Knowledge-Based Systems 274 (2023) 110647

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# KPT++: Refined knowledgeable prompt tuning for few-shot text classification

Shiwen Ni [a], Hung-Yu Kao [b,*]

[a] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[b] Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan

ARTICLE INFO

*Article history:*
Received 7 September 2022
Received in revised form 12 May 2023
Accepted 13 May 2023
Available online 19 May 2023

*Keywords:*
Natural language processing
Prompt tuning
Few-shot learning
Text classification

ABSTRACT

Recently, the new paradigm "pre-train, prompt, and predict" has achieved remarkable few-shot learning achievements compared with the "pre-train, fine-tune" paradigm. Prompt-tuning inserts the prompt text into the input and converts the classification task into a masked language modeling task. One of the key steps is to build a projection between the labels and the label words, i.e., the verbalizer. Knowledgeable prompt-tuning (KPT), which integrates external knowledge into the verbalizer to improve and stabilize prompt-tuning. KPT uses word embeddings and various knowledge graphs to expand the label words space to hundreds of words per class. However, some unreasonable label words in the verbalizer may damage the accuracy. In this paper, a new method called KPT++ is proposed to improve the few-shot text classification. KPT++ is refined knowledgeable prompt-tuning, which can also be regarded as an upgraded version of KPT. Specifically, KPT++ uses two newly proposed *prompt grammar refinement* (PGR) and *probability distribution refinement* (PDR) to refine the knowledgeable verbalizer. Extensive experiments on few-shot text classification tasks demonstrate that our KPT++ outperforms state-of-the-art method KPT and other baseline methods. Furthermore, ablation experiments and case studies demonstrate the effectiveness of both PGR and PDR refining methods.



## 1. Introduction

Pre-trained language models (PLMs) [1–7] have performed exceptionally well on various tasks (e.g., text classification [8], reading comprehension [9], text generation [10], etc.) in the field of natural language processing (NLP). Researchers have conducted extensive studies [11,12] to dissect the rationale for the effectiveness of PLMs, and experimental results show that PLMs learn a wealth of prior knowledge during pre-training. A common approach to exploiting PLMs for various downstream tasks is fine-tuning, which concatenates a trainable task-specific classifier at the last layer of PLMs. However, fine-tuning discards the pre-trained masked language model (MLM) head and requires retraining the task-specific classifier, which does not fully exploit the prior knowledge in PLMs.

Recently, a method to fully exploit prior knowledge in PLMs, prompt-tuning, has received extensive attention from the NLP community due to its astonishing performance on few-shot learning [13,14]. A general approach to prompt tuning is to combine input sentences with natural language templates to transform downstream tasks into masked language model tasks. For example, classify the sentiment of a sentence $\boldsymbol{x}$: "The plot of this movie is interesting.". We first combine the sentence $x$ and the template into: "$\boldsymbol{x}$ It was [MASK].". Assuming the label words are good and bad, the verbalizer [15] maps them to label positive and negative. Which category the sentence $\boldsymbol{x}$ is classified into depends on which word PLM fills in [MASK] with a greater probability of "good" and "bad". Verbalizer builds a projected bridge between the label word space and the label space, which significantly impacts the classification performance [16].

The original and most widely used are manual verbalizers [14,15], in which the developer manually selects a single word to represent each category. Human manual selection is time-consuming and may not be effective enough, and later studies use the discrete search [17–19] to find suitable label words automatically. However, the label words obtained by automatic discrete search lack prior knowledge and have less coverage. Therefore, to improve the coverage and reduce the bias of manual verbalizers, Hu et al. propose knowledgeable prompt tuning (KPT) [20], which incorporates external knowledge into the verbalizer to facilitate prompt-tuning. The core idea of KPT is to use WordNet [21], Word embeddings, ConceptNet [22], etc., as external knowledge bases to expand the label words corresponding to each category. Although KPT has constructed a knowledgeable verbalizer that

* Corresponding author.
*E-mail addresses:* sw.ni@siat.ac.cn (S. Ni), hykao@mail.ncku.edu.tw (H.-Y. Kao).

https://doi.org/10.1016/j.knosys.2023.110647
0950-7051/

contains comprehensive label words, the collected label words can be noisy since the vocabulary of the knowledge base is not tailored for the PLM. Therefore, in order to solve the above shortcomings, we propose refined knowledgeable prompt-tuning (KPT++), which uses *prompt grammar refinement* and *probability distribution refinement* to refine and filter label words in the knowledgeable verbalizer.

The main contributions of this paper can be summarized as follows:

(1) We propose a novel method, Refined Knowledgeable Prompt-tuning (KPT++), which performs better than the original KPT, and the improvement is more obvious when there are more noisy label words in the knowledgeable verbalizer.

(2) We propose two methods, prompt grammar refinement and probability distribution refinement, specifically for refining label words in knowledgeable verbalizers. Furthermore, the effectiveness of these two refining methods is demonstrated by constructing ablation experiments.

(3) Experimental results on five text classification datasets demonstrate that the few-shot performance of KPT++ outperforms the state-of-the-art method KPT and other baseline methods.

This paper is organized as follows. In Section 1, we introduce the research background and motivation of this paper. In Section 2, we review related works to pre-trained language models, and prompt learning, after which we describe the proposed method KPT++ and its two refining methods *prompt grammar refinement* and *probability distribution refinement* in Section 3. In Section 4, we describe the datasets, experimental setting and present the experimental results. In Section 5, we provide an in-depth analysis of our proposed KPT++. Finally, we conclude and look ahead in Section 6.

