GPT Understands, Too

Xiao Liu[1*], **Yanan Zheng**[1*], **Zhengxiao Du**[1], **Ming Ding**[1], **Yujie Qian**[2],
Zhilin Yang[1†], **Jie Tang**[1†]
[1]Tsinghua University [2]Massachusetts Institute of Technology

Abstract

Prompting a pretrained language model with natural language patterns has been proved effective for natural language understanding (NLU). However, our preliminary study reveals that manual discrete prompts often lead to unstable performance—e.g., changing a single word in the prompt might result in substantial performance drop. We propose a novel method P-Tuning that employs trainable continuous prompt embeddings in concatenation with discrete prompts. Empirically, P-Tuning not only stabilizes training by minimizing the gap between various discrete prompts, but also improves performance by a sizeable margin on a wide range of NLU tasks including LAMA and SuperGLUE. P-Tuning is generally effective for both frozen and tuned language models, under both the fully-supervised and few-shot settings.

1 Introduction

Pretrained language models (PLMs; Brown et al., 2020) have significantly advanced the performance of natural language understanding (NLU). PLMs are trained with different pretraining objectives, such as masked language modeling (Devlin et al., 2018), autoregressive language modeling (Radford et al., 2019), seq2seq (Raffel et al., 2019), and permutation language modeling (Yang et al., 2019). PLMs can be further enhanced with prompting (Brown et al., 2020; Schick and Schütze, 2020), which employs manually written prompt patterns as additional input to a language model. With prompting while PLMs are either finetuned on a small labeled dataset or frozen for direct inference on downstream tasks. Prompting has significantly improved the performance of many NLU tasks (Brown et al., 2020; Schick and Schütze, 2020).

† corresponding to: Zhilin Yang (zhiliny@tsinghua.edu.cn) and Jie Tang (jietang@tsinghua.edu.cn)
* indicates equal contribution.

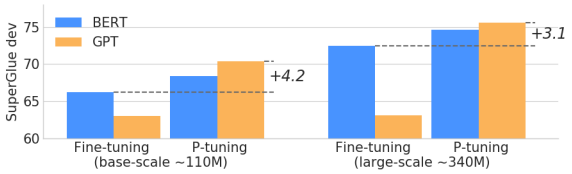


Figure 1: Average scores on 7 dev datasets of SuperGLUE using P-Tuning.

Prompt	P@1 w/o PT	P@1 w/ PT
[X] is located in [Y]. *(original)*	31.3	57.8
[X] is located in which country or state? [Y].	19.8	57.8
[X] is located in which country? [Y].	31.4	58.1
[X] is located in which country? In [Y].	51.1	58.1

Table 1: Discrete prompts suffer from instability (high variance), while P-Tuning stabilizes and improves performance. Results are precision@1 on LAMA-TREx P17 with BERT-base-cased. "PT" refers to P-Tuning, which trains additional continuous prompts in concatenation with discrete prompts.

However, we observe that manual discrete prompts suffer from a large degree of instability. As shown in Table 1, with a frozen language model, changing a single word in the prompt might result in substantial performance drop. As we will show in Section 3, when the language model is tuned, the instability problem is alleviated but the performance difference between different prompts is still sizeable, especially in the few-shot setting. Such an instability issue of discrete prompts poses a critical challenge in practice. Recent approaches of automatic prompting have attempted to search for a better-performing prompt given a task (Shin et al., 2020; Gao et al., 2020; Jiang et al., 2020b), but these methods do not change the unstable nature of discrete prompts.

To reduce the instability of discrete prompts, we propose a novel method P-Tuning that employs trainable continuous prompt embeddings in concatenation with discrete prompts. Specifically,

given a discrete prompt as the input, P-Tuning concatenates continuous prompt embeddings with the discrete prompt tokens and feeds them as the input to the language model. The continuous prompts are updated by backpropagation to optimize the task objective. The intuition is that continuous prompts incorporate a certain degree of learnability into the input, which may learn to offset the effects of minor changes in discrete prompts to improve training stability. To further improve performance, we employ a prompt encoder using LSTMs or MLPs to model the dependency between continuous prompt embeddings.

We experiment with two NLU benchmarks: the LAMA (Petroni et al., 2019) knowledge probing and SuperGLUE (Wang et al., 2019a). On LAMA, with the language model frozen, P-Tuning outperforms manual discrete prompts and searched prompts by 20+ points and 9 points respectively with the same pretrained models. On SuperGLUE, with the language model finetuned, P-Tuning outperforms PET (Schick and Schütze, 2020) with the best discrete prompts under both the fully-supervised and few-shot settings. In addition to improving performance, our results show that across a wide range of tasks and settings, P-Tuning substantially reduces the performance gap between different discrete prompts, which results in improved stability for language model adaptation.

