Prompt-based pre-trained language models (PLMs) paradigm has succeeded substantially in few-shot natural language processing (NLP) tasks. However, prior discrete prompt optimization methods require expert knowledge to design the base prompt set and identify high-quality prompts, which is costly, inefficient, and subjective. Meanwhile, existing continuous prompt optimization methods improve the performance by learning the ideal prompts through the gradient information of PLMs, whose high computational cost, and low readability and generalizability are often concerning. To address the research gap, we propose a Dialogue-comprised Policy-gradient-based Discrete Prompt Optimization (DP2O) method. We first design a multi-round dialogue alignment strategy for readability prompt set generation based on GPT-4. Furthermore, we propose an efficient prompt screening metric to identify high-quality prompts with linear complexity. Finally, we construct a reinforcement learning (RL) framework based on policy gradients to match the prompts to inputs optimally. By training a policy network with only $0.62M$ parameters on the tasks in the few-shot setting, DP2O outperforms the state-of-the-art (SOTA) method by $1.52\%$ in accuracy on average on four open-source datasets. Moreover, subsequent experiments also demonstrate that DP2O has good universality, robustness and generalization ability.

**Abstract**

With the continuous development of pre-trained language models (PLMs) (Liu et al. 2019; Touvron et al. 2023; Anil et al. 2023), e.g., ChatGPT (OpenAI 2022) and GPT-4 (OpenAI 2023), prompt-based methods have shown significant rising competitiveness in few-shot downstream tasks (Schick and Schütze 2020a,b). Unlike the traditional fine-tuning methods, which require the design of additional neural network heads according to downstream tasks, the prompt-based methods join particular extra texts to inputs to transfer downstream tasks into mask-filling tasks. The prompt matches the downstream task with the model’s pre-training task, and the potential of the PLMs can be more comprehensively scheduled. However, PLMs are extremely sensitive to prompts (Holzman et al. 2019; Lester, Al-Rfou, and Constant 2021). Minor gaps with the same semantics in prompts may lead PLMs to completely different performances. Therefore, one core issue of the prompt-based methods is finding high-quality prompts to promote the performance of PLMs.

Currently, prompt optimization methods can be divided into two categories: discrete prompt optimization and continuous prompt optimization.

Due to the discrete nature of the text, prompts can not be directly optimized by using the gradient information from PLMs. Therefore, previous discrete prompt optimization methods heavily relied on the manually designed basic prompt sets and prompt templates (Jiang et al. 2020; Yuan, Neubig, and Liu 2021; Haviv, Berant, and Globerson 2021; Davison, Feldman, and Rush 2019). Moreover, lacking clear evaluation metrics, prior works often use the supervision...
gain of training-set prompt as a screening metric during optimization (Zhou et al. 2022; Gao, Fisch, and Chen 2020). Therefore, discrete prompt optimization methods usually necessitate a number of labeled data, which contradicts the few- or zero-shot learning objectives, and overlooks the potential impact of prompts on the output distribution, and the further effect on the performance of PLMs.

Meanwhile, the continuous prompt optimization methods abandon the text structure of prompts and improve the performance of PLMs by directly optimizing token embedding at specific locations (Vu et al. 2021; Li and Liang 2021; An et al. 2022; Qian et al. 2022). Although these methods can directly use gradient to guide the optimization direction of continuous prompts, the computational cost is often exceedingly expensive. Besides, continuous prompts usually lack readability and are hardly used across different PLMs.

Towards these challenges, we propose a Dialogue-comprised Policy-gradient-based Discrete Prompt Optimization method, named DP$_2$O. As shown in Figure 1, DP$_2$O mainly consists of two stages: prompt set construction and prompt matching. In the prompt set construction stage, we propose a prompt set generation method with a multi-round dialogue alignment strategy by employing the dialogue characteristics of GPT-4, one of the current most capable PLMs on dialogue. Meanwhile, we introduce an innovative prompt quality assessment metric, i.e., Supervised & Unsupervised Entropy Metric (SUE), which considers the supervised and unsupervised impact of prompts on PLMs with linear complexity and facilitates output distribution balance and accuracy in downstream tasks. In the prompt matching stage, we propose a reinforcement learning (RL) framework, which employs a policy network to select appropriate input prompts. The prompts, without breaking textual semantics, ensure their readability and transferability across different PLMs. Extensive experiments show DP$_2$O is significantly superior to baseline and SOTA methods, e.g., DP$_2$O achieve an average improvement of 1.52% in accuracy across four public datasets with only a 10.86% training time of the SOTA method RL-Prompt (Deng et al. 2022). Furthermore, we implement ablation and analysis experiments to demonstrate the effectiveness, robustness and generalization of DP$_2$O.

