Context-Aware Prompt: Customize A Unique Prompt For Each Input

Anonymous ACL submission

Abstract

After the proposal of BERT, pre-trained language models have become the dominant approach for solving many NLP tasks. Typically, a linear classifier is added to the head of the model for fine-tuning to fit downstream tasks, while a more recent approach, also known as prompt-based learning or prompt-learning, using prompts to perform various downstream tasks, is considered to be able to uncover the potential of the language model.

Prior study, however, attempted to find a universal prompt for a certain task across all samples. Therefore, we propose a novel method, Context-Aware Prompt (CAP), which provides a unique continuous prompt for each sample input by combining contextual information to further investigate the potential capabilities of the language models. On the SuperGlue benchmark, our method outperforms multiple models with vanilla fine-tuning. Furthermore, we extend the use of prompts to include Replaced Token Detection (RTD) type prompts, allowing models like ELECTRA and DeBERTaV3 that employ RTD as a training objective to use prompts for downstream tasks.[1]

1 Introduction

Due to their outstanding performance in downstream tasks such as question answering (Rajpurkar et al., 2016), named entity recognition (Sang and Meulder, 2003), and text classification (Sun et al., 2019), pre-trained language models (Devlin et al., 2018; Liu et al., 2019; Raffel et al., 2020; He et al., 2021b) have gained increasing importance and become the primary way to solve various natural language processing (NLP) tasks in recent years.

How to leverage these language model effectively has been a struggle for researchers to figure out. The typical approach is to add a linear classifier to the head of the model and then adapt the parameters of the linear classifier and the pre-trained language model to the supervised target task, also known as fine-tuning (Radford et al., 2018).

More recently, a paradigm, known as prompt-based learning or prompt learning (Liu et al., 2021a), has had great success in zero-shot and few-shot learning. In this way, a task description is provided for downstream tasks that not only more closely resembles the human way of thinking, but also aids the language model to "understand" what the task purpose is (Radford et al., 2019). These results suggest that when prompted by relevant task descriptions, pre-trained language models can solve NLP problems more effectively. As a result, we believe that prompts can be used to improve performance on a variety of NLP tasks.

However, finding a prompt that works for all samples is very difficult, as shown in Table 1 (a sentiment classification task): a prompt that works for one sample may not work for another, although the reason for the results in the table is most likely a bias caused by the pre-training process: "It is" is usually followed by "great", while "It was" is usually followed by "terrible". Accordingly, we strive to alleviate this bias in the pre-trained language model by using contextual information from the input sequence to automatically generate prompts that are suitable for independent samples.

In this paper, we introduce **Context-Aware Prompt (CAP)**, a novel method for generating a unique continuous prompt for each sample input through context-awareness and is based on the extension of P-tuning (Liu et al., 2021c). We construct continuous prompts via the prompt encoder in CAP, which uses contextual information to create different prompts for each sample input by contextualized embeddings that decouples from the pre-trained language model word embeddings. Correspondingly, we also need an answer mapping,

[1] Our code is available at https://anonymous.4open.science/r/CAP_01. To avoid violating the anonymized review principle, an anonymous repository is employed.

Input Example	Prompt	Predict	Label
A sometimes tedious film .	It is [MASK] .	terrible	negative
	It was [MASK] .	great	
Among the year 's most intriguing explorations of alientation .	It is [MASK] .	terrible	positive
	It was [MASK] .	great	

Table 1: Examples from SST-2 (Socher et al., 2013), which are predicted by BERT-base, and verbalizers set as "great" and "terrible".

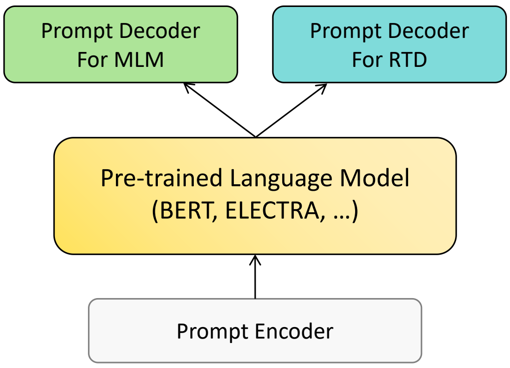


Figure 1: Illustration of CAP.

which we call prompt decoder, to map the output of the pre-trained language model to a specific label. Further, we expand the way of using prompts in models like ELECTRA (Clark et al., 2020), DeBERTaV3 (He et al., 2021a) — prompt decoder for Replaced Token Detection (RTD).

