Pre-train, Prompt, and Predict: Improving BERTs' Classification Performance on Writing Arguments[*]

Hongzun Liu[†]
2022210866
Department of Computer Science and Technology
Tsinghua University
liuhz22@mails.tsinghua.edu.cn

Xuetong Wang[†]
2022311967
School of Economics and Management
Tsinghua University
wangxt22@mails.tsinghua.edu.cn

Jie Wu[†]
2022311968
School of Economics and Management
Tsinghua University
wj22@mails.tsinghua.edu.cn

Abstract

Deep learning methods have been widely used for natural language understanding and classification. Recently, impressive results have improved the automation of language education with the application of large pre-trained models. However, long-text, few-sample, and inconsistencies between the distribution of pre-trained data and downstream data, limited the performance of pre-trained classification models. In this paper, we proposed a BERT-based prompting method for text classification, and evaluated our method on an argumentative essays dataset on Kaggle, on which the model needs to rate arguments in terms of "*Ineffective*, *Adequate* and *Effective*". Compared with traditional head-based fine-tuning methods, the experiment result shows that the prompting method can significantly improve the few-shot classification performance, achieving improvements on NLL$\downarrow$ by -0.034%(0.735% vs. 0.701%), on accuracy$\uparrow$ by +3.3%(66.2% vs. 69.5%). The source code was released at https://github.com/liu-hz18/PromptCLS.

1 Introduction

Language model pre-training has been proven to be effective for improving many natural language processing tasks[Devlin et al., 2018, He et al., 2020, Radford et al., 2019]. Recently, deep learning methods have played an important role in education, especially in students' writing courses[Peters, 2018]. In particular, argumentative writing fosters critical thinking and civic engagement skills. However, resource constraints disproportionately impact some students, thus they are more likely to write at the "below basic" level compared to their peers. An automated feedback tool is one way to make it easier for teachers to grade writing tasks assigned to their students which will also improve their writing skills. Given an argumentative writing snippet from an essay, the goal of our work is to classify its argumentative elements as "*Ineffective*, *Adequate* and *Effective*". The essays of the given dataset[Franklin et al., 2022] were annotated by expert raters for discourse elements commonly found in argumentative writing: "*Lead*, *Position*, *Claim*, *Counterclaim*, *Rebuttal*, *Evidence* and *Concluding Statement*".

[*]This work is a feedback prize competition(2022) on Kaggle, see Feedback Prize - Predicting Effective Arguments.

[†]Equal Contribution. Listing order is random. See section 6 for details.

There are two typical existing strategies for text classification tasks: *supervised-learning* and *fine-tuning*. The *supervised-learning* approach, such as TextCNN[Kim, 2014], TextRNN[Liu et al., 2016] and HAN[Yang et al., 2016], use task-specific architectures trained on specific datasets from scratch. These methods take texts or documents as input and output a probability distribution over a certain label set, and they aim to minimize the negative log-likelihood(NLL) loss. The *fine-tuning* methods, such as BERT[Devlin et al., 2018] and DeBERTa[He et al., 2020], introduce minimal task-specific parameters along with a pre-trained model trained on large-scale datasets and are further trained on the downstream classification tasks by simply fine-tuning all pre-trained parameters. The two approaches share the same training objectives, where they may learn bidirectional or unidirectional language representations at a certain layer's output.

We argue that current techniques limit the power of deep models. As for the *supervised-learning* methods, their performances are limited by the scale of training datasets, and this may lead to a bad sentence representation relatively[Kowsari et al., 2019]. Moreover, typical text classification models, such as CNN-based or RNN-based models, only learn a representation unidirectionally or locally, which achieves unsatisfying performance on texts with very long lengths. As for the *fine-tuning* methods, there's always a gap between their pre-trained datasets and downstream datasets, which causes a mismatch of data distribution[Poth et al., 2021]. This mismatch restricts the power of the pre-trained representations and is sub-optimal for sentence-level tasks.