In summary, our contributions are summarized as follows:

- **Novel Generation Strategy**: We generate the prompt set via the multi-round dialogue alignment strategy, aiming at reducing the cost of human involvement in prompting.
- **Linear Evaluation Metric**: We additionally consider unsupervised information of PLMs prediction in prompts evaluation, proposing a new metric to screen out excellent prompts with linear complexity.
- **Precise Prompt Matching**: We apply RL techniques to achieve sample-level discrete prompt optimization, further improving the performance of PLMs on downstream tasks.
- **Outstanding Task Performance**: Experiments on four public datasets show that DP$_2$O effectively improves the performance of PLMs under few-shot settings with readability, robustness, generalization, and universality.

### Methodology

The main workflow of DP$_2$O can be mainly divided into two stages: prompt set construction and prompt matching, as shown in Figure 2.

#### Prompt Set Construction Stage

**Evaluation Metric.** Most prevailing methods utilize the aggregate accuracy of the prompt on the dataset as their sole metric for assessment, which neglects the impact of the distribution of labels in the dataset. Lu et al. (2021) find that significantly imbalanced prediction distributions typically characterize underperforming prompts. To this end, we introduce a novel evaluation metric termed Supervised & Unsupervised Entropy metric (SUE). SUE aims to provide a more comprehensive appraisal for prompts by additionally considering global balance beyond local accuracy.

SUE consists of two parts: **supervision score** $S_{\text{sup}}$ and unsupervised score $S_{\text{uns}}$. Given a prompt $x$ and input set $Z$, for each input $z_i \in Z$, we first calculate the difference of the probability $p_{\text{LM}}(z_i)$ that the $z_i$ is correctly labeled $c_i$ and wrongly labeled as $c_{\text{else}}$ by a base PLM. Here $c_{\text{else}} \in C \setminus \{c_i\}$ exactly, and $C$ is the label space of the input. Then the super-entropy score $S_{\text{sup}}$ of prompt $x$ is defined as:

$$S_{\text{sup}}(x, Z) = \sum_{z_i \in Z} (p_{\text{LM}}(c_i|x, z_i) - p_{\text{LM}}(c_{\text{else}}|x, z_i))$$ (1)

To prevent some prompts from causing PLMs to be overly biased on all inputs, SUE selects the prompts which guide PLMs to output a more balanced pseudo-label distribution across all given inputs. Given a prompt $x$, we calculate an entropy value $\mathbb{H}(\cdot)$ of each input $z_i$, then add $\mathbb{H}(\cdot)$ of each input as $S_{\text{uns}}$ for the whole input set $Z$:

$$\mathbb{H}(x, z_i) = -\sum_{c_i \in C} p_{\text{LM}}(c_i|x, z_i) \log p_{\text{LM}}(c_i|x, z_i)$$ (2)

$$S_{\text{uns}}(x, Z) = \sum_{z_i \in Z} \mathbb{H}(x, z_i)$$ (3)

Finally, we have our evaluation metric SUE to assess the quality of prompts as

$$SUE(x, Z) = \lambda_1 S_{\text{sup}}(x, Z) + \lambda_2 S_{\text{uns}}(x, Z)$$ (4)

where $\lambda_1$ and $\lambda_2$ are the weights to balance the supervised score $S_{\text{sup}}$ and the unsupervised score $S_{\text{uns}}$. For the input set $Z$ encompassing multiple inputs, the metric SUE characterizes the holistic quality of the prompt. Higher SUE represents the better capability on the specific downstream task (derived from $S_{\text{sup}}$) and more benign confidence on all inputs (derived from $S_{\text{uns}}$). Meanwhile, when $Z$ only comprises a single input, SUE can elucidate the degree of match between the prompt and the input.

**Prompt Set Generation.** Existing discrete prompt optimization methods, such as Black-Box Tuning (Sun et al. 2022) and GrIPS (Prasad et al. 2022), mostly require text editing based on manually designed prompts and vocabularies. Different from these methods, DP$_2$O leverages GPT-4 as a dialogue model to generate pseudo-label inputs which approximate the dataset distribution, utilizing only a limited
set of training data. Then the inputs are used as prompt examples for downstream tasks. Notably, Min et al. (2022) indicate that the label authenticity of these pseudo-label prompt examples has little impact on the performance of PLMs. Hence, for DP$_2$O, we do not validate the authenticity of the labels of the inputs generated by GPT-4, which further eliminates the necessity for human annotation. Our experiments also show that, without verifying the authenticity of labels, DP$_2$O can still achieve praiseworthy and competitive performance to other methods.