Experiments show that our proposed method achieves better performance than vanilla fine-tuning on most tasks in the SuperGLUE (Wang et al., 2019) benchmark. Moreover, our approach changes the parameters of the pre-trained language model to a lesser level than vanilla fine-tuning, which is why we believe that prompt-based learning works well.

2 Related Work

The idea of prompt was initially proposed by Radford et al. (2019) to "recall" knowledge learned during the training of pre-trained language models and apply it to downstream tasks in an unsupervised way. Schick and Schütze (2021) employ cloze problem modeling for text classification and natural language inference. Taking a sentiment classification task as an example, constructs the input as "Best pizza ever! It was ___.", let the pre-trained language model predict the text in the blank. The sentence is regarded positive if the probability of predicting the text as "great" is larger than the probability of predicting the text as "bad" and vice versa. The "Best pizza ever!" is the sample to be classified; "It was ___." and "great" or "bad" are called pattern and verbalizers by Schick and Schütze (2021), respectively.

However, the preceding researches rely on manual prompt creation, which necessitates a great deal of prior knowledge and experimental validation, and the cost of determining which pattern-verbalizer pair (PVP) performs best is extremely expensive. To address this issue, some of the researches look at ways to automatically find the appropriate prompts, which can be divided into two types: discrete prompts and continuous prompts.

2.1 Discrete Prompt

AutoPrompt (Shin et al., 2020) employs gradient-guided search to create prompts for various tasks; LPAQA (Jiang et al., 2020) adopts a mining-based and paraphrase-based approach to prompt generation; LM-BFF (Gao et al., 2021) utilises the generative T5 (Raffel et al., 2020) model to automatically generate templates, all of which have achieved promising results and have contributed significantly to automatically search for prompts, but they limit prompts to discrete prompts that humans can understand. However, do machines need discrete prompts that humans can understand?

2.2 Continuous Prompt

As a matter of fact, prompts are employed to pre-train language models to understand the task purpose rather than supplying them to humans, so it is not necessary to stick to discrete prompts as perceived by humans, but rather continuous prompts are composed of virtual words embedded in a continuous space can be used.

Initialized Prompt WARP (Hambardzumyan et al., 2021), OPTIPROMPT (Zhong et al., 2021), Prompt Tuning (Lester et al., 2021) randomly initialize the word embeddings of the pseudo-tokens or by existing real word embeddings, and optimize the word embedding vector of the pseudo-tokens by gradient descent in the word embedding space. In addition, WARP (Hambardzumyan et al., 2021) also constructs the continuous verbalizer, which is free of the constraints imposed by the human cognitive discrete word.

Generated Prompt After pre-training, the word embeddings of the pre-trained language model have been highly discretized. When the pseudo-token is initialized in the above ways, Allen-Zhu et al. (2019) have shown that the relevant parameters will only be changed within a small domain and are prone to fall into local minima. Therefore, P-tuning (Liu et al., 2021c) use bidirectional long-short term memory networks (BiLSTM) (Graves et al., 2013) to generate the embeddings of the prompt token, obtains satisfactory performance.

Whereas, P-tuning necessitates the addition of a few anchor tokens in order to improve performance further, which is in direct opposition to the intention of the automatic search for prompts. In contrast, our proposed CAP does not require any anchor token, and we believe that prompt tokens generated using CAP are more useful for machine comprehension, rather than relying on prior knowledge, as demonstrated in subsequent experiments (Table 4).

Prefix-tuning (Li and Liang, 2021) trains an upstream prefix across layers and freezes the pre-trained language model, saving storage costs, but it is designed for natural language generation (NLG) tasks and autoregressive LM. Liu et al. (2021b) have extended it to NLU tasks and achieved performance comparable to fine-tuning. Similar to adapter (Houlsby et al., 2019), their motivation is parameter-efficiency, which reduces the storage cost of the language model during training. Nevertheless, their approach requires large training epochs, which raises the cost of training for downstream tasks. That is different from our starting point, where we propose that CAP involves fully training the pre-trained language model for improving performance, making downstream tasks are more relative to the training objectives of the pre-training phase, and thus can be an alternative to fine-tuning for certain tasks.