## 2. Related work

### 2.1. Pre-trained language models

In recent years, the emergence of various pre-trained language models has promoted the development and progress in the field of natural language processing. Devlin et al. [3] proposed a new language representation model called BERT, which stands for bidirectional encoder representations from transformers. BERT achieved state-of-the-art performance on multiple natural language understanding benchmarks, starting the era of pre-trained language models. Unlike BERT, which is based on bidirectional transformers, Alec et al. [23] proposed a left-to-right unidirectional language model (GPT). Yinhan et al. proposed RoBERTa by modifying the training process of the original BERT. Specifically, the static masking strategy is replaced with a dynamic masking strategy and the pre-training task is removed. Lan et al. [5] adopted the method of parameter sharing to reduce the memory consumption of the original BERT model, and replaced the next sentence prediction (NSP) task of the original BERT model with a new sentence order prediction (SOP) task, which further improved the performance. Subsequently, Yang et al. [24] proposed a Transformer-XL-based autoregressive pre-training model, XLNet, which learns bidirectional context via maximizing the expected likelihood of all permutations of the factorization order and overcomes the limitations of BERT due to its autoregressive formulation. Kevin et al. [6] proposed a pre-training model based on a generator-discriminator architecture, ELECTRA, which uses a novel replaced token detection (RTD) pre-training task to improve the pre-training efficiency. Furthermore, pre-trained language models such as BART [25], T5 [7], UniLM [26] and GPT3 [27] have achieved excellent results on natural language generation tasks.

### 2.2. Prompt tuning

Prompt learning has exploded in the field of natural language processing since it was discovered that feeding GPT3 [13] some hand-crafted templates and prompts without updating parameters could lead to surprising learning performance with few or even zero shots. Then an amazing work PET [14] was proposed, which was the first to apply prompt learning to normal-sized language models such as BERT and ALBERT, and achieved satisfactory few-shot learning performance. Different from the previous manual methods, the automatic verbalizer was proposed by Schick et al. [17], which uses labeled data to select the most informative label words in the PLM's vocabulary. Timo et al. [28] explored methods to identify label words automatically. However, it does not bring significant improvement compared to the hand-picked results. LM-BFF [16] is a set of simple and effective methods for fine-tuning language models on few-shot, which automatically generates prompts and searches for the best prompts. Li et al. [29] proposed a lightweight alternative for fine-tuning natural language generation tasks, prefix-tuning, which keeps language model parameters frozen and optimizes a sequence of continuous task-specific vectors called prefixes. At about the same time, P-tuning [18] was proposed, which employs trainable continuous prompt embeddings and improves both GPT and BERT well. Similarly, Hambardzumyan et al. [19] proposed WARP with a soft verbalizer, which uses a continuous vector for each class and uses the dot product between the masked language model output and the class vector to produce the probability for each class. Ni and Kao [30] use ELECTRA for prompt learning. Specifically, they transform the downstream task into a replaced token detection (RTD) pre-training task. Ni and Kao [31] proposed a novel masked Siamese prompt tuning (MSP-tuning) to improve few-shot natural language understanding. Specifically, MSP-tuning randomly masks some of the prompt tokens in order to obtain a pair of masked Siamese prompt words for each sample. Recently, Hu et al.. [20] proposed knowledgeable prompt tuning (KPT), which incorporates external knowledge into the prompt verbalizer for text classification. Unlike previous studies, only KPT and our KPT++ use external knowledge to integrate into the prompt verbalizer, and the KPT++ proposed in this paper is an optimized and improved version of KPT. Compared with KPT, our KPT++ proposes two new refining methods, PGR and PDR, to optimize the knowledgeable verbalizer.

## 3. Proposed method

### 3.1. Preliminaries

In this subsection, we first introduce how standard prompt tuning works for pre-trained language models. For prompt tuning, the size of the pre-trained language model and the knowledge learned in the pre-training phase directly determine the performance of the model in the prompt tuning phase. This paper focuses on comparing various prompt tuning methods, thus the main experimental section afterward uses the RoBERTa-large model, which is most commonly used by other prompt tuning methods. Given a pre-trained masked language model (MLM) $\mathcal{M}$, the discrete input tokens are $x_{in} = x_0, x_1, \ldots, x_n$. Let $\mathcal{V}$ refers to the vocabulary of a language model $\mathcal{M}$ and $e$ serves as the embedding function for $\mathcal{M}$. For instance, we can formulate a binary movie review sentiment classification task using a prompt (e.g., "It was [MASK]".) with input $x_{in}$ (e.g., "This is the most exciting action movie this year".) as:

$$X_{template} = [CLS]e(x_{in})[SEP]e(It)\ e(was)\ e([MASK])e(.) \quad (1)$$

and let language model $\mathcal{M}$ predict whether it is more appropriate to fill in "great" (positive) or "terrible" (negative) for [MASK]

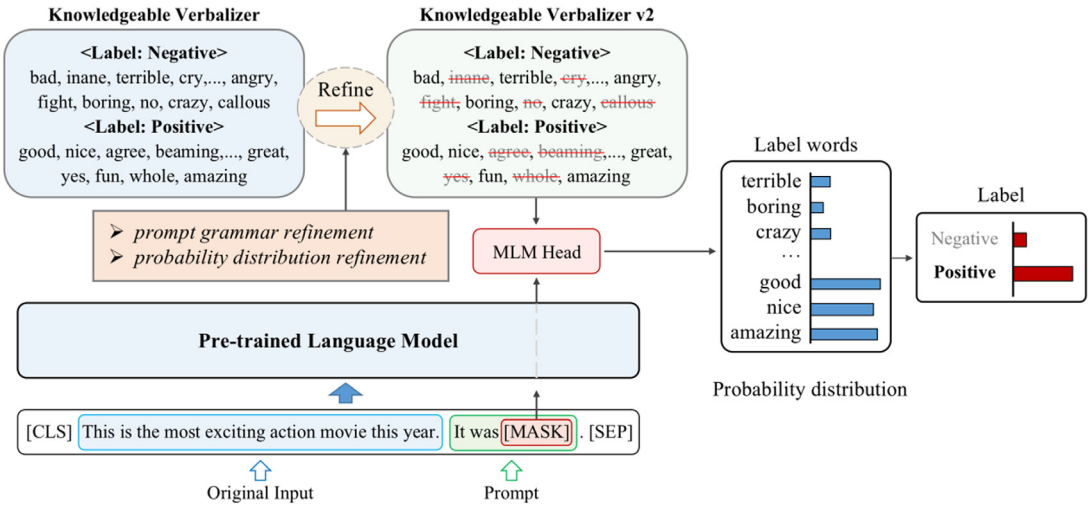


**Fig. 1.** The overall framework of KPT++. This is an example of sentiment binary classification for movie reviews. The main difference compared to KPT is that we propose two new verbalizer refining methods: *prompt grammar refinement* (PGR) and *probability distribution refinement* (PDR) to refine and upgrade the verbalizer.

