Dialogue for Prompting: A Policy-Gradient-Based Discrete Prompt Generation for Few-Shot Learning

Chengzhengxu Li, Xiaoming Liu*, Yichen Wang, Duyi Li, Yu Lan, Chao Shen

Faculty of Electronic and Information Engineering, Xi'an Jiaotong University
{czx.li, yichen.wang, liduyi}@stu.xjtu.edu.cn, {xm.liu, ylan2020, chaoshen}@xjtu.edu.cn

Abstract

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 **D**ialogue-comprised **P**olicy-gradient-based **D**iscrete **P**rompt **O**ptimization (DP_2O) 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, DP_2O 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 DP_2O has good universality, robustness and generalization ability.

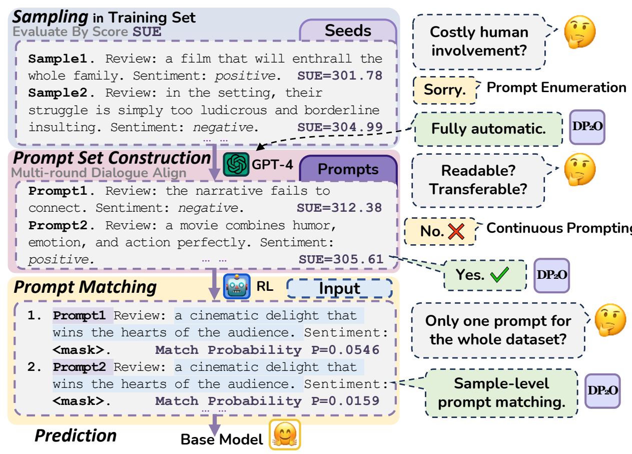

Figure 1: An illustration of the procedure and innovation Q&A of DP_2O. The procedure includes 1) Sampling the seed set from the training set via SUE; 2) Constructing the prompt by multi-round dialogue with GPT-4 to align the inputs with the whole training set's distribution; 3) Employing an RL agent to match prompts with inputs to predict probabilistically; 4) Feeding all prompt-input pairs to a base PLM model for downstream tasks and ensemble predictions by probability weighting.

Introduction

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 (Holtzman 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

*Corresponding author

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 **D**ialogue-comprised **P**olicy-gradient-based **D**iscrete **P**rompt **O**ptimization method, named $\mathrm{DP_2O}$. As shown in Figure 1, $\mathrm{DP_2O}$ 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 $\mathrm{DP_2O}$ is significantly superior to baseline and SOTA methods, *e.g.*, $\mathrm{DP_2O}$ achieve an average improvement of **1.52%** in accuracy across four public datasets with only a **10.86%** training time of the SOTA method RLPrompt (Deng et al. 2022). Furthermore, we implement ablation and analysis experiments to demonstrate the effectiveness, robustness and generalization of $\mathrm{DP_2O}$.

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 $\mathrm{DP_2O}$ effectively improves the performance of PLMs under few-shot settings with readability, robustness, generalization, and universality.

Methodology

The main workflow of $\mathrm{DP_2O}$ 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 **S**upervised & **U**nsupervised **E**ntropy 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_{sup} and *unsupervised score* S_{uns}. Given a prompt x and input set $\mathcal{Z}$, for each input $z_i \in \mathcal{Z}$, we first calculate the difference of the probability p_{LM} that the z_i is correctly labeled c_i and wrongly labeled as c_{else} by a base PLM. Here $c_{\text{else}} \in \mathcal{C} \setminus \{c_i\}$ exactly, and $\mathcal{C}$ is the label space of the input. Then the supervision score S_{sup} of prompt x is defined as:

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

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 a entropy value $\mathbb{H}(.)$ of each input z_i, then add $\mathbb{H}(.)$ of each input as S_{uns} for the whole input set $\mathcal{Z}$:

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

$$S_{\text{uns}}(x, \mathcal{Z}) = \sum\nolimits_{z_i \in \mathcal{Z}} \mathbb{H}(x, z_i) \quad (3)$$