2 Method

2.1 Issues with Discrete Prompts

Prompting employs natural language patterns as additional inputs to pretrained language models for adaptation to downstream tasks (Brown et al., 2020; Schick and Schütze, 2020). Prior work (Zheng et al., 2021) has pointed out that prompting has achieved consistent and substantial improvements on a number of NLP tasks. However, it still remains a challenging problem of how to write high-performing discrete prompts.

We performed preliminary experiments using different manual prompts on the LAMA knowledge probing task (Petroni et al., 2019), which aims to extract triplet knowledge from a language model by predicting the tail entities. Results in Table 1 show that manual discrete prompts lead to unstable performance. For example, if we compare the last two prompts in the table, changing a single word in prompt causes a drastic decrease of 20 points in performance.

In light of the challenge, recent works propose to automate the search procedure of discrete prompts by mining the training corpus (Jiang et al., 2020b), gradient-based searching (Shin et al., 2020), and using pretrained generative models (Gao et al., 2020). However, these works aim at searching for better-performing prompts but do not change the nature of instability for discrete prompts. In addition to the instability issue, searching in the discrete space might not be able to fully leverage the gradients from backpropagation, which will potentially result in suboptimal solutions. To this end, we explore the possibility of training continuous prompts to stabilize and improve the performance of language model adaptation.

2.2 P-Tuning

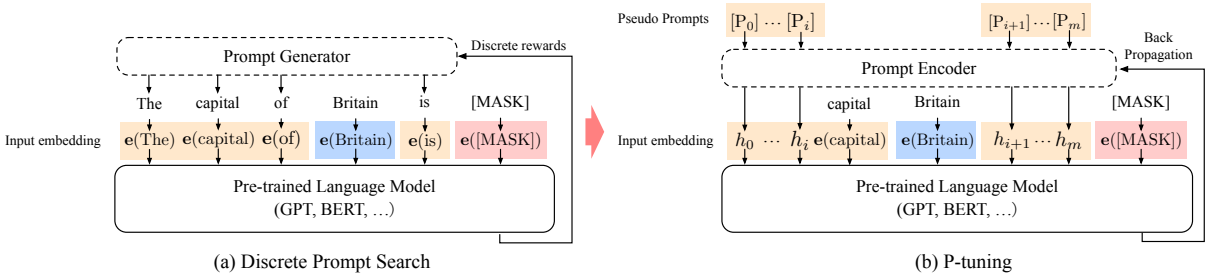
Formally, let $\mathcal{M}$ be a pretrained language model with a hidden size of h and a vocabulary size of $|\mathcal{V}|$. Let $\{(\mathbf{x}_i, \mathbf{y}_i))\}_i$ be a labeled dataset for an NLU task, where $\mathbf{x}_{0:n} = \{x_0, x_1, ..., x_n\}$ is an input consisting of a sequence of discrete tokens, and $\mathbf{y} \in \mathcal{Y}$ is a label. Our goal is to estimate the conditional probability for classification $f_{\mathcal{M}}(x) = \hat{p}(y|x)$ with parameters of $\mathcal{M}$ either finetuned or frozen.

Prompting was proposed in the format of discrete tokens (Schick and Schütze, 2020). Let $[\mathrm{D}_i]$ be a discrete prompt token. Each prompt can be described as a template $T = \{[\mathrm{D}_{0:i}], \mathbf{x}, [\mathrm{D}_{(i+1):j}], \mathbf{y}, [\mathrm{D}_{(j+1):k}]\}$, which could organize the labeled data (including the inputs $\mathbf{x}$ and the label $\mathbf{y}$) into a sequence of text tokens, such that the task could be reformulated as filling in the blanks of the input text. For example, for the task of predicting a country's capital (LAMA-TREx P36), a prompt could be "The capital of [INPUT] is [LABEL]." With a piece of labeled data "(Britain, London)", the reformulated text would be "The capital of Britain is [MASK].", where "[MASK]" should predict the given label "London". Both discrete prompts and discrete data are together mapped into input embeddings:

$$\{\mathbf{e}(D_0)...\mathbf{e}(D_i), \mathbf{e}(x_0), ..., \mathbf{e}(x_n), ..., \mathbf{e}(D_k)\}$$

through the pretrained embedding layer, where $\mathbf{e} \in \mathbb{R}^{|\mathcal{V}| \times d}$.

