Dual-Phase Accelerated Prompt Optimization

Muchen Yang[1], **Moxin Li**[2*], **Yongle Li**[3], **Zijun Chen**[3],
Chongming Gao[1*], **Junqi Zhang**[4], **Yangyang Li**[5], **Fuli Feng** [1,3]
[1]University of Science and Technology of China, [2]National University of Singapore
[3]Institute of Dataspace, Hefei Comprehensive National Science Center
[4]AtomEcho Inc., [5]Academy of Cyber
muchen00@mail.ustc.edu.cn, limoxin@u.nus.edu, liyongle999@gmail.com
zijunchen248@gmail.com, chongminggao@ustc.edu.cn, zhangjunqi@atomecho.xyz
liyangyang@live.com, fulifeng93@gmail.com

Abstract

Gradient-free prompt optimization methods have made significant strides in enhancing the performance of closed-source Large Language Models (LLMs) across a wide range of tasks. However, existing approaches make light of the importance of high-quality prompt initialization and the identification of effective optimization directions, thus resulting in substantial optimization steps to obtain satisfactory performance. In this light, we aim to accelerate prompt optimization process to tackle the challenge of low convergence rate. We propose a dual-phase approach which starts with generating high-quality initial prompts by adopting a well-designed meta-instruction to delve into task-specific information, and iteratively optimize the prompts at the sentence level, leveraging previous tuning experience to expand prompt candidates and accept effective ones. Extensive experiments on eight datasets demonstrate the effectiveness of our proposed method, achieving a consistent accuracy gain over baselines with less than five optimization steps.

1 Introduction

LLMs have demonstrated remarkable capabilities across a wide range of natural language processing (NLP) tasks, including machine translation (**?**), summarization (**?**), and question answering (**?**). The dependency on prompt quality has led to the emergence of prompt engineering (**??**), aiming at crafting effective prompts to elicit the desired responses from LLMs. As the need for efficient prompt design becomes increasingly evident (**?**), automatic prompt optimization has been introduced to streamline the prompt design process, ensuring that LLMs are utilized to their full potential (**???**).

Automatic prompt optimization can be broadly categorized into gradient-based and gradient-free methods. Gradient-based methods (**????**) are devised for open-source LLMs to enable the optimization of prompts through adjustments based on model gradient. Gradient-free methods have emerged as the predominant approach for closed-source LLMs, which focuses on refining prompts without access to the model gradient (**???**). Starting from initial prompts, these methods usually expand candidate prompts using searching methods (**??**) and then accepting the more prominent ones in an iterative manner. This paper focuses on gradient-free methods due to the distinguished abilities of closed-source LLMs and the challenge of optimizing their prompts with limited model information.

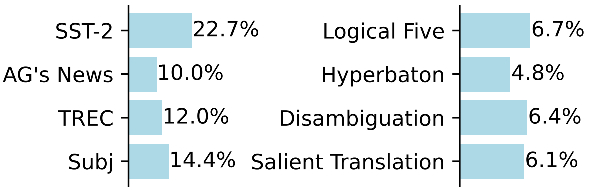

Figure 1: Average accuracy improvement on eight datasets with *four* optimization steps.

We argue that current gradient-free prompt optimization methods have not adequately considered the rate of convergence. Typically, these methods demand an excessive number of optimization steps to obtain satisfactory prompts due to the limited access to model details, the vast discrete search space, and the uncertain optimization directions (**???**). Representative work such as OPRO (**?**) even necessitates nearly 200 optimization steps for some NLP tasks. This requirement for excessive optimization steps makes existing methods impractical for real-world applications since users are understandably reluctant to tolerate extensive optimization steps to achieve satisfactory performance levels. Therefore, we aim to achieve accelerated prompt optimization, obtaining satisfactory performance via few optimization steps (*e.g.,* < 5).

To achieve accelerated prompt optimization, two

[*] Corresponding Author

crucial factors need to be considered: high-quality initial prompts and effective optimization directions. Firstly, the initialization of the prompt plays a crucial role in determining the efficiency of the optimization process (?), whereas existing approaches pay insufficient attention to the impact of initialization on subsequent optimization. Therefore, we aim to obtain initial prompts of high quality, laying a solid foundation to accelerate optimization process. Secondly, the accelerated prompt optimization needs to identify the most effective optimization directions in each step, streamlining efficient optimization from the initial prompts. Thus, we aim to design a more refined expansion tuned by experience and acceptance of candidate prompts enhanced by examination of failure cases.

