UNIFIED PROMPT LEARNING MAKES PRE-TRAINED LANGUAGE MODELS BETTER FEW-SHOT LEARNERS

Feihu Jin[1,2], *Jinliang Lu*[1,2], *Jiajun Zhang*[1,2] *

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

ABSTRACT

Language prompting induces the model to produce a textual output during the training phase, which achieves remarkable performance in few-shot learning scenarios. However, current prompt-based methods either use the same task-specific prompts for each instance, losing the particularity of instance-dependent information, or generate an instance-dependent prompt for each instance, lacking shared information about the task. In this paper, we propose an efficient few-shot learning method to dynamically decide the degree to which task-specific and instance-dependent information are incorporated according to different task and instance characteristics, enriching the prompt with task-specific and instance-dependent information. Extensive experiments on a wide range of natural language understanding tasks demonstrate that our approach obtains significant improvements compared to prompt-based fine-tuning baselines in a few-shot setting with about 0.1% parameters tuned. Moreover, our approach outperforms existing state-of-the-art efficient few-shot learning methods on several natural language understanding tasks.

Index Terms— Unified prompt learning, parameter-efficient, fine-tuning, prompt tuning

1. INTRODUCTION

Recently, prompt-based methods for few-shot language model tuning achieve remarkable performance by utilizing a task-specific prompt and a few labeled samples [1] or carefully engineering of prompts and verbalizers to convert inputs to cloze-format [2, 3]. For example, a textual entailment task can be designed by converting the input text **x** to a *prompt pattern* "**x** The answer is [MASK]" where the [MASK] token can be replaced by the *verbalizers* (e.g., 'entailment' and 'not entailment'). However, designing the appropriate prompts is labor-intensive and requires relevant domain knowledge, several efforts attempt to search for discrete prompt tokens automatically [4, 5]. Recent studies [6, 7] have demonstrated that discrete prompt tokens can be sub-optimal, which would make model performance vary from random guessing to state-of-the-art. Therefore, a lot of studies begin to focus on continuous prompts, which mainly includes Prefix Tuning [8] and Prompt tuning [9].

Prefix tuning and Prompt tuning prepend task-specific prompt vectors to the input layer and optimize these vectors during the training stage. The learnable task-specific prompts learn the general information of the task and provide the same task-specific information for each instance. However, these approaches do not take into account the particularity of each instance and focus more on task-specific information. Therefore, task-specific prompts may not be the best option for model predictions. More recently, several works propose the instance-dependent prompt tuning approaches [10, 11, 12]. These learnable instance-dependent prompts are generated based on each input sentence, aiming at finding a proper prompt for each instance. However, these approaches go to an extreme, ignoring the importance of task-specific information for model predictions. In light of these limitations, we believe that a good prompt should reflect both task-specific and instance-dependent information.

In this paper, we propose a **U**nified **P**rompt **L**earning method, named UPL, which incorporates different prompt tuning methods as submodules and learns prompts with both task-specific and instance-dependent information. The approach can efficiently tune large-scale PLMs with as few as 16 end-task examples of each class. First, following the route of prefix tuning [8] for task-specific prompts, we retain the general prompt information of each task. Second, we design a lightweight and low-rank bottleneck architecture to generate instance-dependent prompts for each instance. Then the activation of each submodule in UPL is controlled by *gating mechanism*, which dynamically decides the degree to which task-specific and instance-dependent information are incorporated according to different task characteristics.

We conduct extensive experiments on 13 natural language understanding tasks with RoBERTa-large [13]. Experimental results on a wide variety of NLP tasks demonstrate that our approach can obtain better performance with only a few training instances across all the tasks and only 0.1% parameters of PLMs tuned.

*Corresponding Author

This work is supported by the National Key R&D Program of China under Grants 2022ZD0160602 and the Natural Science Foundation of China under Grants 62122088.

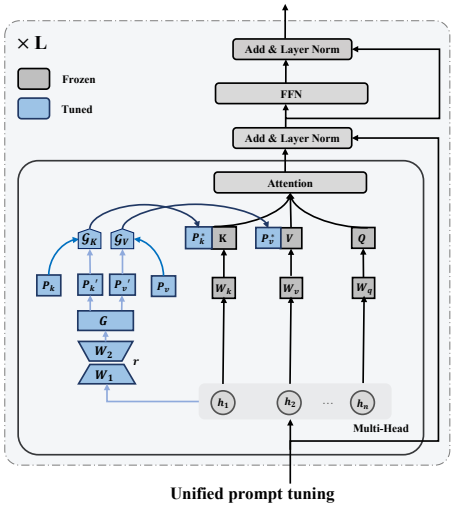


Fig. 1: Illustration of our method. The blue blocks refer to the trainable module, while the gray blocks refer to the frozen module.