However, as is discussed in Section 2.1, such discrete prompts tend to be extremely unstable and might not be optimal with back-propagation. Therefore, we propose P-Tuning that uses continuous prompt embeddings to improve and stabilize



(a) Discrete Prompt Search (b) P-tuning

Figure 2: An example of prompt search for "The capital of Britain is [MASK]". Given the context (blue zone, "Britain") and target (red zone, "[MASK]"), the orange zone refer to the prompt. In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the continuous prompt embeddings and prompt encoder can be optimized in a differentiable way.

prompting. Let $[\mathrm{P}_i]$ be the i^{th} continuous prompt embedding. The prompt template for P-Tuning is as follows:

$$T = \{[\mathrm{P}_{0:i}], \mathbf{x}, [\mathrm{P}_{(i+1):j}], \mathbf{y}, [\mathrm{P}_{(j+1):k}]\}$$

P-Tuning leverages an extra embedding function $f : [\mathrm{P}_i] \rightarrow h_i$ to map the template to

$$\{h_0, ..., h_i, \mathbf{e}(\mathbf{x}), h_{i+1}, ..., h_j, \mathbf{e}(\mathbf{y}), h_{j+1}, ..., h_k\}$$

Finally, we update the embeddings $\{P_i\}_{i=1}^{k}$ to optimize a task loss function.

It is noteworthy that we can also concatenate discrete prompts with continuous prompts, which performs better and is adopted throughout our experiments. P-Tuning is applicable to both frozen and finetuned language models.

2.3 Prompt Encoder

In the aforementioned framework, we employ a mapping function f to map trainable embeddings $\{P_i\}$ to model inputs $\{h_i\}$. The intuition is that by using a mapping function, it is more convenient to model the dependency between different prompt embeddings, compared to using independent learnable embeddings. In our implementation, we use a lightweight neural network to formulate the function f. Specifically, we experiment with using long short-term memory (LSTM) networks, multi-layer perceptrons (MLPs), and the identity mapping function in Section 3.

3 Experiments

We include two NLU benchmarks: LAMA (Petroni et al., 2019) for knowledge probing (§ 3.1) and SuperGLUE (Wang et al., 2019a) for general natural language understanding. On SuperGLUE, we consider both the fully-supervised learning (§ 3.2) and few-shot learning (§ 3.3) settings.

	LAMA	Full SG	Few SG
	frozen	tuned	tuned
Improved	✓	✓	✓
Stabilized	✓	✗	✓

Table 2: Task settings and summary of results in our experiments. P-tuning shows improvement over baselines on all task settings, and can stabilize performance on LAMA and Few SG. For Full SG, the gap between discrete prompts is not large and training is stable even without P-Tuning. (Full SG: fully-supervised learning on SuperGLUE; Few SG: few-shot SuperGLUE; Improved: overall performance improved; Stabilized: training stabilized by minimizing difference between discrete prompts).

On LAMA, following Shin et al. (2020); Jiang et al. (2020b), language models are frozen and only the discrete or continious prompts are tuned. For SuperGLUE, following Schick and Schütze (2020); Zheng et al. (2021), language models are tuned. In our setting, we jointly optimize the language model parameters and the continuous prompts. This setup not only follows the common, standard settings in prior work, but also allows evaluating P-Tuning with both tuned and frozen language models.

The overall task setup and a summary of results are shown in Table 2.

3.1 Knowledge Probing

3.1.1 Setup

Knowledge probing, or referred to as fact retrieval, evaluates how much real-world knowledge has language models gained from pre-training. The LAMA (Petroni et al., 2019) dataset evaluates it with cloze tests created from triples selected in the knowledge bases.