To this end, we propose a dual-phase approach to achieve the accelerated gradient-free prompt optimization. Our approach consists of two phases: high-quality initial prompt generation, and experience-tuned optimization. Firstly, we utilize a well-designed meta-instruction to guide the LLM in generating high-quality and structured initial prompts that contain task-specific information, including task type and description, output format and constraints, suggested reasoning process, and professional tips. After that, we devise a sentence-level prompt optimization strategy for efficiently optimization on the long initial prompt, leveraging previous direction tuning experience, together with failure cases, to select sentences in the initial prompt to be expanded and accept effective prompt candidates. Extensive experiments (*cf.* Figure ??) on three LLMs across several datasets confirm the effectiveness and superiority of our method. Our contributions are threefold:

- We reveal the issue of low convergence rate in gradient-free prompt optimization, and highlight the problem of accelerated prompt optimization.
- We propose a dual-phase approach, achieving accelerated prompt optimization through high-quality initial prompt generation and experience-tuned optimization.
- We conduct extensive experiments, demonstrating that the proposed method achieves satisfying performance within few optimization steps.

2 Related Work

The gradient-free prompt optimization for closed-source LLMs typically contains two phases: initialization and iterative optimization steps, where the optimization step consists of expansion and selection stages.

Initialization. The prompt initialization for optimization can be achieved manually or autonomously. Manual initialization often entails professional machine learning engineers formulating prompts, as delineated in (?). Concurrently, works such as (?), (?), and (?) utilize existing manual prompts as the foundational set to harness human creativity. In contrast, automated initialization leverages the power of LLM generation, which is exemplified by (?), generating prompts from few-shot exemplars and a rudimentary description, and (?), fabricating prompts based on meta-prompts and illustrative input-output examples. Our method belongs to the automated initialization, improving the initial prompt generation for acceleration.

Optimization. The optimization step is achieved by expanding prompt candidates by modifying from the initial prompt and selecting the better candidates for the next iteration. The expansion stage can be executed through rephrasing, as in (?), where high-scoring prompts undergo evolution akin to a Monte Carlo search methodology, or through heuristic algorithms that automatically revise prompts, as in (?) and (?). More complex regeneration strategies are employed by works like (?), where the optimizer LLM progressively expands prompts based on task delineations and historical iterations. The expansion can also be implemented leveraging an open-source LLM (??). Reinforcement learning-based methods have also been adopted for prompt modification (?). Moreover, the granularity of prompt modification exhibits variation across studies. Heuristic-based methods and (?) work operate at the word/token granularity, while classical optimization algorithms like (??) consider the entire prompt. The selection stage generally utilized the performance of the prompt on a held-out validation set (???), while recent work also explores human preference feedback (?) or score feedback from other LLMs (?).

3 Problem Formulation

3.1 Gradient-Free Prompt Optimization

For a target NLP task $\mathcal{T}$ with input x, the closed-source LLM predicts the output $\hat{y}$ given x concatenated with the prompt p, where $x, \hat{y}$ and p are all word sequences. The aim for prompt optimization

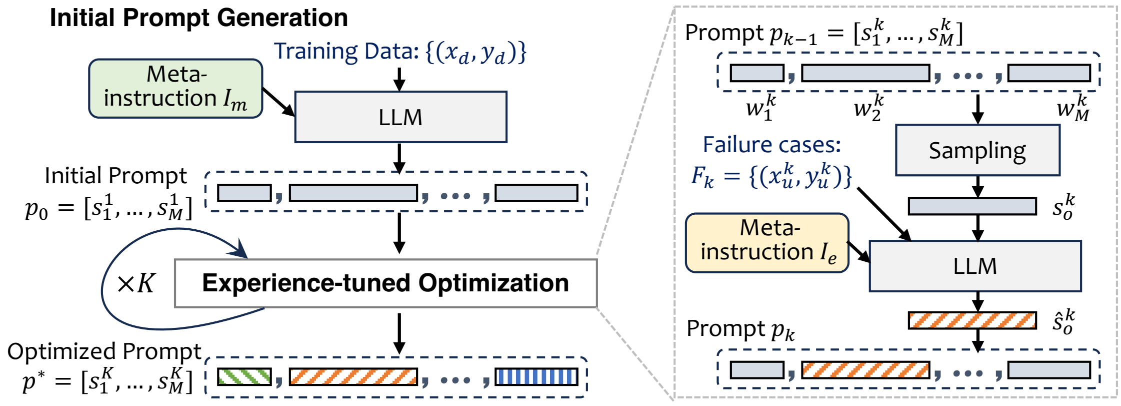
Initial Prompt Generation
Training Data: $\{(x_d, y_d)\}$
Meta-instruction I_m
LLM
Initial Prompt $p_0 = [s_1^1, \ldots, s_M^1]$
$\times K$
Experience-tuned Optimization
Optimized Prompt $p^* = [s_1^K, \ldots, s_M^K]$
Prompt $p_{k-1} = [s_1^k, \ldots, s_M^k]$
w_1^k
w_2^k
w_M^k
Failure cases:
$F_k = \{(x_u^k, y_u^k)\}$
Sampling
s_o^k
Meta-instruction I_e
LLM
Prompt p_k
$\hat{s}_o^k$