In this paper, we improve the *fine-tuning* approaches by applying prompting method[Petroni et al., 2019, Schick and Schütze, 2020, Gao et al., 2020]. Unlike traditional supervised learning, which trains a model to take in an input $\boldsymbol{x}$ and predict an output $\boldsymbol{y}$ as $p(\boldsymbol{y}|\boldsymbol{x})$, prompting is based on language models that model the probability of text directly. To use these models to perform prediction tasks, the original input $\boldsymbol{x}$ is modified using a *template* into a textual string prompt $\boldsymbol{x}'$ that has some unfilled slots, and then the language model is used to o probabilistically fill the unfilled information to obtain a final string $\hat{\boldsymbol{x}}$, from which the final output $\boldsymbol{y}$ can be derived. The contributions of our work are as follows:

1. We demonstrate the importance of prompt-based learning on large pre-trained models on argument classification tasks, which can be further extended to other text classification tasks with few modifications.
2. We show that prompting methods allows the language model to be pre-trained on massive amounts of raw text. And by defining a new prompting function, the model can perform *few-shot* or even *zero-shot* learning.
3. Prompt-based method achieves state-of-the-art results on arguments classification tasks, outperforming the traditional supervised methods or fine-tuned classification models.

2 Related Work

2.1 Text Classification Models

Deep learning models have achieved state-of-the-art results across many domains, including a wide variety of NLP applications. Deep learning for text and document classification includes three basic architectures of deep learning in parallel. We describe typical models in detail below. Given a text x and a target class label $\tilde{y}$, deep text classification models aim to minimize the Negative Log-Likelihood Loss(NLL):

$$-\log L = -\sum_{i=1}^{M} t_i \log p(y_i|x) \tag{1}$$

where M is the number of class label, t_i is 1 when text x belongs to class y_i and 0 otherwise. $p(y_i|x)$ is the predicted probability that x belongs to class i, and it is commonly computed by a `Softmax` operation on the outputs of deep models.

Recurrent Neural Network (RNN) A RNN[Rumelhart et al., 1986] considers the information of previous nodes in a very sophisticated method which allows for better semantic analysis of a data set's structure. Therefore, this technique is a powerful method for text, string, and sequential data processing. RNN mostly works by using LSTM[Hochreiter and Schmidhuber, 1997] or GRU[Gers et al., 1999] for text classification[Liu et al., 2016].

Convolutional Neural Networks (CNN) Although originally built for image processing, CNNs have also been effectively used for text classification. TextCNN[Kim, 2014] used a simple single-layer convolution neural network on top of *word2vec*[Mikolov et al., 2013] on sentiment classification tasks and achieved excellent performance.

Hierarchical Attention Networks (HAN) The structure of a HAN[Yang et al., 2016] focuses on the document-level classification where the lower level contains word encoding and word attention and the upper level contains sentence encoding and sentence attention.

2.2 Pre-trained Language Models

2.2.1 Masked Language Model

While autoregressive language models provide a powerful tool for modeling the probability of text, they also have disadvantages such as requiring representations to be calculated from left-to-right[Liu et al., 2021a]. Large-scale Transformer-Encoder-based PLMs, which are composed of stacked Transformer Encoder blocks[Vaswani et al., 2017], are typically pre-trained on large amounts of text to learn contextual word representations using a self-supervision objective, known as MLM[Devlin et al., 2018]. Specifically, given a sequence $\boldsymbol{X} = \{x_i\}$, we corrupt it into $\tilde{\boldsymbol{X}}$ by masking 15% of its tokens at random and then train a language model parameterized by θ to reconstruct $\boldsymbol{X}$ by predicting the masked tokens $\tilde{x}$ conditioned on $\tilde{\boldsymbol{X}}$:

$$\max_\theta \log p_\theta(\boldsymbol{X}|\tilde{\boldsymbol{X}}) = \max_\theta \sum_{i \in C} \log p_\theta(\tilde{x_i} = x_i|\tilde{\boldsymbol{X}}) \tag{2}$$

where C is the index set of the masked tokens in the sequence. BERT[Devlin et al., 2018] is a typical MLM. The authors of BERT propose to keep 10% of the masked tokens unchanged, another 10% replaced with randomly picked tokens, and the rest replaced with the `[MASK]` token.