token. Among them, the prompt "It was [MASK]". belongs to the vocabulary $\mathcal{V}$. Then, we formalize the training process. Firstly, let verbalizer $\mathcal{VB}$: $\mathcal{Y}$ be the mapping from the task label space in the vocabulary $\mathcal{V}$ in $\mathcal{M}$ to a label word. For example $\mathcal{VB}$: $\mathcal{Y}$ = *positive* $\rightarrow \mathcal{V}$ = {"great"}. We feed the masked language model (MLM) $\mathcal{M}$ with a [MASK] token. In this way, the original binary sentiment classification is converted into a masked language model task. We model the probability of predicting class $y \in \mathcal{Y}$ as:

$$\begin{aligned} P(y|x_{in}) &= p([MASK] = \mathcal{VB}(y)|X_{template}) \\ &= \frac{\exp(w_{\mathcal{VB}(y)} \cdot h_{[MASK]})}{\sum_{y' \in \mathcal{Y}} \exp(w_{\mathcal{VB}(y')} \cdot h_{[MASK]})} \end{aligned} \tag{2}$$

where h[MASK] is the hidden vector of [MASK] token. This method re-uses the pre-trained weights and does not introduce any new parameters.

### 3.2. KPT++: Refined knowledgeable prompt tuning

Unlike standard prompt-tuning, knowledgeable prompt tuning generates multiple label words related to each class y, e.g., $\mathcal{VB}$: $\mathcal{Y}$ = *positive* $\rightarrow \mathcal{V}$ = {"great", "good", "amazing", ...}. KPT++ uses WordNet [21], Word embeddings, ConceptNet [22], etc., as external knowledge bases to expand the label words corresponding to each category. The edges between each of these words represent relevance relationships and are labeled with relevance weights. We assume that the name of each class is the best-labeled word and use it as the central node and get the neighboring nodes with relevance weight greater than a threshold as the labeled word for that class. In this way, each class is given a set of label words. As shown in Fig. 1, a knowledgeable verbalizer containing hundreds of label words is obtained after label word expansion. Although we construct a knowledge-rich linguist with a comprehensive set of label words, the collected label words can be very noisy because the vocabularies of external knowledge bases are not tailored for pre-trained language models [20]. Therefore, we need to refine the verbalizer and filter out the noisy words. In this paper, we propose two refining methods *prompt grammar refinement* and *probability distribution refinement*. Next, we describe each of these two refinement methods.

#### 3.2.1. Prompt grammar refinement

For prompt tuning, label word and prompt are whole and should not be considered separately. In the original knowledgeable verbalizer, it only considered whether the meaning of the label word is reasonable, but after some label words are filled in [MASK], the sentence formed with the prompt word is unreasonable. For example, "can't" has a negative connotation from a certain point of view, but the sentence "It was can't". is clearly a grammatically incorrect sentence and a sentence with confusing meaning. As shown in Fig. 2, we combine each label word and prompt word into a sentence, and then input the trained grammar detection model to filter out those label words with grammatical errors. For the grammar detection model, We use a pre-trained BERT-GEC [32] model to perform grammatical error detection. This is the first refining process of KPT++.

#### 3.2.2. Probability distribution refinement

After PGR refining, the verbalizer is then refined by PDR. The probability distribution refinement does not abandon the original knowledgeable prompt tuning of the relevance refinement, but combines two refinement methods. We want the label words to have high relevance to this class and low relevance to other classes. The relevance refinement treats the vector of the probabilities of the label word on unlabeled support set $\boldsymbol{S}$ as the representation $Q^v$ of the label word. $Q^v$ can be expressed as:

$$Q^v = [p_1^v, p_2^v, \ldots, p_i^v] \tag{3}$$

Here $i$ is the number of data in support set $\boldsymbol{S}$. $Q^v$'s $i$th element is:

$$p_i^v = P_M([MASK] = v|X_{ip}), X_i \in \tilde{S} \tag{4}$$

During the calculation process, the name of each class is selected as the central word, such as "positive" for Positive. Thus, the vector representation $Q^{v0}$ of the these names as the class's representation $Q^y$. Thus, the relevance score R(v, y) between a label word v and a class $y$ is calculated as the cosine similarity between the two representation:

$$R(v, y) = \cos(Q^v, Q^y) = \cos(Q^v, Q^{v0}). \tag{5}$$

Furthermore, we hope the label words do not contribute positively to multiple classes, leading to confusion among classes. Empirically, the higher the correlation between the label word and this class, the better, and the lower the correlation with other classes, the better. As shown in Fig. 3, on the unlabeled support set, we want the distribution probabilities of labeled words to have a high standard deviation, rather than tending to the same probabilities. In order to filter confusing and useless label words, we designed a metric that favors label words that