Algorithm 1: Prompt Set Construction of DP$_2$O

**Input:** Few-shot training set $Z_{\text{train}}$ including inputs and labels, label space $C$, base PLM and access to GPT-4 API.

1. $Z_{\text{seed}} \leftarrow$ top-$m$ inputs in $Z_{\text{train}}$ via SUE($z_i, Z_{\text{train}}$), $z_i \in Z_{\text{train}}$.

   **** inner-loop begins: multi-round dialogue ****

2. round $\leftarrow 0$
3. while round $< round_{\text{max}}$ do
4. Random shuffle $Z_{\text{seed}}$.
5. Input $Z_{\text{seed}}[0 : 1]$ to GPT-4 with prefix introduction of task.
6. GPT-4 output $n$ pseudo-labeled inputs $\{z_{p1}^i, \ldots, z_{pn}^i\}$.

   **** inner-loop begins: one dialogue round ****

7. Initialize number of used inputs in $Z_{\text{seed}}$ num $\leftarrow 2$.
8. while num $< m$ do
9. Input $Z_{\text{seed}}[\text{num}]$ to GPT-4, asking it rewrite the previous $\{z_{p1}^{\text{num}-1}, \ldots, z_{pn}^{\text{num}-1}\}$.
10. GPT-4 output $\{z_{p1}^{\text{num}}, \ldots, z_{pn}^{\text{num}}\}$.
11. num $\leftarrow$ num $+ 1$
12. end while
13. Append $\{z_{p1}^{\text{num}-1}, \ldots, z_{pn}^{\text{num}-1}\}$ to $Z_{\text{cand}}$.
14. end while
15. $X \leftarrow$ top-$h$ inputs of $Z_{\text{cand}}$ via SUE($z_i, Z_{\text{train}}$), $z_i \in Z_{\text{train}}$.

**Output:** Readable and high-quality prompt set $X$.

With the prevalence of PLMs aiming at chatting, dialogue is an effective way to input multi-inputs to models. Instead of concatenating them into long sequence text, dialogue strategy can ease the forgetness of PLMs caused by the sliding window. As shown in Algorithm 1, we utilize dialogue to gradually align our prompts with the distribution of PLM to reduce the potential threat of biased prediction.

Given a training set denoted as $Z_{\text{train}}$, we first individually evaluate each input $z_i \in Z_{\text{train}}$ by score SUE($z_i, Z_{\text{train}}$). SUE signifies the input $z_i$’s efficacy within the given set $Z_{\text{train}}$. We rank $Z_{\text{train}}$ set in descending order based on the SUE($z_i, Z_{\text{train}}$). We then select the top-$m$ inputs to form the seed set $Z_{\text{seed}} = \{z_{\text{seed}1}, z_{\text{seed}2}, \ldots, z_{\text{seed}m}\}$.

Subsequently, utilizing GPT-4, we generate pseudo-label inputs that mirror the distribution of the prompts within the $Z_{\text{seed}}$. Initially, GPT-4 randomly takes inputs from $Z_{\text{seed}}$ and then begins a dialogue round (outer-loop).

In one dialogue round, we first generate $n$ pseudo-labeled inputs $\{z_{p1}^i, \ldots, z_{pn}^i\}$ based on any two inputs from $Z_{\text{seed}}$. Then we randomly select one of the remaining inputs from $Z_{\text{seed}}$ into dialogue to guide GPT-4 to polish the previously generated pseudo-labeled inputs $\{z_{p1}^i, \ldots, z_{pn}^i\}$ and get corresponding $\{z_{p1}^{i+1}, \ldots, z_{pn}^{i+1}\}$. Repeated the polishing stage (inner-loop) $m - 2$ times until all $z_{\text{seed}}$ inputted to the dialogue once. Then we get $\{z_{p1}^{m-1}, \ldots, z_{pn}^{m-1}\}$ and append them in candidate set $Z_{\text{cand}}$.