2.2.2 DeBERTa

DeBERTa[He et al., 2020] improves BERT with disentangled attention and an enhanced mask decoder. The disentangled attention mechanism differs from existing approaches in that it represents each input word using two separate vectors: one for the content and the other for the position. Meanwhile, its attention weights among words are computed via disentangled matrices on both their contents and relative positions. Like BERT, DeBERTa is pre-trained using masked language modeling. The disentangled attention mechanism already considers the contents and relative positions of the context words, but not the absolute positions of these words, which in many cases are crucial for the prediction. DeBERTa uses an enhanced mask decoder to improve MLM by adding absolute position information of the context words at the MLM decoding layer. Further, DeBERTaV3[He et al., 2021], combining DeBERTa with ELECTRA[Clark et al., 2020], introduced gradient-disentangled embedding sharing as a new building block to avoid the "tug-of-war" issue and achieve a better pre-training efficiency.

2.3 Prompting Method

The main issue with supervised learning is that to train a model $p(\boldsymbol{y}|\boldsymbol{x};\theta)$, it is necessary to have supervised data for the task, which for many tasks cannot be found in large amounts. Prompt-based learning methods for NLP attempt to circumvent this issue by instead learning an LM that models the probability $p(\boldsymbol{x};\theta)$ of text $\boldsymbol{x}$ itself and using this probability to predict $\boldsymbol{y}$, reducing or obviating the need for large supervised datasets. Specifically, basic prompting predicts the highest-scoring $\hat{\boldsymbol{y}}$ in three steps.

Prompt Addition In this step a prompting function $f_{prompt}(\cdot)$ is applied to modify the input text $\boldsymbol{x}$ into a *prompt* $\boldsymbol{x}' = f_{prompt}(\boldsymbol{x})$. In the majority of previous work[Kumar et al., 2016, McCann et al., 2018, Schick and Schütze, 2020], this function consists of a two step process:

1. Apply a *template*, which is a textual string that has two slots: an input slot `[X]` for input $\boldsymbol{x}$ and an answer slot `[Z]` for an intermediate generated answer text $\boldsymbol{z}$ that will later be mapped into $\boldsymbol{y}$.
2. Fill slot `[X]` with the input text $\boldsymbol{x}$.

Table 1: Comparasion between various text-classification models

Methods	Test Accuracy	Scalability	Convergence
Supervised-learning	✔	✘	Slow
Pre-train+Fine-tune	✔✔	✔	Fast
Pre-train+Prompt	✔✔	✔✔	No Need
Pre-train+Prompt+Fine-tune	✔✔✔	✔	Fast+

Answer Search Next, we search for the highest-scoring text $\hat{\boldsymbol{z}}$ that maximizes the score of the LM. We first define Z as a set of permissible values for $\boldsymbol{z}$. Z could range from the entirety of the language in the case of generative tasks or could be a small subset of the words in the language in the case of classification. We then define a function $f_{fill}(\boldsymbol{x}', \boldsymbol{z})$ that fills in the location `[Z]` in prompt $\boldsymbol{x}'$ with the potential answer $\boldsymbol{z}$. Finally, we search over the set of potential answers $\boldsymbol{z}$ by calculating the probability of their corresponding filled prompts using a pre-trained LM $p(\cdot;\theta)$.

$$\hat{\boldsymbol{z}} = \max_{\boldsymbol{z}\in Z} p(f_{fill}(\boldsymbol{x}', \boldsymbol{z}); \theta) \tag{3}$$

Answer Mapping Finally, we would like to go from the highest-scoring answer $\hat{\boldsymbol{z}}$ to the highest-scoring output $\hat{\boldsymbol{y}}$. For example, one may use multiple different sentiment-bearing words (e.g. 'excellent', 'fabulous', 'wonderful') to represent a single class (e.g. '++'), in which case it is necessary to have a mapping between the searched answer and the output value.

3 Method

3.1 Overview

We introduce a prompt-based text-classification method on pre-trained deep models. There are three steps in our framework: *Prompt Addition*, *Answer Search*, and *Answer Mapping*. We use pre-trained models, such as BERT and DeBERTaV3 as our pipeline's backbone, and modified the models' input and output mapping to apply prompt learning. The pre-trained models are first initialized with their pre-trained parameters[3]. Thanks to the massive training samples in the pre-training stage, prompt methods can make full use of the models' ability to scale to different scenarios. Thus, prompt learning has advantages in both test accuracy, scalability, and convergence speed compared with other traditional text classification methods. A brief comparison is shown in table 1.