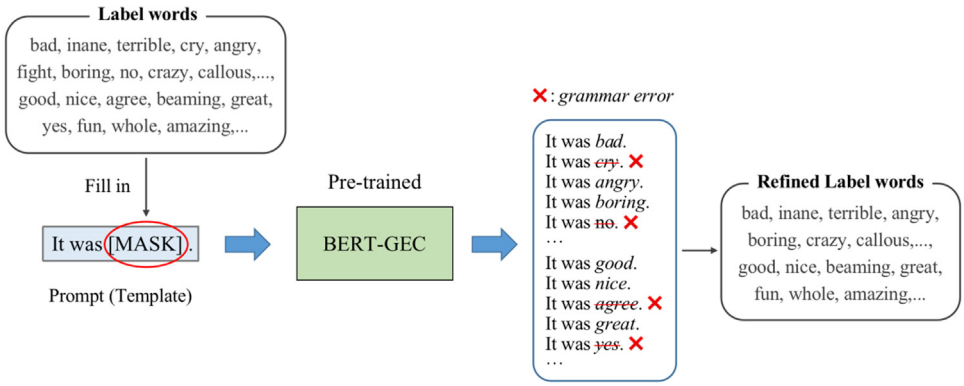


**Fig. 2.** The illustration of *prompt grammar refinement*.

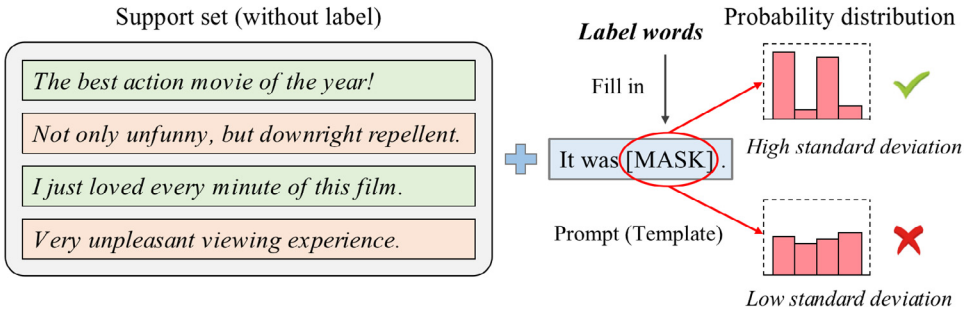


**Fig. 3.** The illustration of *probability distribution refinement*.

are only highly relevant to the category they belong to and lowly relevant to other categories with high standard deviation and differentiation of the distribution. The PDG refined standard value $RS^d(v)$ composed of correlation and the standard deviation is:

$$RS^d(v) = r(v, f(v))\left(\frac{|y| - 1}{\sum_{y \in Y,\, y \neq f(v)} r(v, y)^d}\right)^{1/d} + STD_v \tag{6}$$

where $f(v)$ is the corresponding class of v, the normalization coefficient $d$ is:

$$d = \frac{C}{|y| - 2 + \epsilon} + 1, C > 0, 0 < \epsilon \ll 1 \tag{7}$$

We set $C = 5$ and $\epsilon = 0.0001$ in the experiments. The standard deviation $STD_v$ of the distribution is expressed as:

$$STD_v = \sqrt{(p_1^v - p^{ave}) + (p_2^v - p^{ave}) + \cdots + (p_n^v - p^{ave})/(n-1)} \tag{8}$$

where $p^{ave}$ is the average of the label word probabilities.

### 3.2.3. Training and prediction

During training, the threshold of $RS^d(v)$ is a hyperparameter. We dynamically adjust the threshold of $RS^d(v)$ among {1, 1.2, 1.5} for the setting. We assign a learnable weight $w_v$ to each label word $v$ (may be already refined by the PGR and PDR methods). The weights form a vector $w \in \mathbb{R}^{|V|}$, which is initialized to be a zero vector. The weights are normalized within each $v_y$:

$$\alpha_v = \frac{\exp(w_v)}{\sum_{u \in y,\, u \neq v} \exp(w_v)} \tag{9}$$

We map the weighted average of the predicted probability of each refined label word to the class label y. The predicted $\hat{y}$ can be expressed as:

$$\hat{y} = \max_{y \in \mathcal{Y}} \frac{exp(s(y|x_p))}{\sum_{y'} exp(s(y'|x_p))} \tag{10}$$

**Table 1**
The statistics of datasets used in this paper.

| Dataset | Type | # Class | Test size |
|---|---|---|---|
| AG's News | Topic classification | 4 | 7600 |
| DBPedia | Topic classification | 14 | 70000 |
| Yahoo | Topic classification | 10 | 60000 |
| Amazon | Sentiment classification | 2 | 10000 |
| IMDB | Sentiment classification | 2 | 25000 |

where $s(y|x_p)$ is

$$s(y|x_p) = \sum_{y \in V_y} a_v log P_M([MASK] = v|x_p) \tag{11}$$

Finally, the predicted values are optimized using the cross-entropy function.

## 4. Experiments

In this section, we conduct extensive experiments on the five widely acknowledged text classification datasets. Experimental results show that our proposed KPT++ outperforms state-of-the-art method KPT and other baseline methods.

### 4.1. Datasets and templates

To better evaluate our proposed method KPT++, we use three topic classification datasets: AG's News [33], DBPedia [34], Yahoo [35], and two sentiment classification datasets: IMDB [36] and Amazon [37]. The statistics of the datasets are shown in Table 1. We follow KPT [20] to design the templates. In order to weaken the influence of templates on the experimental results, we design four templates for each dataset, as shown in Table 2.

**Table 2**
The templates of each dataset.

| Dataset | Template |
|---|---|
| AG's News | A [MASK] news : <**Input**> |
| | <**Input**> This topic is about [MASK]. |
| | [ Category : [MASK] ] <**Input**> |
| | [ Topic : [MASK] ] <**Input**> |
| DBPedia | <**Tittle**> <**paragraph**> <**Tittle***> is a [MASK] . |
| | <**Tittle**> <**paragraph**> In this sentence, <**Tittle***> is a [MASK] . |
| | <**Tittle**> <**paragraph**> The type of <**Tittle***> is [MASK]. |
| | <**Tittle**> <**paragraph**> The category of <**Tittle***> is [MASK]. |
| Yahoo | A [MASK] question : <**Input**> |
| | <**Input**> This topic is about [MASK]. |
| | [ Category : [MASK] ] <**Input**> |
| | [ Topic : [MASK] ] <**Input**> |
| Amazon | It was [MASK] . <**Input**> |
| | Just [MASK] ! <**Input**> |
| | <**Input**> All in all, it was [MASK]. |
| | <**Input**> In summary, the film was [MASK]. |
| IMDB | It was [MASK] . <**Input**> |
| | Just [MASK] ! <**Input**> |
| | <**Input**> All in all, it was [MASK]. |
| | <**Input**> In summary, it was [MASK]. |

### 4.2. Expermental setting

We chose the very widely used RoBERTa-large model for the pre-trained language model for the main experiments. Because the datasets are relatively balanced, we use Micro-F1 as the test metric in all experiments, which is described in the following equations:

$$\text{Recall}_{\text{mic}} = \frac{\text{TP}_1 + \text{TP}_1 + \cdots + \text{TP}_n}{\text{TP}_1 + \text{TP}_1 + \cdots + \text{TP}_n + \text{FN}_1 + \text{FN}_1 + \cdots + \text{FN}_n} \tag{12}$$

$$\text{Precision}_{\text{mic}} = \frac{\text{TP}_1 + \text{TP}_1 + \cdots + \text{TP}_n}{\text{TP}_1 + \text{TP}_1 + \cdots + \text{TP}_n + \text{FP}_1 + \text{FP}_1 + \cdots + \text{FP}_n} \tag{13}$$

$$\text{Micro} - \text{F1} = 2\frac{\text{Recall}_{\text{mic}} \times \text{Precision}_{\text{mic}}}{\text{Recall}_{\text{mic}} + \text{Precision}_{\text{mic}}}. \tag{14}$$

where $\text{TP}_i$ is the True Positive of class $i$; $\text{FP}_i$ is the False Positive of class $i$; $\text{FP}_i$ is the False Negative of class $i$.