However, the order of conversation might impact dialogue alignment. We suggest re-ordering the conversation multi-time to counteract the effect of the order. Thus, we shuffle $\{z_{\text{seed}1}, z_{\text{seed}2}, \ldots, z_{\text{seed}m}\}$ and start a new dialogue round (outer-loop), aiming at minimizing the impact of order in dialogue. After finishing all $round_{\text{max}}$ times dialogue rounds (outer-loop), we compute the score SUE of all $n \times round_{\text{max}}$ inputs in $Z_{\text{cand}}$. Then, we select the top-$h$ inputs from the candidate set $Z_{\text{cand}}$ as the final prompt set $X$.
Overall, we introduce the multi-round dialogue alignment strategy, optimizing GPT-4’s utility in prompt generation and leveraging its inherent dialogue characteristic.

Prompt Matching Stage

Previous studies have underscored the high sensitivity of PLMs to prompts (Radford et al. 2018; Dathathri et al. 2019; Raffel et al. 2020). Traditional methods tend to rely on either random selection or a simple cosine similarity metric between the input and prompt embedding for selection (Gao, Fisch, and Chen 2020). This leads to an under-exploration of the prompt space limiting the potential performance enhancements in complex tasks. While for the recently emerging RL-based methods, the complexity associated with brute-force searching escalates exponentially with data size growth. Hence, efficiently matching appropriate prompts for each input is a highly challenging task.

Algorithm 2: Prompt Matching of DP₂O

Input: Training set $Z_{train}$ of size $T$, testing set $Z_{test}$, base PLM, the prompt set $X$ constructed by Algorithm 1.

**** training the RL model ****

1: Initialize the policy network $\pi_\theta$ parameters $\theta$ and $epoch \leftarrow 0$.
2: while epoch $< epoch_{\text{max}}$ do
3:   for step $t$ in $[1, \ldots, T]$ do
4:      Get state $s_t \leftarrow \text{PLM}(z_t)$.
5:      Run policy network $\pi_\theta(a_t|s_t)$ to take an action $a_t$ to select a prompt $x_t$.
6:      Calculate reward by SUE, i.e., $r_t \leftarrow \text{SUE}(x_t, z_t)$.
7:      Add transition to replay buffer.
8:   end for
9:   Update parameters $\theta$ of $\pi_\theta$ with the policy gradient loss.
10: end while

**** testing phase begins ****

11: for each input $z$ in $Z_{test}$ do
12:    Get state $s \leftarrow \text{PLM}(z)$.
13:    Get final prediction according to Eq. 5.
14: end for

Output: A trained policy network $\pi_\theta$, predictions for test inputs.

Model Overview. In response to these challenges, we define the discrete prompt matching problem as a Reinforcement Learning (RL) problem, Markov Decision Process (MDP), as shown in Algorithm 2. For the action space $A$ of the RL agent, an action $a_t$ denotes that the agent selects a prompt $x_t$ from the prompt set $X$ obtained in the prompt set construction stage.

At each step $t$ of the training phase, given a state $s_t = \text{PLM}(z_t)$, i.e., the last hidden layer embedding of input $z_t$, the RL agent takes an action $a_t$ of selecting a prompt $x_t$ from the prompt set $X$ according to the policy $\pi_\theta(a_t|s_t)$ where $\theta$ is the learnable parameter of the policy network. We concatenate $x_t$ with $z_t$ and input them into PLM to complete downstream tasks, and calculate the reward $r_t$ of the RL agent based on the output of the PLM. The goal of the RL agent is to maximize the expected reward $R = E(\sum_{t=1}^{T} \gamma^t r_t)$, where $\gamma^t$ is the discount factor at step $t$.

In the testing phase, we adopt the ensemble decision-making approach for prompt selection. The prompts with top-$k$ probability values are then entered into PLM to perform downstream tasks, which are weighted by the probability from the policy network $\pi_\theta$. Given an input $z$ and its corresponding state $s$, the final prediction obtained by DP₂O at label $c$ can be expressed as

$$P(c|z) = \text{softmax}(\sum_{j=1}^{k} \pi_\theta(a_j|s) \log(\text{PLM}(c|x_j, z)))$$

State Space. In reinforcement learning, the concept of state space describes all the information about an environment at a given point in time. PLM is pre-trained on a large-scale unlabeled corpus based on self-supervised learning, allowing the model to capture complex language patterns, including long-distance dependence, polysemy disambiguation, sentence structure, etc (Dong et al. 2019; Clark et al. 2020). In this work, we use the last hidden layer embedding of the outputs in the PLMs to represent the state $s$, which is subsequently input into the policy network. To ensure that the difference between states is distinguishable to the RL agent, we dynamically maintain a mean and standard deviation during training of the policy network to normalize observations of the state.