3 CAP

In this section, we present the implementation of Context-Aware Prompt (CAP), which consists of two components: a prompt encoder and a prompt decoder, the former generates prompt embeddings, while the latter decodes the hidden states output from the pre-trained language model into the corresponding labels of the task. Let $\mathcal{M}$ be a pre-trained language model with vocabulary $\mathcal{V}$ and pre-trained embedding layer $E_{\mathcal{M}} \in \mathcal{M}$; let $\mathcal{L}$ be a set of labels for our target classification task A, and assign a verbalizer token $V_l \notin \mathcal{V}$ to each label $l \in \mathcal{L}$.

Overall, we need to optimize the parameters $\Theta = \{\Theta_P, \Theta_V, \Theta_{\mathcal{M}}\}$ by gradient descent, for the prompt encoder, the verbalizer embeddings, and the pre-trained language model, respectively.

3.1 Prompt Encoder

Since a direct update of the prompt embeddings would result in parameters that vary only within a small neighborhood and we need a BiLSTM to incorporate contextual information, we propose a prompt encoder to generate prompt embeddings. Given a sequence of sample input tokens $\mathbf{x} = \{x_0, x_1, x_2, ..., x_n\}$, which will be mapped to input embeddings $E_{\mathcal{M}}(\mathbf{x}) = \{e_0, e_1, e_2, ..., e_n\}$, here x_0 refers to the token that can represent the semantics of the whole sentence, which is "[CLS]" in BERT.

We can flexibly insert prompt tokens between a given input sequence and links with label token ([LABEL]), where the label token in MLM and RTD denotes masked token ([MASK]) and verbalizer token (V_l), respectively. For example, let $[P_i]$ refers to the i^{th} prompt token, given the input $\mathbf{x}$ one can compose the template T = $\{[P_1], x_1, ..., x_n, [P_2], \text{[LABEL]}, [P_3], x_{n+1}, ..., x_m, [P_4]\}$. Note that the prompt tokens here are not discrete, the prompt embeddings will be inserted into the appropriate spot.

Naturally, our research turns into finding a set of parameters Θ_P for generating continuous prompt embeddings that allow the pre-trained language model to predict the expected answer to a masked token (for MLM) or determine whether the verbalizer token is replaced (for RTD). Finally, the continuous prompt embeddings generated by the prompt encoder, which is influenced by this set of parameters and inputs, are fed into the pre-trained language model with the embeddings of the original inputs.

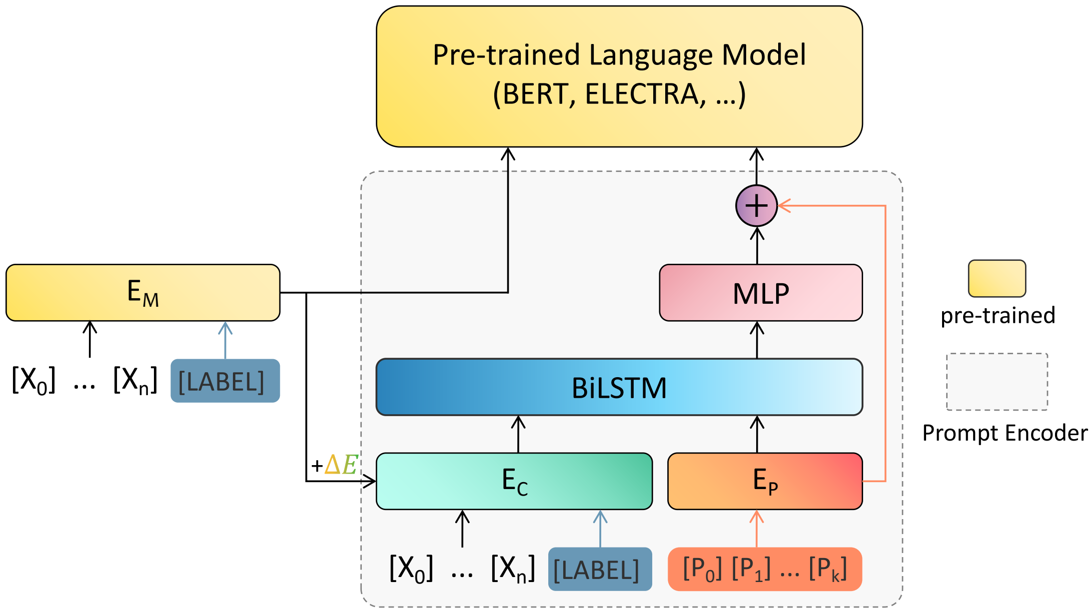


Figure 2: Prompt Encoder in CAP.