We evaluate each prompt-based method using four templates and five random seeds, and the results we report are the average of 20 runs. Repeating the experiment multiple times can reduce the influence of randomness on the experimental results. For the probability distribution refinement based on the unlabeled support set $S$, the size $|S|$ is 200. For k-shot experiments, we sample k instances of each class from the original training set to form the few-shot training set, and resample k instances in each class to form the validation set. We take the value of k as 1, 5, 10 and 20 for few-shot experiments. For hyper-parameters, we run the model for 5 epochs and select the checkpoint with the best validation performance for testing. The maximum input length of the three datasets AG's News, DBPedia, and Yahoo is 128; the maximum input length of the two datasets Amazon, IMDB is 512. The learning rate of the optimizer is 3e-5. For the fairness of the experiments, we keep the same as KPT in terms of experimental settings.

### 4.3. Baselines

In order to thoroughly evaluate the performance of the proposed method, we compare KPT++ with a series of baseline methods:
(1) **Fine-tuning**: directly fine-tune the RoBERTa-large model based on a few samples and then use the hidden embedding of [CLS] to predict.
(2) **Prompt-tuning**: regular prompt-based few-shot learning, where each class corresponds to a single manually selected label word.
(3) **AUTO PT**: prompt-tuning with automatic verbalizer [17], which does not need to manually determine the label words in advance, but uses the label data to automatically select the most suitable label words in the PLM vocabulary.
(4) **SOFT PT**: prompt-tuning with soft verbalizer [19], which does not require specific label words, but uses a continuous vector (generated from the label name) for each class and uses the dot product between the masked language model output and the class vector to generate the probability of each class.
(5) **KPT**: knowledgeable prompt-tuning [20], prompt-tuning with knowledgeable verbalizer, which integrates external knowledge into the verbalizer, and each category correspond to multiple label words.

### 4.4. Experimental results

First of all, from the experimental results in Table 3, it can be seen that the prompt-based methods are much better than the Fine-tuning. The less training data there is, the greater the gap between prompt-tuning and fine-tuning. For example, on the DBPedia dataset, the 1-shot performance of KPT++ is 94.4%, while Fine-tuning is only 8.6%. Although AUTO PT does not need to manually select label words, its performance is still inferior to manual prompt-tuning, especially when there are low-few samples (e.g., 1-shot, 5-shot). SOFT PT slightly outperforms manual prompt-tuning. Compared with other baseline methods, KPT incorporating external knowledge has the best overall performance. Finally, our proposed KPT++ outperforms the state-of-the-art KPT on all datasets. The experimental results not only illustrate the validity of prior knowledge, but also demonstrate the importance of the refining process.

## 5. Analysis

This section provides an in-depth analysis of our proposed KPT++.

### 5.1. Ablation study

This subsection analyzes the impact of our two proposed refining methods (PGR and PDR) on the KPT++ method. For the ablation study, w/o PGR, w/o PDR and w/o BOTH is the variant of KPT++ that does not conduct PGR, PDR and both PGR and PDR, respectively. The training data for the experiments are 10 samples per class. In the experiments of this subsection, we perform prompt learning based on pre-trained RoBERTa-base and RoBERTa-large, respectively. The RoBERTa-base has 12 Transformer layers with 768 dimensions, 12 self-attentive heads per layer, and 125M parameters; The RoBERTa-large has 24 Transformer layers with 1024 dimensions, 12 self-attentive heads per layer, and 355M parameters. The experimental results are shown in Table 4, the performance of both w/o PGR and w/o PDR is lower than the original KPT++. Further, the worst overall performer is

**Table 3**
Results of 1/5/10/20-shot text classification based on RoBERTa-large. We report their mean ± standard deviation of 20 runs (four templates and five random seeds).