First, we defined a transformation template $f_{prompt}(\cdot)$, which turns the input text x to $x' = f_{prompt}(x)$ by fixed rules. Then our model $f_\theta(\cdot)$ outputs $p(z|x', \theta)$, $z \in Z$, where Z is the predicted label set of $f_\theta(\cdot)$. We define a fixed rule $f_{map}(\cdot)$ to transform the output $\tilde{z}$ to a certain given label $\tilde{y} = f_{map}(\tilde{z})$. Our entire pipeline can be formalized as:

$$\tilde{y} = f_{map}\left(\arg\max_{\tilde{z}\in Z} p\left(\tilde{z}|f_{prompt}(x), \theta\right)\right) \tag{4}$$

Our model optimizes the following target:

$$\min_\theta L = \min_\theta -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} t_{ij}\log p\left(f_{map}^{-1}(y_{ij})|f_{prompt}(x_i), \theta\right) \tag{5}$$

where N is the number of text fragments in a certain essay, and M is the number of class labels. t_{ij} is 1 if fragment i's classcification is class j and 0 otherwise, and $p(y_{ij}|x_i)$ is the predicted probability that observation i belongs to class j.

3.2 Prompting Design

In this section, we will introduce our prompting method using pre-trained models. The main procedures include *Prompt Addition*, *Answer Search*, and *Answer Mapping*.

[3]Thanks to open sources in machine learning, most of the popular pre-trained models' parameters can be found and downloaded at https://huggingface.co/models

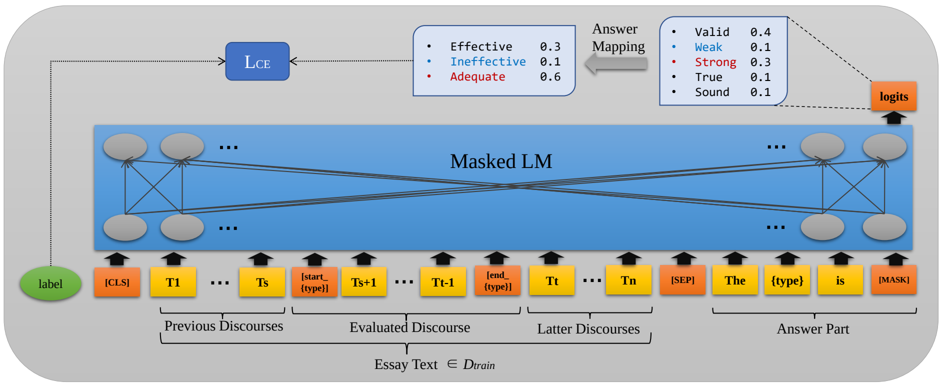

Figure 1: Overall prompting and fine-tuning procedures of the proposed method.

Prompt Addition Firstly, the input essay will be split into sub-discourses according to the split of training samples. Then, special tokens, ''`[START_{type}]`'' and ''`[END_{type}]`'', are added to the vocabulary for notation of the beginning and end of the classified discourse, where `{type}` is one of `{LEAD, POSITION, CLAIM, COUNTERCLAIM, REBUTTAL, EVIDENCE, CONCLUSION}` to clarify the type of the classified discourse. Finally, a prompt query, ''`The discourse is [MASK]`'', is appended to the input text. And there is a `[SEP]` between the essay text and the query. The pre-trained model will give the words' probability over the vocabulary at position `[MASK]`.

Answer Search and Answer Mapping As tested using the open APIs, the MLM models are most likely to present the highest probabilities of `{valid, weak, strong, true, sound}`. So we design an answer mapping to map these words into `{effective, ineffective, adequate}`. That is, `valid, true, sound` is mapped to `effective`, `strong` is mapped to `adequate`, `weak` is mapped to `ineffective`. And we normalized the sum of these probabilities to 1.0. We fine-tuned the model with the negative log-likelihood loss on training samples. Overall prompting and fine-tuning procedures of the proposed method are shown in Figure 1.

3.3 Pre-trained Model Choice

There is a wide variety of pre-trained LMs that could be used to calculate $P(\boldsymbol{x}|\theta)$. While autoregressive language models provide a powerful tool for modeling the probability of text, they also have disadvantages such as requiring representations to be calculated from left-to-right[Markov, 2006]. When the focus is shifted to generating the optimal representations for downstream tasks such as classification, many other options become possible, and often preferable. One popular bidirectional objective function used widely in representation learning is the masked language model(MLM), which aims to predict masked text pieces based on the surrounding contexts. For example, $P(x_i|x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$ represents the probability of the word xi given the surrounding context. Popular pre-trained MLMs include: BERT[Devlin et al., 2018], ERNIE[Sun et al., 2019], DeBERTa[He et al., 2020] and many variants. In prompting methods, MLMs are generally most suitable for natural language understanding or analysis tasks (e.g. text classification). These tasks are often relatively easy to be reformulated into cloze problems, which are consistent with the training objectives of the MLM. Additionally, MLMs have been a pre-trained model of choice when exploring methods that combine prompting with fine-tuning, elaborated further in section 3.4.