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

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

where λ_1 and λ_2 are the weights to balance the supervised score S_{sup} and the unsupervised score S_{uns}. For the input set $\mathcal{Z}$ 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_{sup}) and more benign confidence on all inputs (derived from S_{uns}). Meanwhile, when $\mathcal{Z}$ 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, $\mathrm{DP_2O}$ leverages GPT-4 as a dialogue model to generate pseudo-label inputs which approximate the dataset distribution, utilizing only a limited

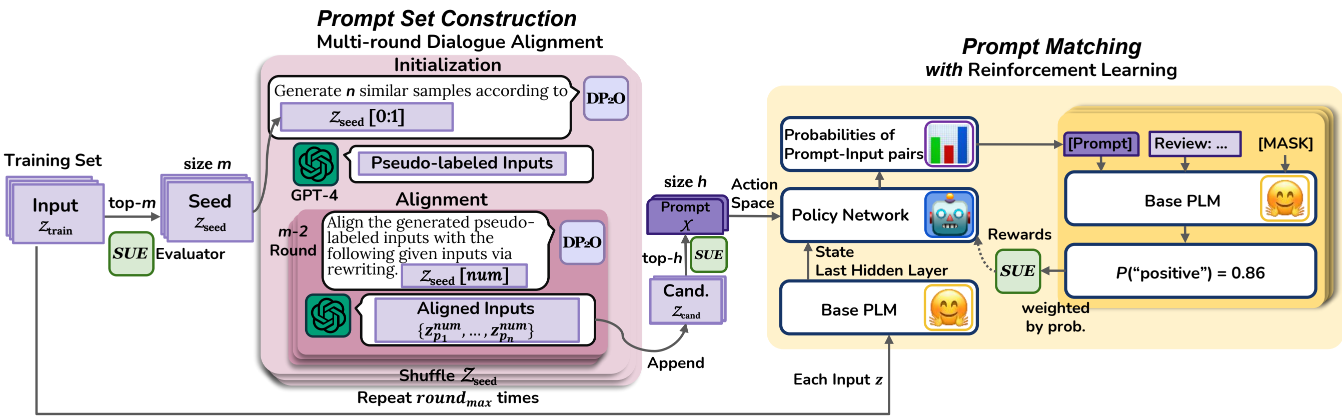

Figure 2: Overview of $\mathrm{DP_2O}$. In the *prompt set construction* stage, we use the multi-round dialogue alignment strategy to generate high-quality discrete prompts continuously. Given the seed inputs $\mathcal{Z}_{\text{seed}}$ with top-m SUE score, $\mathrm{DP_2O}$ have a conversation with GPT-4, which has $round_{max}$ times outer loop and $m-2$ times inner loop, to align inputs semantics with the training set. Then $\mathrm{DP_2O}$ apply the assessment metric SUE to sort the prompts after dialogue to obtain the final candidate set $\mathcal{Z}_{\text{cand}}$. We filter top-$h$ candidates as the final prompt set based on SUE score. In the *prompt matching* stage, we build a reinforcement learning framework to match the appropriate prompt from $\mathcal{X}$ for each input z from $\mathcal{Z}_{\text{train}}$ with probability. The prompt-input pairs are fed into the base PLM to predict downstream tasks. The final prediction is the probability-weighted output of all pairs.

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 $\mathrm{DP_2O}$, 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, $\mathrm{DP_2O}$ can still achieve praiseworthy and competitive performance to other methods.

Algorithm 1: Prompt Set Construction of $\mathrm{DP_2O}$

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

$\mathcal{Z}_{\text{seed}} \leftarrow$ top-m inputs in $\mathcal{Z}_{\text{train}}$ via $\mathrm{SUE}(z_i, \mathcal{Z}_{\text{train}})$, $z_i \in \mathcal{Z}_{\text{train}}$.

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

$round \leftarrow 0$

while $round < round_{max}$ **do**

  Random shuffle $\mathcal{Z}_{\text{seed}}$.

  Input $\mathcal{Z}_{\text{seed}}[0:1]$ to GPT-4 with prefix introduction of task.

  GPT-4 output n pseudo-labeled inputs $\{z_{p_1}^1, \ldots, z_{p_n}^1\}$.

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

  Initialize number of used inputs in $\mathcal{Z}_{\text{seed}}$ $num \leftarrow 2$.

  while $num < m$ **do**

    Input $\mathcal{Z}_{\text{seed}}[num]$ to GPT-4, asking it rewrite the previous $\{z_{p_1}^{num-1}, \ldots, z_{p_n}^{num-1}\}$.