| Shot | Method | AG's News | DBPedia | Yahoo | Amazon | IMDB |
|---|---|---|---|---|---|---|
| 1 | Fine-tuning | 19.8 ± 10.4 | 8.6 ± 4.5 | 11.1 ± 4.0 | 49.9 ± 0.2 | 50.0 ± 0.0 |
| | Prompt-tuning | 80.0 ± 6.0 | 92.2 ± 2.5 | 54.2 ± 3.1 | 91.9 ± 2.7 | 91.2 ± 3.7 |
| | AUTO PF | 52.8 ± 9.8 | 63.0 ± 8.9 | 23.3 ± 4.5 | 66.6 ± 12.5 | 75.5 ± 15.5 |
| | SOFT PF | 80.0 ± 5.6 | 92.3 ± 2.3 | 54.3 ± 2.7 | 90.9 ± 5.8 | 89.4 ± 8.9 |
| | KPT | 83.7 ± 3.5 | 93.7 ± 1.8 | 63.2 ± 2.5 | 93.2 ± 1.3 | 92.2 ± 3.0 |
| | KPT++ | **84.5 ± 3.4** | **94.4 ± 1.5** | **64.1 ± 2.4** | **93.7 ± 1.4** | **92.8 ± 1.9** |
| 5 | Fine-tuning | 37.9 ± 10.0 | 95.8 ± 1.3 | 25.3 ± 14.2 | 52.1 ± 1.3 | 51.4 ± 1.4 |
| | Prompt-tuning | 82.7 ± 2.7 | 97.0 ± 0.6 | 62.4 ± 1.7 | 92.2 ± 3.3 | 91.9 ± 3.1 |
| | AUTO PF | 72.2 ± 10.1 | 88.8 ± 3.9 | 49.6 ± 4.3 | 87.5 ± 7.4 | 86.8 ± 10.1 |
| | SOFT PF | 82.8 ± 2.7 | 97.0 ± 0.6 | 61.8 ± 1.8 | 93.2 ± 1.6 | 91.6 ± 3.4 |
| | KPT | 85.0 ± 1.2 | 97.1 ± 0.4 | 67.2 ± 0.8 | 93.4 ± 1.9 | 92.7 ± 1.5 |
| | KPT++ | **86.1 ± 1.7** | **97.9 ± 1.1** | **68.3 ± 1.3** | **94.5 ± 1.7** | **93.5 ± 1.4** |
| 10 | Fine-tuning | 75.9 ± 8.4 | 93.8 ± 2.2 | 43.8 ± 17.9 | 83.0 ± 7.0 | 76.2 ± 8.7 |
| | Prompt-tuning | 84.9 ± 2.4 | 97.6 ± 0.4 | 64.3 ± 2.2 | 93.9 ± 1.3 | 93.0 ± 1.7 |
| | AUTO PF | 81.4 ± 3.8 | 91.5 ± 3.4 | 58.7 ± 3.1 | 93.7 ± 1.2 | 91.1 ± 5.1 |
| | SOFT PF | 85.0 ± 2.8 | 97.6 ± 0.4 | 64.5 ± 2.2 | 93.9 ± 1.7 | 91.8 ± 2.6 |
| | KPT | 86.3 ± 1.6 | 98.0 ± 0.2 | 68.0 ± 0.6 | 93.8 ± 1.2 | 92.9 ± 1.8 |
| | KPT++ | **87.6 ± 1.6** | **98.8 ± 1.2** | **68.7 ± 1.2** | **94.4 ± 1.5** | **93.5 ± 1.7** |
| 20 | Fine-tuning | 85.4 ± 1.8 | 97.9 ± 0.2 | 54.2 ± 18.1 | 71.4 ± 4.3 | 78.5 ± 10.1 |
| | Prompt-tuning | 86.5 ± 1.6 | 97.9 ± 0.3 | 67.2 ± 1.1 | 93.5 ± 1.0 | 93.0 ± 1.1 |
| | AUTO PF | 85.7 ± 1.4 | 92.2 ± 2.7 | 65.0 ± 1.8 | 93.9 ± 1.1 | 92.8 ± 2.0 |
| | SOFT PF | 86.4 ± 1.7 | 98.0 ± 0.3 | 67.4 ± 0.7 | 93.8 ± 1.6 | 93.5 ± 0.9 |
| | KPT | 87.2 ± 0.8 | 98.1 ± 0.3 | 68.9 ± 0.8 | 93.7 ± 1.6 | 93.1 ± 1.1 |
| | KPT++ | **88.4 ± 1.3** | **98.7 ± 0.4** | **69.6 ± 1.1** | **94.4 ± 1.7** | **93.8 ± 1.2** |

**Table 4**
Ablation study of KPT++.

| Method | | AG's News | DBPedia | Yahoo | Amazon | IMDB |
|---|---|---|---|---|---|---|
| RoBERTa-base | KPT++ | 85.4 ± 1.2 | 97.1 ± 1.1 | 65.4 ± 1.3 | 93.7 ± 1.5 | 91.5 ± 1.6 |
| | w/o PGR | 85.2 ± 1.3 | 96.5 ± 1.4 | 65.1 ± 1.5 | 93.4 ± 1.6 | 91.3 ± 1.6 |
| | w/o PDR | 84.9 ± 1.3 | 96.6 ± 1.2 | 65.3 ± 1.1 | 93.2 ± 1.3 | **91.1 ± 1.6** |
| | w/o BOTH | **84.2 ± 1.3** | **96.2 ± 1.7** | **64.8 ± 1.4** | **93.1 ± 1.5** | 91.2 ± 1.6 |
| RoBERTa-large | KPT++ | 87.6 ±1.6 | 98.8 ±1.2 | 68.7 ± 1.2 | 94.4 ±1.5 | 93.5 ± 1.7 |
| | w/o PGR | 86.4 ± 1.4 | 98.0 ± 0.9 | 67.9 ± 1.1 | **93.9 ± 1.2** | 92.9 ± 1.3 |
| | w/o PDR | 86.3 ± 1.5 | **97.8 ± 0.7** | 68.1 ± 1.3 | 94.1 ± 1.3 | 92.8 ± 1.1 |
| | w/o BOTH | **85.6 ± 1.4** | 97.9 ± 0.2 | **67.5 ± 1.1** | 94.0 ± 1.0 | **92.7 ± 2.1** |

w/o BOTH. The experimental results show that both PGR and PDR have positive effects on the model. The combination of PGR and PDR works best. Of course, the improvement of the refining method depends on the quality of the original knowledgeable verbalizer. For example, the original knowledgeable verbalizer has many noise words, so it is necessary to use the refining method, and the model will be significantly improved.

### 5.2. Performance influence between different pre-trained language models

In this subsection, we investigate the influence of different pre-trained language models on performance. We experimentally evaluate the results of KPT++ on BERT-base, BERT-large, RoBERTa-base, and RoBERT-large, respectively. The training data for the experiments are 10 samples per class. The experimental results are shown in Table 5. The four models, BERT-base, BERT-large, RoBERTa-base, and RoBERT-large, averaged 85.5%, 87.3%, 86.6%, and 88.6% on the five datasets, respectively. We find that RoBERTa-large outperforms BERT-large by 1.3%; RoBERTa-base outperforms BERT-base by 1.1%. The RoBERTa model learns more in the pre-training phase than BERT. Therefore using RoBERTa for prompt tuning is more effective. In addition, we find that BERT-large is 1.8% higher than BERT-base and RoBERTa-large is also 1.3% higher than RoBERTa-base. This phenomenon illustrates that for a prompt tuning method like KPT++, the larger the size of the pre-trained language model, the better the model works on downstream tasks.

### 5.3. Case studies

The RoBERTa-large model trained by KPT in Section 4 is selected for the case studies in this subsection. Table 6 shows three specific cases in which the KPT prediction is wrong but the KPT++ prediction is correct. "Wrong case" is the original input sample, "Label" represents the truth label, and "Prediction" represents the prediction result of KPT. The first sample is from AG's News dataset, and the task is to classify news topics, which is originally labeled as "Business", but KPT incorrectly predicts it as "Technology". The reason for this error is probably the presence of many unreasonable or confusing label words in the KPT, such as {Airplane, makers, wheelbarrow, economy, spoon, companies, pollution} in the label Technology. The phrase "train and plane maker's" in the sentence would lead the model to assume that this is technology news. Because PDR and PGR filtered out these unreasonable or confusing label words, KPT++ predicted correctly.