We used BERT and DeBERTaV3 as our backbone for their strong language understanding ability in English texts. BERT(Base) has 110M parameters and 12 bidirectional transformer-encoder blocks with a hidden size of 768. DeBERTa improves the BERT and RoBERTa models using disentangled attention and enhanced mask decoder. DeBERTaV3 further improved the efficiency of DeBERTa using ELECTRA-Style pre-training with *Gradient Disentangled Embedding Sharing* [He et al., 2021], which significantly improves the model performance on downstream tasks. Both BERT and DeBERTaV3 are pre-trained on BooksCorpus (800M words)[Zhu et al., 2015] and English Wikipedia (2,500M words).

3.4 Parameter Update Methods

Table 2: Characteristics of various parameter update methods

Methods	LM Params	Prompt Params
Tuning-free Prompting	Frozen	No Need
Prompt-less Fine-tuning	Tuned	No Need
Fixed-LM Prompt Tuning	Frozen	Tuned
Prompt+LM Fine-tuning	Tuned	Tuned

In our prompt-based classification task, there are two types of parameters, namely those from (1)pre-trained models and (2) prompt. Which part of parameters should be updated in prompting may affect the prediction performance.

We applied three techniques to leverage the power of both pre-training and prompting: (1)*Prompt-less Fine-tuning*, (2)*Tuning-free Prompting*, (3)*Fixed-LM Prompt Tuning* and (4)*Prompt+LM Fine-tuning*. *Prompt-less Fine-tuning* trained the LM's parameters will be updated via gradients induced from downstream training samples, after which the prompt method will be used to give predictions. *Tuning-free Prompting* uses the original parameters from pre-trained models and applies the prompting method directly into the pipeline, and no training is required. In *Fixed-LM Prompt Tuning*, additional prompt-relevant parameters are introduced besides the pre-trained parameters, and this method updates only the prompts' parameter using a supervision signal from downstream training samples. Here we applied P-Tuning[Liu et al., 2021b] to our model, which only tuned a few embedding weights at the very beginning. *Prompt+LM Fine-tuning* updated both LM parameters and prompts' parameters simultaneously. Similarly, we use fine-tuned hidden embedding weights of P-tuning as well as the pre-trained backbone. All prompt parameters in *Fixed-LM Prompt Tuning* and *Prompt+LM Fine-tuning* come from the hidden prefix embedding layer. The characteristics of these strategies are shown in table 2.

4 Experiment

4.1 Experiment Setting

Datasets The dataset presented here contains argumentative essays written by U.S. students in grades 6-12. These essays were annotated by expert raters for discourse elements commonly found in argumentative writing:

1. **Lead**: an introduction that begins with a statistic, a quotation, a description, or some other device to grab the reader's attention and point toward the thesis.
2. **Position**: an opinion or conclusion on the main question.
3. **Claim**: a claim that supports the position.
4. **Counterclaim**: claim that refutes another claim or gives an opposing reason to the position.
5. **Rebuttal**: a claim that refutes a counterclaim.
6. **Evidence**: ideas or examples that support claims, counterclaims, or rebuttals.
7. **Concluding Statement**: a concluding statement that restates the claims.

Given a certain essay, our model will give the quality rating of each discourse element as one of "*Ineffective*, *Adequate*, and *Effective*". The dataset has 4191 essays with a total of 36766 valid argumentative discourses, which is adequate for fine-tuning and testing.

To avoid overfitting problems, we apply the 10-fold[McLachlan et al., 2005] splitting method on the given dataset, which randomly selects 90% of the dataset as Train set and the rest data as Dev set. For each fold, we fine-tuned the prompting model with the best parameters on each Dev set. We gathered each model's output probability $P(\boldsymbol{y}|\boldsymbol{x})$ and calculate their average as the final prediction result.