Action Space. An action $a \in A$ is proposed to match an appropriate prompt for an input based on the observed state, where the action space size $|A| = h$. To make action decisions, we train a policy network $\pi_\theta(\cdot)$, which is a simple two-layer fully connected network, and parameters $\theta$ are optimized by the policy gradient algorithm (Sutton et al. 1999). For input $z_t$, $\pi_\theta(\cdot)$ outputs the probability distribution of actions by

$$\pi_\theta(s_t) = \text{softmax}(w_2 \cdot \text{tanh}(w_1 \cdot s_t))$$

where $w_1$ and $w_2$ represent the parameters $\theta$ of the two fully connected layers.

Reward Design. The reward received by the RL agent acts as the feedback that directly guides the update direction of the policy network. In this work, we aim to ensure that the RL-agent-selected prompts for the inputs can accurately complete the downstream task while maintaining balanced predicted label distribution. To achieve this, we re-use the SUE score to evaluate the degree of match between the prompt and input. Specifically, given an input $z_t$, we calculate $\text{SUE}(x_t, z_t)$ as the step reward $r_t$ of the RL agent after selecting the prompt $x_t$.

The reward scale obtained by RL agents can vary greatly due to disparities among different inputs. As a result, RL agents may overly focus on certain inputs during the training phase and become trapped in local optima. To address this issue, we normalize all rewards $r_t$ of the RL agent during training to maintain a relatively stable scale.

Other Key Details. During the training process, we utilize the policy gradient algorithm to update the policy network. To enhance the algorithm’s exploratory potential and accelerate the convergence speed, we follow Sutton (1988) and incorporate entropy into the loss computation of the strategy network. This inclusion allows the policy network to continually optimize the primary loss while maximizing the entropy of the strategy, thereby minimizing the possibility of.
the strategy succumbing to local optimum solutions. Additionally, we use the constant decay method (Tesauro 1991) to control the learning rate of the policy network, which helps the algorithm to converge faster in the early stage of training.

Experiments

To demonstrate the effectiveness of DP₂O, we conduct extensive experiments on four open-source datasets of sentiment classification tasks, including SST-2 (Socher et al. 2013), Yelp (Zhang, Zhao, and LeCun 2015), MR (Pang and Lee 2005), and CR (Hu and Liu 2004), and three tasks of GLUE (Wang et al. 2018) in the few-shot setting. We also analyze the superiority of DP₂O from various aspects: a) Ablation experiments to analyze the effect of modules in DP₂O on downstream tasks; b) Universality in few-shot settings; c) Robustness to choice of verbalizers; d) Generalization for PLMs with different sizes; e) Lightweight and Efficiency method deployment.

Experiment Settings

The setting of comparison experiments, including competitors and our model DP₂O, follows Deng et al. (2022). Also, we utilize a few-shot experiment following Perez, Kiela, and Cho (2021), i.e., randomly select 16 samples from each category c of the dataset as the training set. Meanwhile, we use the same sampling method for the validation set. Therefore, the size of our training and validation sets is $16 \times |C|$.

We chose RoBERT-large (Liu et al. 2019) for all downstream tasks. And we use GPT-4 (OpenAI 2023) API to generate 60 prompts on each dataset, screening out 15 of them as action spaces for reinforcement learning. In the policy network, $w_1 \in \mathbb{R}^{1024 \times 600}$ and $w_2 \in \mathbb{R}^{600 \times 15}$. We use AdamW with eps of 0.00001 during training of 200 epochs. The learning rate is 0.001, and mini-batch size is 32. More details are shown in the appendix.