3.1.1 Continuous Prompt Embeddings

Customizing the prompt for each input and enabling the prompt to fully incorporate contextual information, as illustrated in Figure 2, we employ BiLSTM to attempt to pass contextual information to the prompt and comply with P-tuning (Liu et al., 2021c) using MLP to encourage discretization. Besides, we establish a shortcut connection (He et al., 2016) between the raw prompt embeddings and the generated embeddings. For one thing, it prevents the vanishing gradient to raw prompt embeddings. And for another, we employ an idea similar to ELMo (Peters et al., 2018) which combines the internal states of each layer for rich word representation, generating prompt embeddings that combine the raw prompt embeddings with the prompt embeddings that have combined contextual information.

So the continuous prompt embeddings $\mathbf{PE}$ can be given by:

$$\begin{aligned}\mathbf{PE} &= PE_{\Theta_P}(\mathbf{x}) \\ &= \mathrm{MLP}(\mathrm{BiLSTM}[E_{\mathcal{C}}(\mathbf{x}), E_{\mathcal{P}}(\mathbf{P})]) \\ &\quad + E_{\mathcal{P}}(\mathbf{P})\end{aligned} \tag{1}$$

where PE_{Θ_P} means Prompt Encoder, $E_{\mathcal{C}}$ and $E_{\mathcal{P}}$ mean contextualized embedding layer and prompt embedding layer respectively, and $\mathbf{P} = \{[P_1], [P_2], ..., [P_k]\}$ refers to prompt sequence.

3.1.2 Contextualized Embeddings

Since the word embedding layer of the pre-trained language model and the contextualized embedding layer used by CAP are strongly related to each other but have radically different training objectives, they form what He et al. (2021a) call "tug-of-war" dynamics.

Further, inspired by their proposed Gradient-Disentangled Embedding Sharing (GDES) method, we adopt a similar strategy to share the word embedding layer of the pre-trained language model ($E_{\mathcal{M}}$) with the contextualized embedding layer ($E_{\mathcal{C}}$) used by CAP but decouple them with stopping gradients in the CAP's contextualized embeddings from back-propagating to the pre-trained language model's word embeddings.

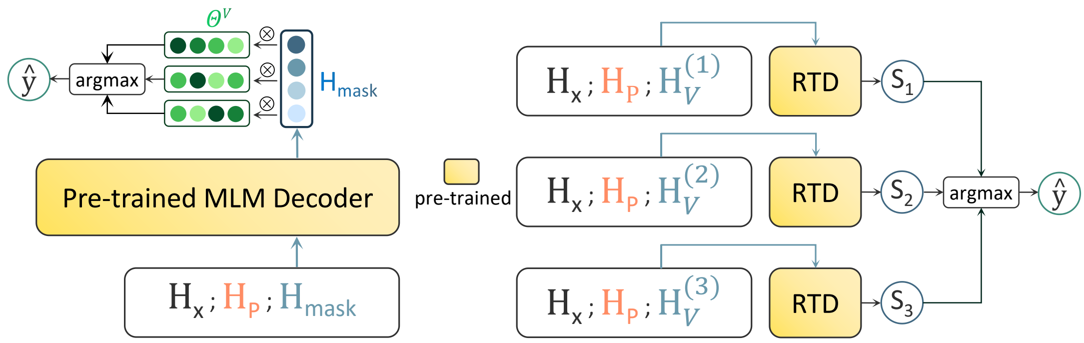
In short, the contextualized embedding can be expressed as:

$$E_{\mathcal{C}} = sg(E_{\mathcal{M}}) + \triangle E \tag{2}$$

where sg is the stop gradient operator which only allows gradients propagation through $\triangle E$. Noteworthy, $\triangle E$ is initialized as a zero matrix.

3.2 Prompt Decoder

After the generated prompt embeddings are sent to the pre-trained language model, a prompt decoder to parse the output of the hidden states is required to obtain the predicted labels.



(a) Prompt Decoder For MLM (b) Prompt Decoder For RTD

Figure 3: The method of prompt decoder for MLM and RTD solve classification tasks, where $H_{\mathbf{x}}$, H_P, H_{mask}, H_V attain the hidden states of x, the hidden states of prompt, the hidden state of masked token, and the hidden state of verbalizer token output by the pre-trained language model, respectively.