The second example is clearly technology news, but KPT predicts it as "Sports" news. Because KPT has some unreasonable and confusing label words, such as the "Sports" category with {home, English, humor, witticism, women, national, european}. We can see that the word "home" appears several times in the input, and the word "home" is actually a label word of the "Sport" category in the KPT, which will directly affect the model's judgment. Also, the PDR and PGR filter out these unreasonable or confusing label words, therefore KPT++ predicts correctly. The third case is from the sentiment classification dataset Amazon, which has a large number of negative words in this sample, such as depressing, dark side, puking, depravity and cruelty. These words basically match

**Table 5**
Results of KPT++ on different pre-trained language models.

| PLM | | AG's News | DBPedia | Yahoo | Amazon | IMDB | Avg |
|---|---|---|---|---|---|---|---|
| BERT | base | 84.2 ± 1.4 | 96.1 ± 1.3 | 64.3 ± 1.2 | 92.5 ± 1.4 | 90.3 ± 1.4 | 85.5 |
| | large | 86.5 ± 1.3 | 97.7 ± 1.4 | 66.8 ± 1.1 | 93.6 ± 1.6 | 91.8 ± 1.3 | 87.3 |
| RoBERTa | base | 85.4 ± 1.2 | 97.1 ± 1.1 | 65.4 ± 1.3 | 93.7 ± 1.5 | 91.5 ± 1.6 | 86.6 |
| | large | 87.6 ± 1.6 | 98.8 ± 1.2 | 68.7 ± 1.2 | 94.4 ± 1.5 | 93.5 ± 1.7 | 88.6 |

**Table 6**
Cases where KPT predicted wrongly but KPT++ predicted correctly.

| Wrong case | Label | Prediction |
|---|---|---|
| CEO quits at world's top maker of trains Paul Tellier stepped down as president and chief executive of Bombardier Inc. Yesterday, surprising investors and sending the train and plane maker's shares down as much as 26 percent to a 10-year low. (**AG's News**) | Business | Technology |
| The Kid Stays in His Home, Dressed in PJ's, With a Live Mike Robert Evans, the fabled Hollywood producer and man about town, will be enjoying his latest gig, as a satellite radio talk-show host, from the comfort of his home. (**AG's News**) | Technology | Sports |
| You sure you want to read this?. This book is not for a rainy day. Rainy day's are depressing and exploring the dark side of the human soul is no less. If you are into dreams leave this book on the shelf. If you want to feel like puking from the pit of stomach at the depts of human depravity this is the book for you. One sentence to describe this book: A daring expose of the cruelty of children and a slap on the wrist of anybody who thinks kids don't need discipline. (**Amazon**) | Positive | Negative |

**Table 7**
Cases where KPT++ predicted wrongly but KPT predicted correctly.

| Wrong case | Label | Prediction |
|---|---|---|
| If the ending hadn't been so fantastically unexpected, I don't think I could rate this movie so well. This movie has a lot of uncomfortable, distressing, "marriage falling apart" character interaction. ... I guessed every typical plot twist except the one that occurred. The ending definitely makes this movie worth watching. The intrigue and the drama, not quite as much. (**IMDB**) | Positive | Negative |
| This film is perfect for over the top cheesy zombie lovers. its a film you can laugh at from the acting to the terrible zombie action. ... if it was any better i don't think it would of made any difference but it wouldn't be interesting to see a remake with all the same cast as i believe they have possibly improved over the last 7 years. (**IMDB**) | Negative | Positive |

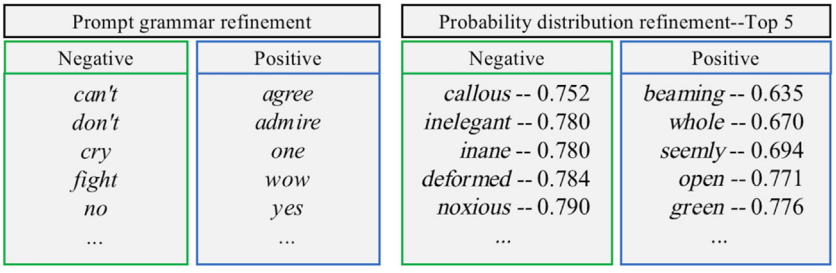


**Fig. 4.** Left: The label words filtered out by PGR. Right: The top 5 label words with the smallest standard deviation of the probability distribution. The values represent $STD_v$ in Eq. (8).

with the "Negative" category label words in KPT, therefore the model directly predicts "Negative". However, the true sentiment of this review is positive. We can see in these cases examples of confusing labels of KPT which caused misclassifications and were successfully removed by KPT++.

In addition, there are some samples where KPT++ is incorrectly classified but KPT is correctly classified, as shown in Table 7. The first example should be negative sentiment but KPT++ predicts positive sentiment, and the second example should be positive sentiment but KPT++ predicts negative sentiment. This indicates that filtering by PDR and PGR also have side effects on certain samples. Some of the deleted label words may be valuable for some specific samples. However, the overall effect of PDR and PGR is positive. In the IMDB test set, 227 samples were correctly predicted by KPT++ and incorrectly predicted by KPT, while only 25 samples were correctly predicted by KPT and incorrectly predicted by KPT++.

### 5.4. Label words filtered out by PGR and PDR

To more intuitively understand the role of PGR and PDR, we conduct case studies. As shown in Fig. 4 (Left), we show the labeled words for some sentiment classification tasks filtered by PGR. For example, "don't" has a negative connotation from a certain point of view, but the sentence "It was don't". is clearly a grammatically incorrect sentence and a sentence with confusing meaning. Intuitively, such words are not suitable as label words.

As shown in Fig. 4 (Right), we show the top 5 label words with the smallest standard deviation of the probability distribution. For example, the label words "callous" and "beaming" with the smallest standard deviation are very rare words. Too many such rare words can also hurt the performance of the model.