Competitors

The baselines for comparison are as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>SST-2</th>
<th>MR</th>
<th>CR</th>
<th>Yelp</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Prompt</td>
<td>Soft Prompt Tuning</td>
<td>73.84 ± 10.9</td>
<td>74.17 ± 14.6</td>
<td>75.89 ± 11.8</td>
<td>88.76 ± 4.73</td>
<td>78.17</td>
</tr>
<tr>
<td></td>
<td>Black-Box Tuning</td>
<td>89.11 ± 0.92</td>
<td>86.60 ± 1.32</td>
<td>87.45 ± 1.06</td>
<td>93.22 ± 0.54</td>
<td>89.10</td>
</tr>
<tr>
<td></td>
<td>AutoPrompt</td>
<td>75.04 ± 7.64</td>
<td>62.02 ± 0.85</td>
<td>57.53 ± 5.88</td>
<td>79.81 ± 8.39</td>
<td>68.60</td>
</tr>
<tr>
<td></td>
<td>Manual Prompt †</td>
<td>82.82 ± 0.00</td>
<td>80.88 ± 0.00</td>
<td>79.60 ± 0.00</td>
<td>83.01 ± 0.00</td>
<td>81.58</td>
</tr>
<tr>
<td></td>
<td>Instruction ‡</td>
<td>89.03 ± 0.00</td>
<td>85.18 ± 0.00</td>
<td>80.81 ± 0.00</td>
<td>84.44 ± 0.00</td>
<td>84.87</td>
</tr>
<tr>
<td>Discrete Prompt</td>
<td>In-Context Demo</td>
<td>85.91 ± 0.72</td>
<td>80.58 ± 1.44</td>
<td>85.50 ± 1.52</td>
<td>89.67 ± 0.48</td>
<td>85.42</td>
</tr>
<tr>
<td></td>
<td>GrIPS</td>
<td>87.14 ± 1.57</td>
<td>86.11 ± 0.33</td>
<td>80.02 ± 2.57</td>
<td>88.23 ± 0.17</td>
<td>85.38</td>
</tr>
<tr>
<td></td>
<td>RLPrompt (SOTA)</td>
<td>90.87 ± 0.86</td>
<td>86.85 ± 0.51</td>
<td>89.62 ± 1.36</td>
<td>93.78 ± 2.98</td>
<td>90.28</td>
</tr>
<tr>
<td></td>
<td>DP₂O</td>
<td>93.62 ± 0.72</td>
<td>88.58 ± 0.91</td>
<td>90.76 ± 0.50</td>
<td>94.25 ± 0.41</td>
<td>91.80</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the accuracy of DP₂O and baseline methods on few-shot text classification tasks. The last column shows the average accuracy of each method on the four datasets. Overall, the DP₂O method outperforms baseline methods in all cases. † Methods not affected by random seeds.

Soft Prompt Tuning (Lester, Al-Rfou, and Constant 2021) replaces discrete prompt tokens with learnable feature vectors, and optimizes prompt through gradient information of PLMs.

Black-Box Tuning (Sun et al. 2022) combines the characteristics of discrete and continuous prompt optimization methods, optimizing the sequence of continuous prompt tokens attached to PLMs inputs without gradient.

AutoPrompt (Shin et al. 2020) performs multiple rounds of iteration based on gradient information, replaces the vocabulary in the prompt, and optimizes the discrete prompt template.

Manual Prompt applies the prompt designs of Bach et al. (2022), directly combines the prompt with the input for downstream tasks.

Instruction is a basic form of discrete prompting that facilitates PLMs to complete downstream tasks through an explanatory text. We design prompts for each task according to Mishra et al. (2021).

In-Context Demo (Brown et al. 2020) randomly selects training data as examples to prompt PLMs to process subsequent input.

GrIPS (Prasad et al. 2022) optimizes discrete prompts by lexical-level editing on basic prompts, i.e., substitution, deletion, and swapping, etc.

RLPrompt (Deng et al. 2022) uses reinforcement learning techniques to individually train partial parameters of PLMs to generate discrete prompts for PLMs on downstream tasks.

Performance Comparison

As shown in Table 1, the DP₂O method outperforms its competitors on all datasets. Specifically, compared with the SOTA method RLPrompt, DP₂O achieves accuracy improvements of 2.75%, 1.73%, and 1.14% on SST-2, MR, and CR datasets, respectively. Additionally, on the Yelp dataset, DP₂O still achieved a 0.47% performance improvement with greater stability, despite RLPrompt performing well. Furthermore, compared with other prompt optimization methods using solely supervision (i.e., AutoPrompt and GrIPS), SUE, which combines the unsupervised and su-
pervised components excel. In terms of average accuracy over all datasets, DP₂O performs 23.20% better than Auto-Prompt and 6.42% better than GrIPS in accuracy. Compared to Soft Prompt Tuning, one of the most popular prompt optimization methods, DP₂O achieves 13.63% better accuracy on all four datasets while ensuring prompt readability. Moreover, our proposed multi-round dialogue alignment strategy can build the high-quality prompt set stably, resulting in a smaller standard deviation of DP₂O’s performance compared to Soft Prompt Tuning.

Ablation Study

To study the impact of each component of DP₂O on the final performance, we perform ablation experiments on generation strategy, selection metric, and matching strategy.