Datasets and vocabulary. LAMA enforces all

Prompt type	Model	P@1
Original (MP)	BERT-base	31.1
	BERT-large	32.3
	E-BERT	36.2
Discrete	LPAQA (BERT-base)	34.1
	LPAQA (BERT-large)	39.4
	AutoPrompt (BERT-base)	43.3
P-tuning	BERT-base	48.3
	BERT-large	**50.6**

Model	MP	P-tuning
BERT-base (109M)	31.7	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	45.2
BERT-large (335M)	33.5	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	40.0
RoBERTa-large (355M)	22.1	53.5 (+31.4)
GPT2-medium (345M)	20.3	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	54.4 (+31.6)
MegatronLM (11B)	23.1	**64.2** (+41.1)

Table 3: Knowledge probing Precision@1 on LAMA-34k (left) and LAMA-29k (right). P-tuning outperforms all the discrete prompt searching baselines. (MP: Manual prompt; PT: P-tuning).

answers in single-token format. We first adopt the original LAMA-TREx dataset, consisting of 41 Wikidata relations and altogether 34,039 testing triples (namely LAMA-34k, which covers all BERT vocabularies). Since different pretrained models share distinct vocabularies, to allow direct comparison, we follow previous work (Shin et al., 2020) to adopt a subset that covers the intersection of GPT's and BERT's vocabularies. This is caled LAMA-29k. We again follow Shin et al. (2020) to construct the training, development, and test data to allow for fair comparison.

Setup. LAMA has provided a handcraft prompt for each relation, as shown in Table 1, which are effective but likely sub-optimal. For bidirectional masked language models, we only need to replace "[X]" with the subject entity and "[Y]" with the [MASK] token; for unidirectional language models such as GPT, following LAMA's original setting on Transformer-XL (Dai et al., 2019), we use the network output just before the target position.

The number of prompt tokens and positions are selected based on the development sets, and for simplicity we choose the (3, sub, org_prompt, 3, obj, 3) template for bidirectional models and (3, sub, org_prompt, 3, obj) for unidirectional models as this configuration performs well for most relations (where the number indicates the number of continuous prompt tokens). Continuous prompts are concatenated with original discrete prompts. During the prompt training, we set the learning rate to 1e-5 and use the Adam optimizer.

3.1.2 Main results

The results are presented in Table 3. P-tuning significantly improves the best results of knowledge probing from 43.3% to 50.6% on LAMA-34k and from 45.2% to 64.2% on LAMA-29k. Moreover, P-tuning outperforms previous discrete prompt searching approaches such as AutoPrompt (Shin et al., 2020) and LPAQA (Jiang et al., 2020b) on the same-size models. This confirms our intuition in Section 2 that discrete prompts might not be optimal.

3.2 Fully-supervised Learning

3.2.1 Setup

Dataset. To evaluate P-tuning on fully-supervised learning tasks, we adopt the SuperGLUE benchmark (Wang et al., 2019b), consisting of 8 challenging natural language understanding (NLU) tasks. We focus on 7 of them since the ReCoRD (Zhang et al., 2018) task adopts no discrete prompts, thus P-tuning is not directly applicable. The tasks include question answering (BoolQ (Clark et al., 2019a) & MultiRC (Khashabi et al., 2018)), textual entailment (CB (De Marneffe et al., 2019) & RTE (Dagan et al., 2005)), co-reference resolution (WiC (Pilehvar and Camacho-Collados, 2018)), causal reasoning (COPA (Roemmele et al., 2011)), and word sense disambiguation (WSC (Levesque et al., 2012)).

Comparison methods. We experiment with P-tuning on both unidirectional and bidirectional pretrained models, i.e., GPT and BERT. We include four variants BERT-Base, BERT-Large, GPT2-Base, and GPT-medium. For each model, we compare standard classification finetuning, PET (Schick and Schütze, 2020) (a typical finetuning method based on manual discrete prompts) and our P-tuning.

Configuration. We use the same metrics as in (Wang et al., 2019b). For fully-supervised learning, we use a large training set to finetune pretrained models and use a development set for hyper-

(a) Fully-supervised performance with base-scale models.

	Method	BoolQ (Acc.)	CB (Acc.)	CB (F1)	WiC (Acc.)	RTE (Acc.)	MultiRC (EM)	MultiRC (F1a)	WSC (Acc.)	COPA (Acc.)	Avg.
BERT-Base (109M)	CLS-FT	72.9	85.1	73.9	71.1	68.4	16.2	66.3	63.5	67.0	66.2
	PET-FT	73.7	87.5	90.8	67.9	70.4	13.7	62.5	60.6	70.0	67.1
	P-tuning	73.9	89.2	92.1	68.8	71.1	14.8	63.3	63.5	72.0	68.4
GPT2-Base (117M)	CLS-FT	71.2	78.6	55.8	65.5	67.8	17.4	65.8	63.0	64.4	63.0
	PET-FT	74.8	87.5	88.1	68.0	70.0	23.5	69.7	66.3	78.0	70.2
	P-tuning	75.0	91.1	93.2	68.3	70.8	23.5	69.8	63.5	76.0	70.4

(b) Fully-supervised performance with large-scale models.