Parameters and Settings For *fine-tune* scenarios, we used a batch size of 64 and fine-tuned for 5 epochs for each backbone pre-trained model. We selected the best fine-tuning learning rate *1e-5* on the Dev set. We use Adam[Kingma and Ba, 2014] as our optimizer with $eps = 10^{-6}, beta = (0.9, 0.98), weight\_decay = 0.01$ and warm up[He et al., 2015] for 1 epoches. We clip the gradients at an L2-norm of 1.0. For *zero-shot* scenarios, we directly use the outputs and take their average as the final prediction. We use 1 NVIDIA A100 GPU (80GB) to train our models.

Baselines We use both supervised models and fine-tuned classification models as our baselines. For supervised models, we trained TextRNN with 1-layer bidirectional LSTM and fed the last cell's output into a linear layer for classification. And we trained TextCNN with parameters recommended by the original paper. We trained supervised models at a learning rate of *1e-2* and stopped at the best loss on the Dev set. For fine-tuned classification models, we fed the `[CLS]` encoding from BERT or DeBERTa into a linear classification layer. We trained the classification parameters at a learning rate of *1e-4* and fine-tuned the pre-trained model at a learning rate of *1e-5*. For both settings, we use Adam for optimization with the same parameters as the proposed method.

4.2 Performance on Various Models

Results of proposed models and baselines are presented in table 3. Both BERT+*prompt* and DeBERTa+*prompt* outperform all systems by a substantial margin, obtaining +2.2% and +2.1% average accuracy improvement over the head-based pre-trained models respectively. And we found that DeBERTaV3 significantly outperforms BERT's, showing DeBERTaV3's strong ability on downstream tasks. 10-fold cross-validation also improves performance slightly for +1.2% accuracy. The NLL, F1, and recall also show similar performance.

Table 3: Performance on various Models. *TextCNN* and *TextRNN* are supervised trained on Train Set. {MODEL}+*head-based* are head-based fine-tuned classification models on Train Set. {MODEL}+*prompt* are models trained by the prompting method which are fine-tuned on Train Set.

System	NLL↓	Accuracy↑	Recall(macro)↑	F1(macro)↑
TextCNN[Kim, 2014]	0.809	62.6	47.8	48.2
TextRNN[Liu et al., 2016]	0.796	63.1	47.6	48.2
BERT+*head-based*	0.762	65.4	55.6	55.9
DeBERTaV3+*head-based*	0.735	66.2	54.2	54.7
BERT+*prompt* (ours)	0.726	67.6	55.9	56.5
BERT+*prompt+10fold* (ours)	0.720	68.0	56.1	56.8
DeBERTaV3+*prompt* (ours)	0.715	68.3	56.3	56.7
DeBERTaV3+*prompt+10fold* (ours)	**0.701**	**69.5**	**56.9**	**57.6**

4.3 Ablation Study

In this section, we perform ablation experiments over several facets of our prompting method to better understand their relative importance.

4.3.1 Effect of fine-tuning method

In this section, we explored the effect of fine-tuning methods on the same metrics as section 4.2. We trained several models with different train set sizes, namely *zero-shot*, *few-shot*, and *fine-tune*. *Zero-shot*, *few-shot* and *fine-tune* models are trained on **0**, **10**(2 classes, 10 samples each class) and **100%**(all 36766 samples from 3 classes).

Results are shown in table 4. We can see that the *fine-tune* models lead to a strict accuracy improvement across all metrics. It is also perhaps surprising that the *few-shot* method can achieve close performance to those of *fine-tune* models, indicating a strong language understanding ability of large-scale pre-trained models. This also demonstrates that the prompting method can make full use of pre-trained models' abilities.

Table 4: Ablation over the *fine-tune*, *few-shot* or *zero-shot* methods using the DeBERTaV3-base model. *Few-shot* method is trained on only 2 classes (`effective` and `ineffective`) of the train set which contains 10 training samples each (2-way, 10-shot).