Generation Strategy. We compare the prompt generation strategy of DP₂O with two commonly used strategies: Examples-Only and Prompt-Examples (Ubani, Polat, and Nielsen 2023; Min et al. 2022; Dai et al. 2023). Example-Only prompt generation strategy first concatenates a certain number of inputs into a piece of text in random order, then enters the text into GPT-4 in a single round of dialogue for generating the pseudo-label inputs. Prompt-Examples strategy is based on the Examples-Only strategy, applying an expository text prefix to the input combination text. In the experiment, we use the same training data, utilize different generation strategies to generate 20 pseudo-label inputs as prompts and calculate their average accuracy on the test set. We provide the specific input used by the three prompt generation strategies in the appendix.

Table 2 demonstrates the superior performance of SUE in prompt screening. For example, on the SST-2, the average accuracy of the prompts screened by SUE is 0.59% higher than that of the best-performing comparison metric. It is noteworthy that prompts selected solely using unsupervised information achieves comparable performance to supervised information. This finding indicates that DP₂O can potentially perform well on zero-shot tasks.

Matching Strategy. To prove the superiority of utilizing reinforcement learning in matching prompts, we compare it with the two other prompt matching methods, i.e., Random and Similarity-based. The Random method randomly matches the prompt and the input, while the Similarity-based method matches them based on the cosine similarity between the inputs and the prompt feature embeddings.

Table 3 demonstrates the superior performance of SUE in prompt screening. For example, on the SST-2, the average accuracy of the prompts screened by SUE is 0.59% higher than that of the best-performing comparison metric. It is noteworthy that prompts selected solely using unsupervised information achieves comparable performance to supervised information. This finding indicates that DP₂O can potentially perform well on zero-shot tasks.

Table 4: Comparison of the matching strategies.

Table 3: Ablation study on selection metrics.

As shown in Table 4, the matching method for RL in DP₂O achieves the best performance, e.g., a 1.90% improvement in accuracy on SST-2. It indicates our RL agent can capture the implicit connection between the prompt and the input while matching.

Discussions

Analysis on Universality. To demonstrate DP₂O’s universality in few-shot settings, we compared it with baseline methods on the GLUE (Wang et al. 2018) natural language inference and reading comprehension task, using the base template of Gao, Fisch, and Chen (2020). Lacking settings and design of some aforementioned baseline methods on these tasks, here we compared with Soft Prompt Tuning, Black-Box Tuning, Manual Prompt, and In-Context Demo. As shown in Table 5, results show that DP₂O outperforms all baseline methods significantly, including the prevailing methods, Soft Prompt Tuning (Lester, Al-Rfou, and Constant 2021) and Black-Box Tuning (Sun et al. 2022), e.g., DP₂O achieves a performance gain of 2.7% in the QNLI task, and the improvement reaches an astonishing 5.4% in the MRPC task. This results demonstrate that DP₂O’s good universality in the few-shot setting across various tasks, which greatly stimulates the downstream ability of PLMs.

Analysis on Robustness. Prompt-based methods must map the verbalizer probabilities from PLMs’ output into the label
space that downstream tasks require. Therefore, the choice of verbalizer directly affects the final performance of PLMs. Previous work has discussed choosing suitable verbalizers for PLMs. Here we focus on the robustness of DP2O when facing different verbalizer choices, as the results shown in Table 6. We follow the experimental setup of RLPrompt (Deng et al. 2022). Experiments show that DP2O outperforms Manual Prompt at three different verbalizer settings significantly. Meanwhile, compared to the SOTA method RL Prompt, DP2O also surpasses it slightly, which accounts for DP2O’s better robustness to the choice of verbalizer.

### Table 6: Analysis on DP2O’s robustness to verbalizers.

<table>
<thead>
<tr>
<th>Verbalizer</th>
<th>Manual</th>
<th>RL Prompt</th>
<th>DP2O</th>
</tr>
</thead>
<tbody>
<tr>
<td>bad/good</td>
<td>79.73</td>
<td>91.22 ± 1.46</td>
<td>91.96 ± 0.41</td>
</tr>
<tr>
<td>neg./pos.</td>
<td>76.89</td>
<td>92.20 ± 0.65</td>
<td>93.64 ± 0.77</td>
</tr>
<tr>
<td>ter./great</td>
<td>82.86</td>
<td>92.81 ± 0.85</td>
<td>93.58 ± 0.51</td>
</tr>
</tbody>
</table>

**Analysis on Generalization.** We analyze the model generalization for PLMs with different sizes, which is involved in two modules of DP2O: prompt generalization and policy generalization.