	Method	BoolQ (Acc.)	CB (Acc.)	CB (F1)	WiC (Acc.)	RTE (Acc.)	MultiRC (EM)	MultiRC (F1a)	WSC (Acc.)	COPA (Acc.)	Avg.
BERT-Large (335M)	CLS-FT[1]	77.7	94.6	93.7	74.9	75.8	24.7	70.5	68.3	69.0	72.5
	PET-FT	77.2	91.1	93.5	70.5	73.6	17.7	67.0	80.8	75.0	73.1
	P-tuning	77.8	96.4	97.4	72.7	75.5	17.1	65.6	81.7	76.0	74.6
GPT2-Med. (345M)	CLS-FT	71.0	73.2	51.2	65.2	72.2	19.2	65.8	62.5	66.0	63.1
	PET-FT	78.3	96.4	97.4	70.4	72.6	32.1	74.4	73.0	80.0	74.9
	P-tuning	78.9	98.2	98.7	69.4	75.5	29.3	74.2	74.0	81.0	75.6

[1] We report the same results taken from SuperGLUE (Wang et al., 2019a).

Table 4: Fully-supervised performance on SuperGLUE development set.

parameter and model selection. Specifically, the AdamW optimizer with a linearly decayed learning rate is used for training. We use a learning rate of $\{1e-5, 2e-5, 3e-5\}$, a batch size of $\{16, 32\}$, and a warm-up ratio of $\{0.0, 0.05, 0.1\}$. For small datasets (i.e., COPA, WSC, CB, RTE), we fine-tune pretrained models for 20 epochs. For larger datasets (i.e., WiC, BoolQ, MultiRC), we reduce the number of training epochs to be 10 as the model converges earlier. Early stopping is used to avoid over-fitting the training data.

3.2.2 Main Results

The main results of fully-supervised learning are shown in Table 4. We observe that P-tuning can improve fully-supervised learning performance on both BERTs and GPTs. (1) Specifically, on the BERT-Base model, P-tuning achieves best performance on 5/7 tasks, while with BERT-Large, P-tuning outperforms other methods on 4/7 tasks. The exceptions are WiC and MultiRC, both of which have relatively large training sets. We find that P-tuning might not have large gains over CLS-FT on such high-resource tasks, while benefits more on low-resource tasks. On average, P-tuning improves over the considered baselines. (2) On GPT2-Base and GPT2-Medium models, P-tuning consistently achieves the best performance on all tasks.

3.3 Few-Shot Learning

While GPT-3 has shown decent few-shot learning potential with handcrafted prompts, it still struggles on some of the challenging tasks (e.g., natural language inference) (Brown et al., 2020). We are motivated to study whether P-tuning can also improve the few-shot learning performance of pretrained models on challenging tasks.

3.3.1 Setup

Few-shot Evaluation. The few-shot performance is sensitive to lots of factors (e.g., the order of training examples, random seed, and prompt patterns), and thus suffers from high variance (Zhao et al., 2021a; Lu et al., 2021; Zhang et al., 2020). Therefore, the few-shot evaluation strategy should make sure that the improvements are indeed from an improved method instead of variance. To this end, we follow the FewNLU evaluation procedure (Zheng et al., 2021) that has addressed and handled the issue. Specifically, we use random data splits to perform model selection only on a small labeled set to prevent overfitting a large dev set.

Dataset. We use the few-shot SuperGLUE (also known as FewGLUE) benchmark (Schick and Schütze, 2020) and follow the setting in prior work (Zheng et al., 2021) in terms of data split construction.

Baseline and Hyper-parameter. In few-shot learn-

ing, we again compare P-tuning with PET (Schick and Schütze, 2020), which was shown to outperform GPT-3 on some of the tasks. Similar to (Schick and Schütze, 2020), we use ALBERT-xxLarge as the base model. For hyper-parameters that are shared by PET and P-tuning (e.g., learning rate, maximum training step, evaluation frequency), we use the same search space for fair comparison. Specifically, we search the learning rate in $\{1e-5, 2e-5\}$, the maximum training step in $\{250, 500\}$, and the evaluation frequency in $\{0.02, 0.04\}$.

Construction of Prompt Patterns. For PET, we use the same manual prompts reported by Schick and Schütze (2020). When constructing prompt patterns for P-tuning, based on the same manual prompts as PET, we insert different numbers of continuous prompt tokens into different positions, thus formulating a number of pattern candidates. We then select the best pattern for P-tuning using the validation strategy of FewNLU (Zheng et al., 2021). We also conduct further analysis of the number and the position of continuous prompt tokens in §3.3.3.