Methods	NLL↓	Accuracy↑	F1(macro)↑	Recall(macro)↑
DeBERTaV3+zero-shot	1.260	56.5	33.2	24.5
DeBERTaV3+few-shot	0.771	66.0	46.7	48.5
DeBERTaV3+fine-tune	**0.715**	**68.3**	**56.3**	**56.7**

4.3.2 Effect of parameter update method

Different parameter update methods apply for different scenarios. In this section, we gathered performances of each parameter update method mentioned in section 3.4. Results are presented in table 5. It can be seen that methods with tuned LM parameters outperform others in accuracy. And a simple transformation from a fixed prompt template to learnable embedding parameters(a.k.a P-tuning[Liu et al., 2021b]) also improves the accuracy. This indicates learning prompts from a continuous space are more practical. Future works may explore the effect of introducing a larger number of prompts' parameters with more complicated or more efficient pipelines.

Table 5: Ablation over the parameter update methods using DeBERTaV3 backbone, see their details in section 3.4.

Methods	NLL↓	Accuracy↑	F1(macro)↑	Recall(macro)↑
Tuning-free Prompting	1.260	56.5	33.2	24.5
Prompt-less Fine-tuning	0.715	68.3	56.3	56.7
Fixed-LM P-Tuning[Liu et al., 2021b]	**0.704**	69.1	**56.9**	**57.2**
P-Tuning + LM Fine-tuning	0.706	**69.2**	56.8	56.9

4.3.3 Effect of pre-trained model size

It has long been known that increasing the model size will bring further improvements in large-scale tasks such as machine translation and language modeling. In this section, we explored the effect of the pre-trained model's size on downstream tasks. We fine-tuned several DeBERTaV3 models under the prompting method. Results are shown in table 6. We can see that larger models have strict improvements in both accuracy(+1.0%) and NLL(-0.016%). This result shows that extreme model sizes can also achieve big improvements on small-scale classification tasks.

Table 6: Ablation over pre-trained model sizes.

Models	Params	NLL↓	Accuracy↑	F1(micro)↑	Recall(micro)↑
DeBERTaV3-xsmall	22M	0.774	67.5	54.5	54.9
DeBERTaV3-small	44M	0.766	67.9	56.8	57.7
DeBERTaV3-base	86M	0.715	68.3	56.3	56.7
DeBERTaV3-large	304M	**0.699**	**69.3**	**59.6**	**59.7**

5 Conclusion

Impressive results have improved the automation of language education with the application of large pre-trained models. Our major contribution is making full use of pre-trained models by applying the prompting method, achieving a state-of-the-art result on a certain text classification task. Our research shows promising results on deep pre-trained models, outperforming traditional head-based fine-tuning approaches by +3.3% accuracy. Future works may focus on extending learnable prompts to general text-generation tasks or even knowledge-graph-based tasks. Automated answer mapping or searching may also improve our method's performance.

6 Contribution Attribution

Equal Contribution. **Hongzun**, who proposed the BERT-based prompting method and started the effort to evaluate this idea, was responsible for our codebase and further designed, implemented, and tuned the proposed system. **Xuetong** and **Jie** designed and implemented various baselines and evaluated countless model variants in our codebase, and took part in every aspect of this work, which crucially accelerated our research. Together, we spent countless days tuning various parts in nearly every detail and accomplished our paper and presentation.

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555, 2020. URL https://arxiv.org/abs/2003.10555.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Alex Franklin, Maggie, Meg Benner, Natalie Rambis, Perpetual Baffour, Ryan Holbrook, and ulrichboser Scott Crossley. Feedback prize - predicting effective arguments, 2022. URL https://kaggle.com/competitions/feedback-prize-effectiveness.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999. doi: 10.1049/cp:19991218.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654, 2020. URL https://arxiv.org/abs/2006.03654.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543, 2021. URL https://arxiv.org/abs/2111.09543.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR, 2016.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021b. URL https://arxiv.org/abs/2103.10385.

Andreui Andreevich Markov. An example of statistical investigation of the text eugene onegin concerning the connection of samples in chains. *Science in Context*, 19(4):591–600, 2006.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

Geoffrey J McLachlan, Kim-Anh Do, and Christophe Ambroise. Analyzing microarray gene expression data. 2005.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. URL http://arxiv.org/abs/1310.4546.

Michael A Peters. Deep learning, education and the final stage of automation, 2018.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. What to pre-train on? efficient intermediate task selection. *arXiv preprint arXiv:2104.08247*, 2021.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*, 2020.

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. *CoRR*, abs/2001.07676, 2020. URL https://arxiv.org/abs/2001.07676.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223, 2019. URL http://arxiv.org/abs/1904.09223.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.