    GPT-4 output $\{z_{p_1}^{num}, \ldots, z_{p_n}^{num}\}$.

    $num \leftarrow num + 1$

  end while

  Append $\{z_{p_1}^{m-1}, \ldots, z_{p_n}^{m-1}\}$ to $\mathcal{Z}_{\text{cand.}}$

end while

$\mathcal{X} \leftarrow$ top-h inputs of $\mathcal{Z}_{\text{cand.}}$ via $\mathrm{SUE}(z_i, \mathcal{Z}_{\text{train.}})$, $z_i \in \mathcal{Z}_{\text{cand.}}$

Output: Readable and high-quality prompt set $\mathcal{X}$.

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 $\mathcal{Z}_{\text{train}}$, we first individually evaluate each input $z_i \in \mathcal{Z}_{\text{train}}$ by score $\mathrm{SUE}(z_i, \mathcal{Z}_{\text{train}})$. SUE signifies the input z_i's efficacy within the given set $\mathcal{Z}_{\text{train}}$. We rank $\mathcal{Z}_{\text{train}}$ set in descending order based on the $\mathrm{SUE}(z_i, \mathcal{Z}_{\text{train}})$. We then select the top-m inputs to form the seed set $\mathcal{Z}_{\text{seed}} = \{z_{\text{seed}_1}, z_{\text{seed}_2}, \ldots, z_{\text{seed}_m}\}$.

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

In one *dialogue round*, we first generate n pseudo-labeled inputs $\{z_{p_1}^1, \ldots, z_{p_n}^1\}$ based on any two inputs from $\mathcal{Z}_{\text{seed}}$. Then we randomly select one of the remaining inputs from $\mathcal{Z}_{\text{seed}}$ into dialogue to guide GPT-4 to polish the previously generated pseudo-labeled inputs $\{z_{p_1}^1, \ldots, z_{p_n}^1\}$ and get corresponding $\{z_{p_1}^2, \ldots, z_{p_n}^2\}$. Repeated the polishing stage (inner-loop) $m-2$ times until all z_{seed} inputted to the dialogue once. Then we get $\{z_{p_1}^{m-1}, \ldots, z_{p_n}^{m-1}\}$ and append them in candidate set $\mathcal{Z}_{\text{cand}}$.

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 $round_{max}$ times *dialogue rounds* (outer-loop), we compute the score SUE of all $n \times round_{max}$ inputs in $\mathcal{Z}_{\text{cand}}$. Then, we select the top-h inputs from the candidate set $\mathcal{Z}_{\text{cand}}$ as the final prompt set $\mathcal{X}$. From now on, the selected inputs go into the role of prompt.