3.3.2 Main Results

Few-Shot Performance. Table 5 shows the main results of few-shot learning. We find that, on ALBERT, P-tuning consistently outperform PET on average by more than 1 points. It outperforms PromptTuning by more than 13 points. It proves that by automatically learning continuous prompt tokens, the pretrained models can achieve better few-shot performance on NLU tasks.

3.3.3 Ablation Study

Type of Prompt Encoder Prior work (Shin et al., 2020) proposes to simply use an MLP as the prompt encoder, we perform further ablation analysis for prompt encoder selection, and results are shown in Table 8. We consider LSTM, MLP, and EMB (i.e., we directly optimize the word embeddings without using additional parameters). From the results, we can see that LSTM, MLP, and EMB all work as a prompt encoder. Results show that both LSTM and MLP generally work well on these tasks, while EMB is unstable and can substantially under-perform the other two on some tasks (e.g,. WiC and CB). To sum up, both LSTM and MLP could be taken into account when working on new tasks.

Location of Prompt Tokens To study at which location to insert continuous prompt tokens, we perform experiments as Table 7 shows. From the results, we have the following findings.

1. By comparing #1 (or #2) with #3 (or #4), we find that it would be better if we insert continuous prompt tokens at the location where it does not segment the sentences. For example, in case#1, "[P]" breaks the completeness of sentence "[Hypothesis]?" while in case#3, "[P]" is located between sentences.
2. By comparing #2 (or #3) with #4, we find that there's no special preference for placing on the edge or in the middle of the inputs.
3. It is suggested to write a number of pattern candidates and then search over them for the best for each task.

Number of Prompt Tokens We also study the influence of the number of prompt tokens and show the results in Table 7. By comparing #3, #6, #7, and #8, we can conclude that the number of prompt tokens has a great impact on the few-shot performance. However, it is not that a larger number of prompt tokens would always be better. We conjecture that it could be that due to the limited training data, it becomes difficult to learn the parameters when excessively increasing the number of continuous prompt tokens. In practice, it is suggested to search for the best number of prompt tokens through model selection.

3.3.4 Comparison with Discrete Prompt Search

Prior work (Gao et al., 2020) proposed to automatically search discrete prompts and achieved better results than those of manual prompts. We now proceed to compare P-Tuning with auto-searched discrete prompts. For fair comparison, we follow the setting of LM-BFF (Gao et al., 2020) to also conduct experiments on some of the GLUE tasks (Wang et al., 2018) with RoBERTa-Large model (Liu et al., 2019). Since the the evaluation protocols have large impacts on few-shot performance, we use the top-3 discrete prompts searched by LM-BFF and experiment with using only the discrete prompts and additionally applying P-Tuning. For P-Tuning, the prompt patterns are constructed by concatenating the same discrete prompts as well as continuous prompts. Results in Table 9 show that additionally incorporating continuous prompts can further improve few-shot performance. P-Tuning is

Method	BoolQ (Acc.)	RTE (Acc.)	WiC (Acc.)	CB (Acc.)	CB (F1.)	MultiRC (F1a.)	MultiRC (EM.)	WSC (Acc.)	COPA (Acc.)	Avg
Prompt Tuning	58.47 ±1.00	54.42 ±3.05	52.74 ±2.36	75.45 ±2.25	67.73 ±5.70	59.28 ±4.73	15.03 ±4.11	74.04 ±2.99	61.50 ±4.36	58.56
PET-FT	76.70 ±1.85	72.83 ±1.30	53.87 ±4.47	84.38 ±4.47	62.56 ±7.66	76.51 ±1.52	36.46 ±2.13	80.05 ±2.53	81.75 ±4.03	70.74
P-tuning	76.55 ±2.68	63.27 ±3.63	55.49 ±1.21	88.39 ±3.72	84.24 ±5.15	75.91 ±1.74	38.01 ±0.78	78.85 ±1.76	85.25 ±3.30	71.81

Table 5: The few-shot performance of PET (Schick and Schütze, 2020), Prompt Tuning (Lester et al., 2021) and our P-tuning over seven tasks based on ALBERT. Each result is averaged over 4 runs with different data splits. Results show that P-tuning consistently improves average few-shot performance by more than 1 point compared to PET and by more than 13 points compared to Prompt Tuning.

