promising direction for future work is to investigate how the prompt design and the learning objective influence the performance and robustness of PLMs on few-shot NLU tasks.

Acknowledgements

We thank for the anonymous reviewers for their insightful suggestions to improve this paper.

References


The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
   *In the section on Experiments.*

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
   *In the section on Experiments.*

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
   *Not applicable. Not used existing packages.*

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
   *No response.*

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
   *No response.*

☐ D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
   *No response.*

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
   *No response.*

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
   *No response.*