2. BACKGROUND

2.1. Task-specific Prompt Tuning

Prefix tuning [8] and P-tuning v2 [14] prepend learnable task-specific vectors $\boldsymbol{P_k}, \boldsymbol{P_v}$ to Key($\boldsymbol{K}$) and Value($\boldsymbol{V}$) of the multi-head attention at each layer of the Transformer:

$$\text{Attn}(\boldsymbol{Q}, concat(\boldsymbol{P_k}, \boldsymbol{K}), concat(\boldsymbol{P_v}, \boldsymbol{V})) \tag{1}$$

Prompt tuning [9] and P-tuning [15] insert learnable task-specific vectors into the input sequence at the embedding layer, which can be formulated as follows:

$$\boldsymbol{E_{in}} = concat(\boldsymbol{W_p}, \boldsymbol{E}(\text{ [SEP] } X \text{ [EOS] })) \tag{2}$$

where $\boldsymbol{E_{in}}$ is the input embeddings, $\boldsymbol{W_p}$ is the embeddings of the inserted prompts, X is the input sentence, and $\boldsymbol{E}$ is the embedding table of the input sentence.

2.2. Instance-dependent Prompt Tuning

Different from the task-specific prompt tuning approaches, the instance-dependent prompt tuning methods aim to generate an instance-dependent prompt for each input sequence [10, 11, 12]. Specifically, given $D_{train} = \{(x_i, y_i)\}_{i=1}^{K}$ of the task T, they suppose that the generation of prompt should correspond to the input sequence x_i. Let $\mathcal{M}$ be the masked language model, and the instance-dependent prompt can be generated as follows:

$$\boldsymbol{W_p}(x_i, T) = \boldsymbol{G}(\mathcal{M}(x_i), T) \tag{3}$$

where $\boldsymbol{G}$ is a generator with a lightweight bottleneck architecture, $\mathcal{M}(x_i)$ is the representation of the input sequence, and the $\boldsymbol{W_p}(x_i, T)$ is the instance-dependent prompt based on each instance of the task T.

3. OUR APPROACH

We propose UPL, a unified and efficient prompt learning method that enriches the prompt with instance-dependent and task-specific information. As shown in Figure 1, the success of UPL consists of three main components: a) the prompts with task-specific information tell the model about the given task; b) a low-rank generator helps the model obtain instance-dependent information; c) a dynamical gating mechanism that can incorporate the task-specific and instance-dependent information according to different task characteristics. We detail the components of our UPL model below.

Let $x_{in} = \{x_1, x_2, \cdots, x_n\}$ be the input sentence of the model $\mathcal{M}$, and let $\boldsymbol{h} = \{\boldsymbol{h_1}, \boldsymbol{h_2}, \cdots, \boldsymbol{h_n}\} \in \mathbb{R}^{n \times d}$ be the hidden states in each layer of the model $\mathcal{M}$, where n is the length of input $\boldsymbol{h}$ and d is the the dimension of the hidden states. We suppose that a good prompt should reflect both task-specific and instance-dependent information. Following the previous works [8, 14], we firstly define two task-specific prompt vectors at each layer of the Transformer: $\boldsymbol{P_k} = \{\boldsymbol{p_{k1}}, \boldsymbol{p_{k2}}, \cdots, \boldsymbol{p_{kt}}\} \in \mathbb{R}^{t \times d}$ and $\boldsymbol{P_v} = \{\boldsymbol{p_{v1}}, \boldsymbol{p_{v2}}, \cdots, \boldsymbol{p_{vt}}\} \in \mathbb{R}^{t \times d}$, where t is the number of tokens in prompt representation and d is the hidden dimension. Different from the previous methods, to reduce the number of trainable parameters, we do not use a reparameterization encoder such as an MLP to generate learnable prompt vectors $\boldsymbol{P}$.