	Method	P#0	P#1	P#2	P#3	P#4	P#5	STD
FSL (BoolQ)	PET-FT	77.10 ±2.21	67.96 ±2.69	74.14 ±1.38	72.48 ±4.31	71.77 ±2.56	60.86 ±3.99	5.68
	P-tuning	75.41 ±3.09	75.11 ±1.61	73.43 ±2.60	71.35 ±4.57	71.31 ±8.58	65.86 ±3.80	3.52
LAMA (P17)	MP	31.3	19.8	31.4	51.1	34.0	32.7	10.1
	P-tuning	57.8	57.8	58.1	58.1	58.9	58.7	0.46

Table 6: Upper table: Few-shot learning (FSL) of PET and P-tuning in terms of each pattern on SuperGLUE with ALBERT; Lower table: Manual prompt (MP) and P-tuning performance on LAMA-P17 with BERT-base-cased. For each column, P-tuning and compared methods share the same manual prompts, while P-tuning additionally concatenates continuous prompt tokens. We report the standard deviation over multiple results of different patterns. Results show that P-tuning achieves smaller standard deviation, proving that P-tuning can improve stability w.r.t. the choice of discrete patterns.

easy to be combined with existing discrete prompts, while further improving stability as discussed in Section 3.4.

3.4 Stabilizing Language Model Adaptation

In the above sections, we have shown that P-Tuning improves over performance across multiple settings. Now we present results to demonstrate that P-Tuning also stabilizes language model adaptation; i.e., reducing the differences between different prompts. As we have shown in Table 1, manual prompts have a large impact on the performance. When it comes to few-shot learning, the performance gap of different prompts is prominent due to the sensitivity of few-shot learning (Zheng et al., 2021). Results in Table 6 show that P-tuning improves the performance of the worst-performing patterns (e.g., P#5), and achieves a smaller standard deviation over multiple patterns. Compared to PET-FT, P-tuning increases the stability w.r.t. the choice of patterns.

On LAMA, we observe similar a phenomenon that while manual prompts often yield quite volatile results, appending trainable continuous prompts on top of the manual prompts can stabilize their performances, reducing the standard deviation from 10.1 to 0.46.

4 Related work

Language Model Prompting. GPT-3 (Brown et al., 2020) uses in-context examples (Liu et al., 2021; Zhao et al., 2021b) as a way of prompting to transfer knowledge from pretraining to downstream tasks. Schick and Schütze (2020) proposed to use cloze patterns, which removes the constraint that the masked token is the last token of the sentence. This further minimizes the gap between pretraining and downstream tasks. To improve prompting for NLU, recent works have proposed methods to automatically search for high-performing prompts by mining the training corpus (Jiang et al., 2020b), gradient-based search (Shin et al., 2020), or using pretrained generative models (Gao et al., 2020). Our approach is different from these prior works in that we resort to using continuous prompt embeddings, which are found to be complementary to discrete prompts in our experiments.

Recently, some concurrent works also proposed the use of continuous prompts. Prefix-tuning (Li and Liang, 2021) adds continuous prompts at the beginning of the sequence for each layer. In contrast to our work, prefix-tuning targets natural language generation tasks.

In the area of NLU, a few concurrent methods were proposed based on continuous prompts, fo-

ID	Prompt Patterns of P-tuning	Seg.	Pos.	#[P]	Acc.	F1.	Avg.
1	[Premise] Question: [Hypothesis] [P] ? Answer: [M].	Yes	Mid	1	87.95	76.70	82.33
2	[Premise] Question [P]: [Hypothesis] ? Answer: [M].	Yes	Mid	1	88.39	78.57	83.48
3	[Premise] Question: [Hypothesis] ? [P] Answer: [M].	No	Mid	1	89.29	79.86	84.58
4	[Premise] [P] Question: [Hypothesis] ? Answer: [M].	No	Miid	1	89.73	82.15	85.94
5	[Premise] Question: [Hypothesis] ? Answer: [M]. [P]	No	Edge	1	87.50	83.39	85.45
6	[Premise] Question: [Hypothesis] ? [P][P] Answer: [M].	No	Mid	2	88.39	84.74	86.57
7	[Premise] Question: [Hypothesis] ? [P][P][P][P] Answer: [M].	No	Mid	4	88.39	85.14	86.76
8	[Premise] Question: [Hypothesis] ? [P][P][P][P][P][P][P][P] Answer: [M].	No	Mid	8	83.48	73.32	78.40

Table 7: The few-shot performance of P-tuning on the CB task on ALBERT with different prompt patterns. "Seg." means whether the inserted prompt tokens segment complete sentences. "Pos." indicates inserting the prompt tokens at the edge or in the middle of the inputs. "[P]" is continuous prompt token. "[M]" is the mask token.