For each label, a verbalizer for answer mapping is required. It is worth noting that for MLM (Masked Language Model) [2], we follow WARP (Hambardzumyan et al., 2021) to replicate continuous verbalizers.

3.2.1 Prompt Decoder For MLM

Masked Language Model (MLM), such as BERT (Devlin et al., 2018), ALBERT (Lan et al., 2020), RoBERTa (Liu et al., 2019), typically masks a specific percentage of words in a given sentence, and the model predicts these masked words based on other remaining words in this sentence. So the last hidden state of the masked token can be easily obtained by MLM:

$$H_{mask} = \mathcal{M}(E_{\mathcal{M}}(\mathbf{x}), PE_{\Theta_P}(\mathbf{x})) \tag{3}$$

where H_{mask} refers to the last hidden state of the masked token and has been pooled by pre-trained MLM decoder.

And the probability of labels are given by:

$$P_{\Theta}(l \mid \mathbf{x}) = \frac{\exp(\Theta_l^V H_{mask})}{\sum_{l' \in \mathcal{L}} \exp(\Theta_{l'}^V H_{mask})},\ l \in \mathcal{L} \tag{4}$$

where Θ_l^V is the parameters of the verbalizer embedding corresponding to the label l.

3.2.2 Prompt Decoder For RTD

ELECTRA (Clark et al., 2020) and DeBERTaV3 (He et al., 2021a), applying Replaced Token Detection (RTD) as training objective, use a generator to generate ambiguous tokens and a discriminator to distinguish the ambiguous tokens from the original inputs, similar to Generative Adversarial Networks (GAN, Goodfellow et al. 2014).

For RTD, analogously to Candidates-Contrast proposed by Sun et al. (2021), we consider different verbalizer tokens as input label tokens, use RTD to detect verbalizer tokens, derive the score of each verbalizer token is not replaced, and consider the label corresponding to the verbalizer token with the highest score as the predicted label, which is as shown in Figure 3.

So the score can be considered as the negative of the score of the model output that detects as a replaced token:

$$S(l|\mathbf{x}) = -\mathcal{M}(E_{\mathcal{M}}(\mathbf{x}), PE_{\Theta_P}(\mathbf{x}), E_V(l)) \tag{5}$$

where $E_V(l)$ denotes the verbalizer embeddings representing the label l.

Resemble equation 4, the probability of labels are given by:

$$P_{\Theta}(l|\mathbf{x}) = \frac{\exp(S(l|\mathbf{x}))}{\sum_{l' \in \mathcal{L}} \exp(S(l'|\mathbf{x}))},\ l \in \mathcal{L} \tag{6}$$

This multiple choice approach, on the other hand, demands numerous calculations of the pre-trained language model (once for each label), implying that the training cost will be several times higher than MLM (depending on the total count of labels).

4 Experiments

We select six natural language understanding (NLU) tasks from the SuperGLUE (Wang et al.,

[2]It can be easily extended to Permuted Language Model (PLM) such as XLNet, which is not described in this paper.

Method	CB		RTE	BoolQ	WiC	WSC	MultiRC		Avg.
	Acc.	F1	Acc.	Acc.	Acc.	Acc.	EM	F1a	
BERT$_{base}$									
Vanilla Fine-tuning	89.3	89.0	70.0	75.2	71.6	63.5	17.9	66.0	68.6
P-tuning(Liu et al., 2021c)	89.2	92.1	71.1	73.9	68.8	63.5	14.8	63.5	67.9
CAP	100	100	73.6	76.2	71.9	63.5	19.2	67.8	71.5
RoBERTa$_{base}$									
Vanilla Fine-tuning	96.4	97.4	76.2	78.5	69.3	63.5	33.9	75.2	73.2
CAP	100	100	84.1	80.0	69.6	63.5	34.3	75.1	75.3
DeBERTaV3$_{base}$									
Vanilla Fine-tuning	94.6	93.7	84.8	83.3	74.1	63.5	49.4	82.1	77.6
CAP	100	100	86.3	86.3	70.7	63.5	50.3	82.5	78.9

Table 2: Dev set results on SuperGLUE tasks.