Secondly, we take inspiration from the recent works [19, 20, 18], which show that the learned large-scale models reside on a low intrinsic dimension. Therefore, we design a lightweight and low-rank generator to generate instance-dependent prompt vectors for each input sequence. Specifically, as illustrated in Figure 1, we first project the original d-dimensional sentence representation $\boldsymbol{h}$ into r dimensions through the low-rank matrix $\boldsymbol{W_1}$, then we project the hidden representation back to a d dimensions representation $\hat{\boldsymbol{h}}$ with the low-rank matrix $\boldsymbol{W_2}$, where $\boldsymbol{W_1} \in \mathbb{R}^{d \times r}$, $\boldsymbol{W_2} \in \mathbb{R}^{r \times d}$, and the rank $r \ll d$ (i.e., r in Figure 1 (d) can be two or four).

$$\hat{\boldsymbol{h}} = \boldsymbol{h}\boldsymbol{W_1}\boldsymbol{W_2},\ \hat{\boldsymbol{h}} \in \mathbb{R}^{n \times d} \tag{4}$$

Subsequently, the instance-dependent prompt vectors $\boldsymbol{P'_k} = \{\boldsymbol{p'_{k1}}, \boldsymbol{p'_{k2}}, \cdots, \boldsymbol{p_{kt'}}\} \in \mathbb{R}^{t \times d}$ and $\boldsymbol{P'_v} = \{\boldsymbol{p'_{v1}}, \boldsymbol{p'_{v2}}, \cdots, \boldsymbol{p'_{vt}}\} \in \mathbb{R}^{t \times d}$ can be generated through the linear mapping matrix $\boldsymbol{G} \in \mathbb{R}^{t \times n}$:

$$\begin{aligned} \boldsymbol{P'_k} &= \boldsymbol{G}\hat{\boldsymbol{h}} \\ \boldsymbol{P'_v} &= \boldsymbol{G}\hat{\boldsymbol{h}} \end{aligned} \tag{5}$$

where t is the number of instance-dependent prompt vectors, n is the length of the input, and d is the hidden dimension.

Finally, considering that a good prompt should vary appropriately according to different task characteristics and instance characteristics, we design a dynamical gating mechanism to control the proportion of task-specific and instance-dependent information. As shown in Figure 1, $\boldsymbol{\mathcal{G}_K} \in \mathbb{R}^{t \times t}$ and $\boldsymbol{\mathcal{G}_V} \in \mathbb{R}^{t \times t}$ are two gating matrices that estimate the importance of task-specific and instance-dependent information,

Method	#Params	SST-2 Acc.	SST-5 Acc.	MR Acc.	CR Acc.	MPQA Acc.	Subj Acc.	TREC Acc.	Avg
Single-Sentence									
Prompt-based zero shot [4] †	0%	83.6	35	80.8	79.5	67.6	51.4	32.0	61.4
"GPT-3" in-context learning [4] †	0%	84.8(1.3)	30.6(0.9)	80.5(1.7)	87.4(0.8)	63.8(2.1)	53.6(1.0)	26.2(2.4)	70.0(1.5)
Fine-tuning [13] †	100%	81.4(3.8)	43.9(2.0)	76.9(5.9)	75.8(3.2)	72.0(3.8)	90.8(1.8)	88.8(2.1)	75.7(3.2)
LM-BFF [4] †	100%	92.3(1.0)	49.2(1.6)	85.5(2.8)	89.0(1.4)	**85.8(1.9)**	**91.2(1.1)**	88.2(2.0)	83.0(1.7)
Adapter tuning [16]	3%	92.2(1.8)	50.4(3.2)	87.7(2.6)	90.4(3.9)	73.8(5.8)	90.8(1.6)	88.6(6.4)	82.0(3.6)
Bitfit [17]	0.09%	92.8(1.6)	48.6(3.4)	87.7(3.7)	90.5(1.2)	64.3(9.4)	88.8(3.8)	85.7(10.3)	79.8(4.8)
LoRA [18]	0.1%	**93.0(2.4)**	49.2(2.3)	87.2(2.0)	90.5(2.7)	68.7(7.8)	90.1(2.4)	88.8(5.8)	81.1(3.6)
Prefix tuning [8]	0.2%-1%	91.9(1.5)	49.6(2.0)	87.4(2.3)	90.6(2.2)	74.2(3.1)	88.4(2.3)	88.5(3.7)	81.5(2.4)
IDPT [12]	0.08%	91.5(1.6)	49.0(2.5)	86.4(3.2)	90.9(2.0)	67.3(7.4)	89.7(1.2)	86.4(3.4)	80.2(3.0)
Fixed-UPL	0.1%	92.5(1.2)	**50.8(1.9)**	87.9(2.2)	91.6(1.8)	76.4(4.0)	89.3(1.7)	89.6(3.0)	82.6(2.3)
Dynamic-UPL	0.12%	92.6(1.4)	50.6(1.3)	**88.3(1.6)**	**91.6(1.4)**	78.1(2.3)	90.5(1.4)	**90.4(3.6)**	**83.2(1.9)**