## 6. Conclusion

In this paper, we propose a powerful method, KPT++, based on knowledgeable prompt tuning for few-shot text classification. The core idea of KPT++ is that we propose prompt grammar refinement and probability distribution refinement, two methods dedicated to refining knowledgeable verbalizers. Specifically, the prompt grammar refinement refines the knowledgeable verbalizer by the grammar of prompt sentences with label words; the probability distribution refinement refines the knowledgeable verbalizer based on the predicted probability distribution of label words over the support set. Extensive experiments on five few-shot text classification datasets demonstrate that our proposed KPT++ outperforms state-of-the-art method KPT and other baseline methods. The experimental results show that our KPT++ performs better than KPT when there are more noise label words. In the future, we will study the application of KPT++ to more complex natural language processing tasks. Moreover, combining KPT++ with continuous prompts is an interesting research direction.

## CRediT authorship contribution statement

**Shiwen Ni:** Data curation, Funding acquisition, Investigation, Methodology, Software, Writing – original draft. **Hung-Yu Kao:** Supervision, Writing – review & editing.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Hung-Yu Kao reports financial support was provided by Ministry of Science and Technology, Taiwan. Hung-Yu Kao reports a relationship with Ministry of Science and Technology that includes: funding grants.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was funded in part by Qualcomm, United States through a Taiwan University Research Collaboration Project NAT-487842 and in part by the Ministry of Science and Technology, Taiwan, under grant MOST 111-2221-E-006-001.

## References

[1] Matthew E. Peters, et al., Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018.
[2] Alec Radford, et al., Improving language understanding by generative pre-training, 2018.
[3] Kenton, Jacob Devlin, Ming-Wei Chang, et al., BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL-HLT, 2019.
[4] Yinhan Liu, et al., Roberta: A robustly optimized bert pretraining approach, 2019, arXiv preprint arXiv:1907.11692.
[5] Zhenzhong Lan, et al., Albert: A lite bert for self-supervised learning of language representations, 2019, arXiv preprint arXiv:1909.11942.
[6] Kevin Clark, et al., ELECTRA: pre-training text encoders as discriminators rather than generators, in: International Conference on Learning Representations, 2019.
[7] Colin Raffel, et al., Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (140) (2020) 1–67.
[8] Kamran Kowsari, et al., Text classification algorithms: A survey, Information 10 (4) (2019) 150.
[9] Razieh Baradaran, Razieh Ghiasi, Hossein Amirkhani, A survey on machine reading comprehension systems, Nat. Lang. Eng. (2020) 1–50.
[10] Junyi Li, et al., Pretrained language models for text generation: A survey, 2021, arXiv preprint arXiv:2105.10311.
[11] Fabio Petroni, et al., Language models as knowledge bases? in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, 2019, pp. 2463–2473.
[12] Joe Davison, Joshua Feldman, Alexander M. Rush, Commonsense knowledge mining from pretrained models, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, 2019, pp. 1173–1178.
[13] Tom Brown, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[14] Timo Schick, Hinrich Schütze, It's not just size that matters: small language models are also few-shot learners, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 2339–2352.
[15] Timo Schick, Hinrich Schütze, Exploiting cloze-questions for few-shot text classification and natural language inference, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, 2021, pp. 255–269.
[16] Tianyu Gao, Adam Fisch, Danqi Chen, Making pre-trained language models better few-shot learners, in: Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021. Association for Computational Linguistics, ACL, 2021, pp. 3816–3830.
[17] Timo Schick, Helmut Schmid, Hinrich Schütze, Automatically identifying words that can serve as labels for few-shot text classification, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 5569–5578.
[18] Xiao Liu, et al., GPT understands, too, 2021, arXiv preprint arXiv:2103.10385.
[19] Karen Hambardzumyan, Hrant Khachatrian, Jonathan May, WARP: word-level adversarial ReProgramming, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 4921–4933.
[20] Shengding Hu, et al., Knowledgeable prompt-tuning: incorporating knowledge into prompt verbalizer for text classification, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 2225–2240.
[21] Ted Pedersen, et al., WordNet:: similarity-measuring the relatedness of concepts, in: AAAI, 2004, pp. 25–29.
[22] Robyn Speer, Joshua Chin, Catherine Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[23] Alec Radford, et al., Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.
[24] Zhilin Yang, et al., Xlnet: Generalized autoregressive pretraining for language understanding, Adv. Neural Inf. Process. Syst. 32 (2019).
[25] Mike Lewis, et al., BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871–7880.
[26] Li Dong, et al., Unified language model pre-training for natural language understanding and generation, Adv. Neural Inf. Process. Syst. 32 (2019).
[27] Luciano Floridi, Massimo Chiriatti, GPT-3: Its nature, scope, limits, and consequences, Minds Mach. 30 (4) (2020) 681–694.
[28] Timo Schick, Helmut Schmid, Hinrich Schütze, Automatically identifying words that can serve as labels for few-shot text classification, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 5569–5578.
[29] Xiang Lisa Li, Percy Liang, Prefix-tuning: optimizing continuous prompts for generation, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 4582–4597.
[30] Shiwen Ni, Hung-Yu Kao, ELECTRA is a zero-shot learner, too, 2022, arXiv preprint arXiv:2207.08141.
[31] Shiwen Ni, Hung-Yu Kao, Masked siamese prompt tuning for few-shot natural language understanding, IEEE Trans. Artif. Intell. (2023) http://dx.doi.org/10.1109/TAI.2023.3275132.

[32] Masahiro Kaneko, et al., Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction, 2020, EMNLP.
[33] Xiang Zhang, Junbo Zhao, Yann Lecun, Character-level convolutional networks for text classification, Adv. Neural Inf. Process. Syst. 28 (2015).
[34] Jens Lehmann, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semantic Web 6 (2) (2015) 167–195.
[35] Xiang Zhang, Junbo Zhao, Yann Lecun, Character-level convolutional networks for text classification, Adv. Neural Inf. Process. Syst. 28 (2015).
[36] Andrew Maas, et al., Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp. 142–150.
[37] Julian Mcauley, Jure Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: Proceedings of the 7th ACM Conference on Recommender Systems, 2013, pp. 165–172.