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 DP_2O

```
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 ****
Initialize the policy network π_θ parameters θ and epoch ← 0.
while epoch < epoch_max do
    for step t in [1, ..., T] do
        Get state s_t ← PLM(z_t).
        Run policy network π_θ(a_t|s_t) to take an action a_t to select a prompt x_t.
        Calculate reward by SUE, i.e., r_t ← SUE(x_t, z_t).
        Add transition to replay buffer.
    end for
    Update parameters θ of π_θ with the policy gradient loss.
end while
                    **** testing phase begins ****
for each input z in Z_test do
    Get state s ← PLM(z).
    Get final prediction according to Eq. 5.
end for
Output: A trained policy network π_θ, 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_k denotes that the agent selects a prompt x_k from the prompt set $\mathcal{X}$ 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 $\mathcal{X}$ according to the policy $\pi_\theta(a_t|s_t)$ where θ 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 $R = \mathbb{E}(\sum_{t=1}^{T} \gamma^t r_t)$, where γ^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 π_θ. Given an input z and its corresponding state s, the final prediction obtained by DP_2O at label c can be expressed as

$$P(c|z) = \text{softmax}(\sum_{j=1}^{k} \pi_\theta(a_j|s) \log(p_{\text{LM}}(c|x_j, z))) \quad (5)$$

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 $\pi_\theta(.)$, which is a simple two-layer fully connected network, and parameters θ are optimized by the policy gradient algorithm (Sutton et al. 1999). For input z_t, $\pi_\theta(.)$ outputs the probability distribution of actions by

$$\pi_\theta(s_t) = \text{softmax}(w_2 \cdot \tanh(w_1 \cdot s_t)) \quad (6)$$

where w_1 and w_2 represent the parameters θ 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

Category	Method	*SST-2*	*MR*	*CR*	*Yelp*	*Avg.*
Continuous Prompt	Soft Prompt Tuning	73.84 ± 10.9	74.17 ± 14.6	75.89 ± 11.8	88.76 ± 4.73	78.17
	Black-Box Tuning	89.11 ± 0.92	86.60 ± 1.32	87.45 ± 1.06	93.22 ± 0.54	89.10
	AutoPrompt	75.04 ± 7.64	62.02 ± 0.85	57.53 ± 5.88	79.81 ± 8.39	68.60
Discrete Prompt	Manual Prompt †	82.82 ± 0.00	80.88 ± 0.00	79.60 ± 0.00	83.01 ± 0.00	81.58
	Instruction †	89.03 ± 0.00	85.18 ± 0.00	80.81 ± 0.00	84.44 ± 0.00	84.87
	In-Context Demo	85.91 ± 0.72	80.58 ± 1.44	85.50 ± 1.52	89.67 ± 0.48	85.42
	GrIPS	87.14 ± 1.57	86.11 ± 0.33	80.02 ± 2.57	88.23 ± 0.17	85.38
	RLPrompt (SOTA)	90.87 ± 0.86	86.85 ± 0.51	89.62 ± 1.36	93.78 ± 2.98	90.28
	DP$_2$O	$\mathbf{93.62 \pm 0.72}$	$\mathbf{88.58 \pm 0.91}$	$\mathbf{90.76 \pm 0.50}$	$\mathbf{94.25 \pm 0.41}$	**91.80**

Table 1: Comparison of the accuracy of DP$_2$O 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 DP$_2$O method outperforms baseline methods in all cases. † Methods not affected by random seeds.

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 DP$_2$O, 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 DP$_2$O from various aspects: a) **Ablation experiments** to analyze the effect of modules in DP$_2$O 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 DP$_2$O, 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 $16 \times |\mathcal{C}|$.

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, $w_1 \in \mathbb{R}^{1024\times600}$ and $w_2 \in \mathbb{R}^{600\times15}$. 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:

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 DP$_2$O method outperforms its competitors on all datasets. Specifically, compared with the SOTA method RLPrompt, DP$_2$O achieves accuracy improvements of **2.75%**, **1.73%**, and **1.14%** on SST-2, MR, and CR datasets, respectively. Additionally, on the Yelp dataset, DP$_2$O still achieved a **0.47%** performance improvement with greater stability, despite RLPropmt performing well. Furthermore, compared with other prompt optimization methods using sorely supervision (*i.e.*, AutoPrompt and GrIPS), SUE, which combines the unsupervised and supervised components excels. In terms of average accuracy over all datasets, DP_2O performs **23.20%** better than AutoPrompt and **6.42%** better than GrIPS in accuracy. Compared to Soft Prompt Tuning, one of the most popular prompt optimization methods, DP_2O 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 DP_2O's performance compared to Soft Prompt Tuning.

Ablation Study

To study the impact of each component of DP_2O 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 DP_2O 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 explanatory 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.

Method	***SST-2***	***MR***	***CR***	***Yelp***
Examples Only	86.34	75.80	80.16	90.60