Method	#Params	MNLI Acc.	MNLI-mm Acc.	SNLI Acc.	QNLI Acc.	RTE Acc.	MRPC F1.	QQP F1.	Avg
Sentence-Pair									
Prompt-based zero shot [4] †	0%	50.8	51.7	49.5	50.8	51.3	61.9	49.7	52.2
"GPT-3" in-context learning [4] †	0%	52.0(0.7)	53.4(0.6)	47.1(0.6)	53.8(0.4)	60.4(1.4)	45.7(6.0)	36.1(5.2)	49.8(2.1)
Fine-tuning [13] †	100%	45.8(6.4)	47.8(6.8)	48.4(4.8)	60.2(6.5)	54.4(3.9)	76.6(2.5)	60.7(4.3)	56.3(5.0)
LM-BFF [4] †	100%	68.3(2.5)	**70.1(2.6)**	**77.1(2.1)**	68.3(7.4)	**73.9(2.2)**	76.2(2.3)	67.0(3.0)	71.6(3.2)
Adapter tuning [16]	3%	67.6(4.3)	67.5(4.1)	74.0(4.6)	64.3(4.9)	68.8(7.4)	81.8(2.9)	67.5(4.6)	70.2(4.7)
Bitfit [17]	0.09%	67.2(5.7)	67.6(4.6)	74.4(3.7)	64.7(3.7)	66.6(10.7)	76.6(10.4)	66.8(3.7)	69.1(6.1)
LoRA [18]	0.1%	68.3(4.2)	67.4(2.7)	75.9(2.1)	**68.6(3.9)**	69.6(6.7)	81.0(4.8)	**68.1(2.6)**	71.3(3.3)
Prefix tuning [8]	0.2%-1%	68.1(3.5)	68.3(2.4)	75.4(1.8)	67.5(3.5)	70.2(5.1)	81.4(2.8)	66.8(6.5)	71.1(3.7)
IDPT [12]	0.08%	66.3(3.6)	66.0(2.1)	73.5(2.3)	66.6(3.7)	70.3(4.3)	80.6(3.4)	66.4(4.6)	70.0(3.4)
Fixed-UPL	0.1%	68.9(1.5)	68.7(1.9)	76.2(1.1)	68.2(1.4)	71.2(5.3)	81.8(1.3)	66.1(2.8)	71.6(2.2)
Dynamic-UPL	0.12%	**69.3(2.4)**	68.8(2.2)	75.9(2.1)	68.3(2.4)	72.8(3.9)	**82.1(2.1)**	66.6(3.2)	**72.0(2.6)**

Table 1: Performance of all methods on RoBERTa-large. † indicates the results in [4]. We report average(and standard deviation) performance over 5 different splits. Bold fonts indicate the best results.

where t is the length of prompt vectors. We apply a sigmoid function σ to obtain the contribution scores for task-specific and instance-dependent prompts.

$$\begin{aligned} \boldsymbol{P_k^\star} &= \sigma(\boldsymbol{\mathcal{G}_K P_k}) \cdot \boldsymbol{P_k} + \sigma(\boldsymbol{\mathcal{G}_K P_k'}) \cdot \boldsymbol{P_k'} \\ \boldsymbol{P_v^\star} &= \sigma(\boldsymbol{\mathcal{G}_V P_v}) \cdot \boldsymbol{P_v} + \sigma(\boldsymbol{\mathcal{G}_V P_v'}) \cdot \boldsymbol{P_v'} \end{aligned} \tag{6}$$