2019) benchmark for experiments to evaluate our method. CB (De Marneffe et al., 2019), RTE (Dagan et al., 2005), BoolQ (Clark et al., 2019), WiC (Pilehvar and Camacho-Collados, 2018), WSC (Levesque et al., 2012) and MultiRC (Khashabi et al., 2018) are among the tasks, which contain two textual entailment tasks (CB, RTE), two question answering tasks (BoolQ, MultiRC), a co-reference resolution task (WiC), and a word sense disambiguation task (WSC). These NLU tasks are reformulated as cloze problems, with prompt tokens inserted in the intervals between sentence1, label, and sentence2 (if it exists).

4.1 Experiment Settings

In our experiments, BERT and RoBERTa are used as MLM representatives and DeBERTaV3 as RTD representative, and the base-scale model (Layer=12, Hidden Size=768, Attention Head=12) is used uniformly. Pre-trained language models we use are from the Hugging Face Transformers (Wolf et al., 2020) library, and we employ the **AutoModelForSequenceClassification** it provides for fine-tuning as a baseline. To build CAP, the prompt embeddings for $[P_1], [P_2], ..., [P_n]$ are generated by prompt encoder, and verbalizer embeddings for $[V_1], ..., [V_L]$ are initialized with verbalizer token embeddings in pre-trained language model for their corresponding labels. For MLM, the bias of the verbalizer classifier is also initialized with the bias of the MLM classifier head of the pre-trained language model in order to be consistent with the pre-training phase.

We choose the AdamW optimizer with a learning rate that decreases linearly after a warmup period and train for 3-6 epochs on various task. Specifically, we set the learning rate from 2e-5, 3e-5, 4e-5, 5e-5 and batch size from 6, 8, 16, 24, 32.[3]

4.2 Results

The results are presented in Table 2. We can see that CAP outperforms vanilla fine-tuning on most tasks. For the average results, CAP outperforms vanilla fine-tuning by 2.9, 2.1, and 1.3 percent for BERT, RoBERTa, and DeBERTaV3, respectively. Besides, CAP also outperforms P-tuning. In particular, CAP outperforms vanilla fine-tuning more significantly in tasks such as CB, RTE and BoolQ, which are easy to construct prompts for. We then focus our subsequent experiments on these three tasks.

It can be seen that our method achieves relatively well performance in both the small sample size task (CB) and the multiple sample task (MultiRC). But for WSC, a word sense disambiguation task, our approach does not show any improvement over vanilla fine-tuning, and we conjecture that it is difficult to construct prompts that allow pre-trained language models to understand the purpose of such difficult tasks. The same is true for WiC, where there is only a small improvement when using BERT and RoBERTa, and even a drop in performance when using DeBERTaV3, so we get a similar conclusion to Liu et al. (2021c), that these types of more complex tasks are not suitable for

[3]To reproduce our work better, we pushed the hyperparameters we used in our experiments to our open-source code repository.

Method	CB		RTE	BoolQ
	Acc.	F1	Acc.	Acc.
BERT + FT	94.6	93.7	70.4	77.7
BERT + CAP	98.2	98.7	75.8	78.1
RoBERTa + FT	96.4	90.4	85.6	85.4
RoBERTa + CAP	100	100	87.4	86.1
DeBERTaV3 + FT	94.7	89.0	88.4	87.5
DeBERTaV3 + CAP	100	100	88.8	88.2

Table 3: Best results on SuperGLUE tasks based on large-scaled model, where BERT + FT results are from (Wang et al., 2019), FT refers to vanilla fine-tuning.

Method	CB		RTE	BoolQ
	Acc.	F1	Acc.	Acc.
Vanilla Fine-tuning	89.3	89.0	70.0	75.2
WARP$_{init}$	91.1	89.2	70.8	76.6
P-tuning + CV	92.9	94.7	69.0	75.6
CAP	100	100	73.6	76.2
+ anchor token	92.9	92.3	68.2	75.0
- CA (i.e. NUP)	92.9	90.5	69.0	75.2
- GDES (i.e. NES)	98.2	96.4	71.1	76.0

Table 4: Dev set results (BERT-base) on SuperGLUE tasks, where WARP$_{init}$ differs from the original paper and trains the entire model, CV refers to continuous verbalizer. - CA denotes CAP without contextualized awareness, i.e. no unique prompt (NUP). GDES means Gradient-Disentangled Embedding Sharing, while NES is No Embedding Sharing.

current prompt-based learning.