Task	LSTM	MLP	EMB
WiC-ACC	56.27 ±1.54	55.25 ±3.09	53.96 ±3.23
CB-ACC.	81.70 ±7.49	88.39 ±3.72	82.59±3.69
CB-F1.	77.41 ±9.15	84.24±5.15	67.27±6.78
BoolQ-ACC.	75.41±3.09	76.46±2.84	76.87±1.69

Table 8: The few-shot performance on WiC, CB and BoolQ tasks with ALBERT using different prompt encoders. Results show that both LSTM and MLP generally work well on these tasks, while EMB is unstable and can substantially under-perform the other two on some tasks (e.g,. WiC and CB). "EMB" means using an identity mapping for the prompt encoder.

Task	LM-BFF (Auto)	P-Tuning
SST-2	92.89	92.78
MNLI	57.53	58.70
MRPC	68.26	69.49

Table 9: Few-shot performance of automatically searched prompts and P-Tuning. We evaluated LM-BFF (Auto) using the reported top-3 searched patterns under our evaluation procedure. P-Tuning also uses the same discrete prompts, in concatenation with continuous prompts. Results show that P-Tuning can be effectively combined with existing discrete patterns and achieve further performance improvement.

cusing on improving knowledge probing (Qin and Eisner, 2021; Zhong et al., 2021). Lester et al. (2021) showed that with large pretrained models, only tuning continuous prompts with a frozen language model achieves comparable performance to full-model tuning.

Compared to these concurrent works on NLU, P-Tuning reaches a unique conclusion that continuous prompts improve performance and stabilize training with either frozen or tuned models under both the few-shot and fully-supervised settings. For example, no concurrent works have shown that continuous prompts can improve performance with a tuned language model. Technically, P-Tuning also has a few unique designs such as using hybrid continuous-discrete prompts and employing a prompt encoder.

Knowledge in Language Models. Self-supervised (Liu et al., 2020) pre-trained language models (Han et al., 2021) including GPT (Radford et al., 2019), BERT (Devlin et al., 2018), XL-Net (Yang et al., 2019), RoBERTa (Liu et al., 2019) have been observed to learn not only contextualized text representations but also linguistic and world knowledge. (Hewitt and Manning, 2019) demonstrates that contextualized representations produced by language models can form a parse tree in the embedding space. (Vig, 2019; Clark et al., 2019b) look into the multi-head attention patterns within transformers and discover that certain attention heads may correspond to some grammatical functions, including co-reference and noun modifiers. LAMA (Petroni et al., 2019, 2020) propose the LAMA task that leverages cloze tests to predict the fact triples of knowledge bases to examine language model's ability of memorizing facts with answers in the single-token format. In (Wang et al., 2020), the authors investigate the attention matrices to find evidence about knowledge triples contained in the context. (Jiang et al., 2020a) develops a multi-token fact retrieval dataset based on LAMA.

5 Conclusions

In this paper, we present a method P-Tuning that uses continuous prompts in concatenation with discrete prompts. P-Tuning improves performance and stabilizes training for pretrained language model adaptation. P-Tuning is effective with both tuned and frozen language models under both the few-shot and fully-supervised setings.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019b. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

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. *arXiv preprint arXiv:1810.04805*.

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

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. 2021. Pre-trained models: Past, present and future. *AI Open*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. X-factr: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020b. 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.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *ArXiv*, abs/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*. Citeseer.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218*, 1(2).

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*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *CoRR*, abs/2104.08786.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *arXiv preprint arXiv:2005.04611*.

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

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

Guanghui Qin and J. Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *ArXiv*, abs/2104.06599.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 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. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pages 90–95.

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

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *NeurIPS 2019*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ArXiv*, abs/1804.07461.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning. *CoRR*, abs/2006.05987.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021a. Calibrate before use: Improving few-shot performance of language models. *CoRR*, abs/2102.09690.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Yanan Zheng, Jing Zhou, Yujie Qian, Ming Ding, Jian Li, Ruslan Salakhutdinov, Jie Tang, Sebastian Ruder, and Zhilin Yang. 2021. Fewnlu: Benchmarking state-of-the-art methods for few-shot natural language understanding.

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