We then concatenate the prompt vectors $\boldsymbol{P_k^\star}$ and $\boldsymbol{P_v^\star}$ that contain task-specific and instance-dependent information with the original key $\boldsymbol{K}$ and value $\boldsymbol{V}$:

$$\text{Attn}(\boldsymbol{Q}, concat(\boldsymbol{P_k^\star}, \boldsymbol{K}), concat(\boldsymbol{P_v^\star}, \boldsymbol{V})) \tag{7}$$

4. EXPERIMENTAL RESULTS

4.1. Experimental Settings

We conduct our experiments with the same setting following LM-BFF [4] and DART [21], which measure the average performance of models trained on 5 different randomly sampled D_{train} and D_{dev} splits. We train the model for 100 epochs for each split and take the best checkpoint as measured on D_{dev}. We report the accuracy and F1-score for 13 NLU tasks and use a default setting training for a batch size of 8, a learning rate of 4e-4, and a prompt length of 16. We set the rank r of the low-rank matrix to 4. We train our proposed UPL on one NVIDIA A100 with 80G of memory. We explore two versions of unified prompt learning methods: Fixed-UPL, and Dynamic-UPL. The first version uses a fixed hyperparameter $\alpha = 0.3$ (e.g.,$\alpha \in (0, 1)$) to control the proportion of task-specific and instance-dependent information. The second one uses learnable gating matrices that can incorporate task-specific and instance-dependent information dynamically.

4.2. Main Results

Table 1 shows the results of all methods on RoBERTa-large [13] with 5 different randomly sampled splits (e.g., each split contains 16 examples of each class) across 13 NLU tasks. UPL outperforms all other methods and achieves new state-of-the-art results for few-shot learning on several NLU tasks.

First, our proposed method Dynamic-UPL improves the performance compared to Prefix tuning [8] by 1.7 and 0.9 points for single-sentence and sentence-pair datasets respectively with fewer model parameters tuned. Compared to IDPT [10, 12], Dynamic-UPL improves the performance by 3.0 and 2.0 points for single-sentence and sentence-pair datasets respectively. Dynamic-UPL performs better than Prefix tuning and IDPT across 12 and 13 datasets respectively (e.g., especially on RTE, TREC, and SST-5), demonstrating that incorporating task-specific and instance-dependent prompt tuning methods as submodules is advantageous.

Second, compared with the currently prevailing parameter-efficient tuning methods, Dynamic-UPL obtains state-of-the-art results. Our method Dynamic-UPL performs better than Adapter [16], LoRA [18], and BitFit [17] across 11 tasks, 10 tasks, and 12 tasks respectively. Moreover, we report the percentage of trained parameters for parameter-efficient tuning methods in Table 1. Dynamic-UPL requires fewer parameters and achieves a better performance than Adapter tuning. For LoRA, BitFit, and Dynamic-UPL, only about 0.1% parameters are tuned, while Dynamic-UPL learns the task-specific and instance-dependent information and yields better performance.

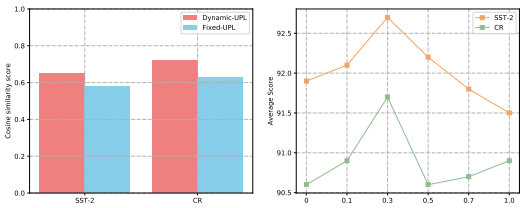
Third, compared with LM-BFF [4], a prompt-based fine-tuning method with all parameters tuned, Dynamic-UPL achieves comparable results with LM-BFF on average and even outperforms LM-BFF across 7 datasets with only about 0.1% parameters tuned.

Finally, these promising results illustrate that Dynamic-UPL has better generalization in few-shot settings and makes the pre-trained language model a better efficient few-shot learner. As an ablation, when learning prompts with a hyperparameter α to control the proportion of task-specific and instance-dependent information in Fixed-UPL, we observe that the performance consistently lags behind the Dynamic-UPL but performs better than other methods (except on LM-BFF), showing that learn prompts with both task-specific and instance-dependent information is a good alternative for few-shot learning.