In addition, we also do experiments based on large-scale models (Layer=24, Hidden Size=1024, Attention Head=16). Analogous results based on the large-scale models are shown in Table 3.

Furthermore, we compare existing methods based on prompts. To be fair, BERT is applied as the base pre-training model, and the same pattern and verbalizers are used. As shown in Table 4, our method has certain advantages over existing methods.

4.3 Ablation Experiments

To further analyze CAP, a series of ablation experiments are carried out to investigate which part plays a crucial role. Specifically, we evaluate CAP with or without anchor tokens, contextualized awareness and the GDES method.

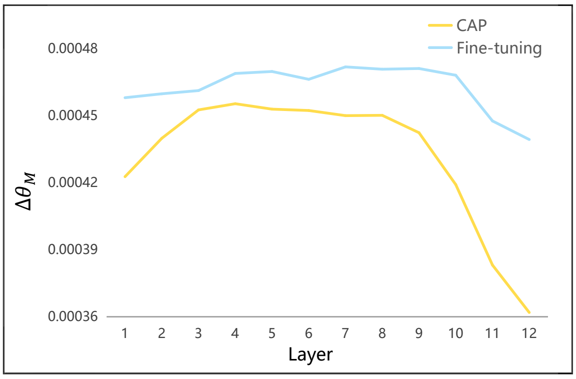


Figure 4: Parameter changes of pre-trained language model (BERT-base) for different layers after fitting downstream task (RTE).

Several conclusions can be drawn from the experiment results in Table 4. First, as predicted by our earlier supposition, the automatic learning of the prompt token using CAP surpasses the addition of the anchor token. Second, prompt combines contextual information and decouples the CAP's contextualized embeddings from the original pre-trained language model word embeddings gradient is effective. Third, creating unique prompts with contextualized awareness plays a key role, with the GDES (Gradient-Disentangled Embedding Sharing) method serving as the icing on the cake.

5 Discussion

The experiment results show that CAP outperforms vanilla fine-tuning in the experimental NLU task. Vanilla fine-tuning is an approach similar to or rather a transfer learning, the inputs are passed through a pre-trained language model (which can be thought of as feature extraction), are then fed into an added linear output layer to predict.

The prompt-based learning approach, on the other hand, employs a pre-trained language model that is more closely aligned with the pre-training process's training objectives. As shown in Figure 4, fitting the same downstream task with CAP changes the parameters less than vanilla fine-tuning. It's not unexpected that CAP achieves better performance than vanilla fine-tuning because we believe the less the pre-trained luaguage model changes, the more it keeps its learnt knowledge. Besides, just as adding a task description such as "Please classify the sentiment of the following sentences" provides a clear grasp of the job's objective, while it is not evident what one needs to do when given a pile of text and must guess at the assignment's intention.

Although prompt-based learning is hard to adapt to all tasks at the moment, it makes sense to explore in the prompt-based learning direction when using pre-trained language models for downstream tasks with "not only fine-tuning" in mind.

6 Conclusion

In this paper, we propose a new method, Context-Aware Prompt (CAP), as an alternative to fine-tuning using pre-trained language models. Specifically, CAP constructs a unique continuous prompt for each diverse input by combining contextual information. Experiment results show that our method can make better and full use of pre-trained language models, thus outperforms vanilla fine-tuning and existing methods for most tasks on the SuperGLUE benchmark. In addition, we further extend to the RTD-based prompts usage scheme, making it possible to use prompt-based learning as a method for the majority of existing pre-trained language models.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *North American Chapter of the Association for Computational Linguistics*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Meeting of the Association for Computational Linguistics*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. In *Meeting of the Association for Computational Linguistics*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzębski, Bruna Halila Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Meeting of the Association for Computational Linguistics*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics*.

Mohammad Taher Pilehvar and José Camacho-Collados. 2018. Wic: 10,000 example pairs for evaluating context-sensitive representations. *CoRR, abs/1808.09121*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *North American Chapter of the Association for Computational Linguistics*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Yi Sun, Yu Zheng, Chao Hao, and Hangping Qiu. 2021. Nsp-bert: A prompt-based zero-shot learner through an original pre-training task–next sentence prediction. *arXiv preprint arXiv:2109.03564*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Neural Information Processing Systems*.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. *arXiv preprint arXiv:2104.05240*.