Prompt Examples	78.07	83.23	88.43	87.95
DP_2O Examples	**89.46**	**85.69**	**88.99**	**91.67**

Table 2: Comparison of prompt generation strategies. DP_2O Examples are generated via the *multi-round dialogue alignment strategy*.

Table 2 demonstrates that the DP_2O's prompt generation strategy, *i.e.*, *multi-round dialogue alignment strategy*, results in an average accuracy improvement of **3.12%**, **2.46%**, **0.56%** and **1.07%** on the SST-2, MR, CR and Yelp datasets than the best comparison strategy, respectively. We also found that *Examples-Only* and *Prompt-Examples* strategies show significant performance fluctuations when the dataset changes. In contrast, our *multi-round dialogue alignment* strategy is much more stable, indicating that DP_2O generates a superior set of prompts by aligning with the training set data via GPT-4 dialogue.

Selection Metric. Our prompt screening metric SUE consists of two parts: *supervised information* and *unsupervised information*. To evaluate the effectiveness of each component, we compare SUE to use these two sole parts. We select the top-15 prompts with the highest scores on each metric from the same prompt set and then calculate their average accuracy on the test set.

Method	***SST-2***	***MR***	***CR***	***Yelp***
Supervised	85.07	78.45	87.55	90.78
Unsupervised	87.13	78.37	87.35	91.32
SUE in DP_2O	**87.72**	**78.60**	**88.01**	**92.71**

Table 3: Ablation study on selection metrics.

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 DP_2O 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.

Method	***SST-2***	***MR***	***CR***	***Yelp***
Random	92.13	87.34	89.50	92.03
Similarity-based	91.38	88.60	89.33	92.61
RL in DP_2O	**94.03**	**89.07**	**90.95**	**94.32**

Table 4: Comparison of the matching strategies.

As shown in Table 4, the matching method for RL in DP_2O 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 DP_2O'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 DP_2O outperforms all baseline methods significantly, including the prevailing methods, Soft Prompt Tunning (Lester, Al-Rfou, and Constant 2021) and Black-Box Tuning (Sun et al. 2022). *e.g.*, DP_2O, 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 DP_2O'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

Method	***RTE***	***QNLI***	***MRPC***
Soft Prompt Tuning	54.7 ± 10.6	49.7 ± 1.73	51.6 ± 2.39
Black-Box Tuning	52.9 ± 0.44	48.8 ± 0.61	61.6 ± 0.97
Manual Prompt	51.6 ± 0.00	50.8 ± 0.00	61.1 ± 0.00
In-Context Demo.	59.7 ± 0.85	52.4 ± 0.67	45.8 ± 0.80
DP_2O	$\mathbf{61.2 \pm 0.81}$	$\mathbf{55.1 \pm 0.39}$	$\mathbf{67.0 \pm 1.03}$

Table 5: Analysis on model universality. We use the GLUE Benchmark[1] online evaluation, whose results are three-digit decimal numbers.

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 DP_2O 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 DP_2O outperforms Manual Prompt at three different verbalizer settings significantly. Meanwhile, compared to the SOTA method RLPrompt, DP_2O also surpasses it slightly, which accounts for DP_2O's better robustness to the choice of verbalizer.

Verbalizer	**Manual**	**RLPrompt**	**DP_2O**
`bad/good`	79.73	91.22 ± 1.46	$\mathbf{91.96 \pm 0.41}$
`neg./pos.`	76.89	92.20 ± 0.65	$\mathbf{93.64 \pm 0.77}$
`ter./great`	82.86	92.81 ± 0.85	$\mathbf{93.58 \pm 0.51}$

Table 6: Analysis on DP_2O's robustness to verbalizers. For short, `neg.` is `negative`, `pos.` is `positvie`, and `ter.` is `terrible`.

Analysis on Generalization. We analyze the model generalization for PLMs with different sizes, which is involved in two modules of DP_2O: *prompt generalization* and *policy generalization*.

First, the prompts generated by DP_2O 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.

Method	***SST-2***	***MR***	***CR***	***Yelp***
Manual Prompt	82.82	80.88	79.60	83.01
DP_2O generalized	83.33	80.38	84.66	89.26
DP_2O	93.62	88.58	90.76	94.25

Table 7: Analysis on generalization ability of DP_2O's prompts on different size PLMs.

In this experiment, we use the metric scores output from the RoBERTa-base (110M parameters) to select the prompts and test their generalization on RoBERTa-large (354M parameters) to get the downstream task prediction accuracy. 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 DP_2O 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 DP_2O only shows a slight decrease in accuracy to the vanilla.

Method	***SST-2***	***MR***	***CR***	***Yelp***
Random	88.48	85.07	86.00	90.22
DP_2O generalized	89.34	86.40	87.12	91.36
DP_2O	93.62	88.58	90.76	94.25

Table 8: Analysis on generalization ability of the policy.

Analysis on Lightweight and Efficiency. DP_2O 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 RLPrompt in the experiment.