5. ANALYSIS AND DISCUSSION

5.1. Dynamic-UPL or Fixed-UPL?

As shown in Figure 2(a), we calculate the cosine similarity scores of task-specific prompts and instance-dependent prompts under both Dynamic-UPL and Fixed-UPL. The Dynamic-UPL obtains high scores than Fixed-UPL on both SST-2 and CR, showing that incorporating task-specific and instance-dependent information dynamically can further enhance the degree of information integration and yield better performance. To further analyze how Fixed-UPL influences the fusion of task-specific and instance-dependent information, we control the degree of incorporating task-specific and instance-dependent information through the proportion of hyperparameter α. As shown in Figure 2(b), for the two datasets SST-2 and CR, Fixed-UPL performs better than task-specific prompt tuning and instance-dependent prompt tuning when the hyperparameter α is between 0.1 and 0.4, showing that Fixed-UPL does need both the task-specific and instance-dependent information.



(a) (b)

Fig. 2: (a) The cosine similarity scores of task-specific prompts and instance-dependent prompts under both Dynamic-UPL and Fixed-UPL. (b) We utilize the hyperparameter α to control the degree of incorporating task-specific and instance-dependent information.

Dataset	r=2	r=4	r=8	r=16	r=64
SST-5	49.8	50.3	50.5	49.9	50.0
CR	90.8	91.2	91.4	91.2	91.1

Table 2: Validation accuracy on SST-5 and CR with different rank r. We conduct the experiments with the hyperparameter $\alpha = 0.3$.

5.2. Impact of the Rank r

We utilize a low-rank matrix to generate the instance-dependent prompt vectors. To explore the impact of low rank r on the instance-dependent prompt generation, in Table 2, we show the accuracy of SST-5 and CR with different rank r. Surprisingly, Dynamic-UPL performs competitively with a very small rank r, which suggests that a low-rank generation matrix is sufficient.

6. CONCLUSION AND FUTURE WORK

This paper presents UPL, a simple and efficient method that improves few-shot learning with pre-trained language models. Our approach enriches the prompt with task-specific and instance-dependent information by dynamically controlling the proportion of the two types of prompts. Through extensive experiments over 13 NLP tasks, we demonstrate that UPL generalizes better than the existing state-of-the-art efficient methods in few-shot learning scenarios. Intuitively, the simplicity and effectiveness of UPL proposed in this study makes it a promising method and can stimulate future research directions in the few shot with PLMs.

7. REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, "Language models are few-shot learners," in *NeurIPS*, 2020.

[2] Timo Schick and Hinrich Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *EACL*, 2021.

[3] Timo Schick and Hinrich Schütze, "It's not just size that matters: Small language models are also few-shot learners," in *NAACL-HLT*, 2021.

[4] Tianyu Gao, Adam Fisch, and Danqi Chen, "Making pre-trained language models better few-shot learners," in *ACL*, 2021.

[5] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh, "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts," in *EMNLP*, 2020.

[6] Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh, "Calibrate before use: Improving few-shot performance of language models," in *ICML*, 2021.

[7] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp, "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity," in *ACL*, 2022.

[8] Xiang Lisa Li and Percy Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *ACL*, 2021.

[9] Brian Lester, Rami Al-Rfou, and Noah Constant, "The power of scale for parameter-efficient prompt tuning," in *EMNLP*, 2021.

[10] Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong, "Instance-aware prompt learning for language understanding and generation," *CoRR*, vol. abs/2201.07126, 2022.

[11] Xiaodong Gu, Kang Min Yoo, and Sang-Woo Lee, "Response generation with context-aware prompt learning," *CoRR*, vol. abs/2111.02643, 2021.

[12] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, V. G. Vinod Vydiswaran, and Hao Ma, "IDPG: an instance-dependent prompt generation method," *CoRR*, vol. abs/2204.04497, 2022.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, vol. abs/1907.11692, 2019.

[14] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *CoRR*, vol. abs/2110.07602, 2021.

[15] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang, "Gpt understands, too," *ArXiv*, vol. abs/2103.10385, 2021.

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly, "Parameter-efficient transfer learning for nlp," in *ICML*, 2019.

[17] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *ACL*, 2022.

[18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen, "Lora: Low-rank adaptation of large language models," *ArXiv*, vol. abs/2106.09685, 2021.

[19] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *ACL*, 2021.

[20] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski, "Measuring the intrinsic dimension of objective landscapes," *ArXiv*, vol. abs/1804.08838, 2018.

[21] Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen, "Differentiable prompt makes pre-trained language models better few-shot learners," *ArXiv*, vol. abs/2108.13161, 2021.