First, the prompts generated by DP2O can transfer between PLMs of different sizes. That is, the prompts computed SUE and selected based on a smaller PLM can also achieve good performance for downstream tasks in another larger PLM.

### Table 7: Analysis on generalization ability of DP2O’s prompts on different size PLMs.

<table>
<thead>
<tr>
<th>Method</th>
<th>SST-2</th>
<th>MR</th>
<th>CR</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Prompt</td>
<td>82.82</td>
<td>80.88</td>
<td>79.60</td>
<td>83.01</td>
</tr>
<tr>
<td>DP2O generalized</td>
<td>83.33</td>
<td>80.38</td>
<td>84.66</td>
<td>89.26</td>
</tr>
<tr>
<td>DP2O</td>
<td>93.62</td>
<td>88.58</td>
<td>90.76</td>
<td>94.25</td>
</tr>
</tbody>
</table>

Impressively, Table 7 shows that the prompts selected by smaller PLMs are well-transferable with a minor decline in accuracy than the vanilla, still achieving comparable performance to the Manual Prompt baseline.

The policy generalization concerns whether the trained policy network of DP2O can function well on different PLMs. We train a policy network on RoBERTa-base and apply it to RoBERTa-large. In this test, we keep the prompts unchanged and only focus on evaluating the policy’s performance. Table 8 shows that even if using a smaller model to train the policy network, its performance on the large model version is still better than the commonly used random policy. Also, generalized DP2O only shows a slight decrease in accuracy to the vanilla.

### Table 8: Analysis on generalization ability of the policy.

<table>
<thead>
<tr>
<th>Method</th>
<th>SST-2</th>
<th>MR</th>
<th>CR</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>88.48</td>
<td>85.07</td>
<td>86.00</td>
<td>90.22</td>
</tr>
<tr>
<td>DP2O generalized</td>
<td>89.34</td>
<td>86.40</td>
<td>87.12</td>
<td>91.36</td>
</tr>
<tr>
<td>DP2O</td>
<td>93.62</td>
<td>88.58</td>
<td>90.76</td>
<td>94.25</td>
</tr>
</tbody>
</table>

**Analysis on Lightweight and Efficiency.** DP2O only needs to train a two-layer fully connected network for its policy network. The number of parameters is 0.62M, which is only 0.73% of the whole policy network (distilGPT-2 with 82.0M parameters and an additional MLP with 3.15M parameters) used by RL Prompt in the experiment.

Meanwhile, as shown in Table 9, we compare the time consumption of DP2O and the SOTA method RL Prompt on the SST-2 dataset using a single NVIDIA GeForce RTX 3090 GPU. We find that the compact action space design in DP2O dramatically reduces the training time, which is only 10.86% of RL Prompt’s, while DP2O’s accuracy exceeds RL Prompt as mentioned in Table 1.

### Table 9: Time consumption on SST-2 dataset.

<table>
<thead>
<tr>
<th>Metric</th>
<th>RL Prompt</th>
<th>DP2O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time per Iterator</td>
<td>1.09 s</td>
<td>1.01 s</td>
</tr>
<tr>
<td>Training Time</td>
<td>218.63 min</td>
<td>23.75 min</td>
</tr>
</tbody>
</table>

**Conclusion**

In this paper, we propose DP2O, a novel discrete prompt optimization method. To efficiently and accurately select high-quality prompts, we design a prompt generation strategy through multi-round dialogue alignment on GPT-4 and propose an efficient prompt evaluation metric, SUE. In addition, we design a reinforcement learning framework based on policy gradients to match suitable prompts for a single input. Our experimental results demonstrate that DP2O significantly improves the performance of PLMs in various downstream tasks while ensuring prompt readability and transferability. In subsequent analysis experiments, we also verify DP2O’s good universality, robustness, generalization ability, lightweight and efficiency.
Acknowledgments

This work is supported by National Key R&D Program (2020YFBJ0406900), National Natural Science Foundation of China (62272371, 62103323, U21B2018, 62161160337, 62132011, 62376210, 62006181, U20B2049), Initiative Postdocs Supporting Program (BX20190275, BX20200270), China Postdoctoral Science Foundation (2019M663723, 2021M692565), Fundamental Research Funds for the Central Universities under grant (xj032021013, xtr052023004, xtr022019002), and Shaanxi Province Key Industry Innovation Program (2021ZDLGY01-02).

References


OpenAI. 2022. ChatGPT. Website.