Meanwhile, as shown in Table 9, we compare the time consumption of DP_2O and the SOTA method RLPrompt on the SST-2 dataset using a single NVIDIA GeForce RTX 3090 GPU. We find that the compact action space design in DP_2O dramatically reduces the training time, which is only **10.86%** of RLPrompt's, while DP_2O's accuracy exceeds RLPrompt as mentioned in Table 1.

Metric	**RLPrompt**	**DP_2O**
Time per Iterator	1.09 s	1.01 s
Training Time	218.63 min	23.75 min

Table 9: Time consumption on SST-2 dataset.

Conclusion

In this paper, we propose DP_2O, 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 DP_2O significantly improves the performance of PLMs in various downstream tasks while ensuring prompt readability and transferability. In subsequent analysis experiments, we also verify DP_2O's good universality, robustness, generalization ability, lightweight and efficiency.

Acknowledgments

This work is supported by National Key R&D Program (2020YFB1406900), 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 (xhj032021013, xtr052023004, xtr022019002), and Shaanxi Province Key Industry Innovation Program (2021ZDLGY01-02).

References

An, S.; Li, Y.; Lin, Z.; Liu, Q.; Chen, B.; Fu, Q.; Chen, W.; Zheng, N.; and Lou, J.-G. 2022. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*.

Anil, R.; Dai, A. M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Bach, S. H.; Sanh, V.; Yong, Z.-X.; Webson, A.; Raffel, C.; Nayak, N. V.; Sharma, A.; Kim, T.; Bari, M. S.; Fevry, T.; et al. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Dai, H.; Liu, Z.; Liao, W.; Huang, X.; Wu, Z.; Zhao, L.; Liu, W.; Liu, N.; Li, S.; Zhu, D.; et al. 2023. Chataug: Leveraging chatgpt for text data augmentation. *arXiv preprint arXiv:2302.13007*.

Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; and Liu, R. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Davison, J.; Feldman, J.; and Rush, A. M. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 1173–1178.

Deng, M.; Wang, J.; Hsieh, C.-P.; Wang, Y.; Guo, H.; Shu, T.; Song, M.; Xing, E. P.; and Hu, Z. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32.

Gao, T.; Fisch, A.; and Chen, D. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Haviv, A.; Berant, J.; and Globerson, A. 2021. BERTese: Learning to speak to BERT. *arXiv preprint arXiv:2103.05327*.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Hu, M.; and Liu, B. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 168–177.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8: 423–438.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, X. L.; and Liang, P. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lu, Y.; Bartolo, M.; Moore, A.; Riedel, S.; and Stenetorp, P. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; and Zettlemoyer, L. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Mishra, S.; Khashabi, D.; Baral, C.; and Hajishirzi, H. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

OpenAI. 2022. ChatGPT. Website.

OpenAI. 2023. GPT-4 Technical Report. *ArXiv*, abs/2303.08774.

Pang, B.; and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Perez, E.; Kiela, D.; and Cho, K. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34: 11054–11070.

Prasad, A.; Hase, P.; Zhou, X.; and Bansal, M. 2022. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv preprint arXiv:2203.07281*.

Qian, J.; Dong, L.; Shen, Y.; Wei, F.; and Chen, W. 2022. Controllable natural language generation with contrastive prefixes. *arXiv preprint arXiv:2202.13257*.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1): 5485–5551.

Schick, T.; and Schütze, H. 2020a. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

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

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Sun, T.; Shao, Y.; Qian, H.; Huang, X.; and Qiu, X. 2022. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, 20841–20855. PMLR.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Tesauro, G. 1991. Practical issues in temporal difference learning. *Advances in neural information processing systems*, 4.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ubani, S.; Polat, S. O.; and Nielsen, R. 2023. ZeroShotDataAug: Generating and Augmenting Training Data with ChatGPT. *arXiv preprint arXiv:2304.14334*.

Vu, T.; Lester, B.; Constant, N.; Al-Rfou, R.; and Cer, D. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yuan, W.; Neubig, G.; and Liu, P. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34: 27263–27277.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Zhou, Y.; Muresanu, A. I.; Han, Z.; Paster, K.; Pitis, S.; Chan, H.; and Ba, J. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.