Figure 2: Illustration of the proposed method.

is to find an optimal prompt p^* that obtains the desired $\hat{y}$, which can be evaluated by metrics such as accuracy with reference to the ground truth y. The gradient-free prompt optimization contains an initialization phase followed by K iterative optimization steps. The k-th optimization step starts from an initial prompt $p_{k-1}, k \in [1, K]$, and sequentially performs two stages: expansion of prompt candidates, and acceptance of the prominent prompts as the next initial prompts, as detailed below.

Expansion of Prompt Candidates. At the k-th optimization step, The expansion stage search for new prompt candidates with potential better performance starting from p_{k-1}, with searching methods such as edit-based (**?**) and LLM rewriting (**?**). Formally, the expansion function $f_E(\cdot)$ generates prompt candidate set $P_k^c = \{p_{k_1}^c, \cdots, p_{k_Q}^c\}$ with size Q.

$$P_k^c = f_E(p_{k-1}). \tag{1}$$

Acceptance of Prominent Prompts. The acceptance stage evaluates the performance of each prompt candidate in P_k^c to determine whether it should be continued for next optimization step. This is usually achieved by evaluation on a held-out validation set $V = \{(x^v, y^v)\}$, and accepting the top-performing prompt candidates. Formally, with the evaluation function on LLM as $f_S(\cdot)$,

$$\begin{aligned} r_i^k &= f_S(p_{k_i}^c, V), i \in [1, \cdots, Q], \qquad (2)\\ p_k &= p_{k_j}^c, \text{where } j = \text{argmax}(\{r_1^k, \ldots, r_Q^k\}). \end{aligned}$$

where argmax$(\cdot)$ denotes the index of the maximum value. At the final optimization step, the top-performing prompt p_K will be accepted as the optimized prompt p^*.

3.2 Accelerated Prompt Optimization

Although current research on gradient-free prompt optimization can achieve significant performance gains on multiple tasks, demands for a great number of optimization steps hinder their practicability in real-world scenarios. For instance, **?** does not converge even after over 150 steps in some tasks; **?** finds a good solution in 50 to 75 steps. Therefore, we highlight the problem of accelerated prompt optimization, *i.e.,* obtaining p^* with satisfactory performance in few optimization steps, *e.g.,* $K < 5$.

4 Proposed Method

4.1 Motivation

We believe that two factors are crucial for achieving accelerated prompt optimization, which current gradient-free prompt optimization methods fail to achieve. Firstly, the initial prompt p_0 plays a crucial role in accelerating the prompt optimization process (**?**), where p_0 with better LLM performance makes the optimization towards better prompts easier, preventing LLMs from excessively exploring suboptimal prompt regions. This is generally overlooked by existing research that utilizes uninformative initial prompts, *e.g.,* (**?**). Therefore, we propose to devise high-quality p_0 by crafting a novel initial prompt schema. Furthermore, a more precise expansion and acceptance of prompt candidates ensure highly efficient optimization direction and fewer optimization steps. Current expansion and acceptance techniques optimize the prompt towards improving the general task performance, where effective optimization direction in each step is hard to ensure. To tackle this, we propose to utilize the past failure cases from previous optimization steps to further navigate the expansion and acceptance of prompt candidates. We illustrate our dual-phase approach as follows (*cf.* Figure **??**).

4.2 High-Quality Initial Prompt Generation

We think that a high-quality initial prompt that can elicit the desired response from LLMs should be able to provide clear task instruction and detailed task-related information. Specifically, it should 1) give a clear definition of the task type and provide a detailed task description, 2) define the output format and constraints, 3) provide insights on the reasoning processes and professional tips. To achieve such initial prompts, we are inspired by the step-back prompting (**?**) which demonstrates LLM's ability to derive high-level concepts and principles from examples. Thus, following (**?**), we design a meta-instruction I_m (*cf.* Figure **??**), leveraging LLM's ability to generate p_0 by observing the input-output exemplars of the target task $\mathcal{T}$ and inferring the above required information. Formally, defining input-output exemplars as $D = \{(x_d, y_d)\}$,

$$p_0 = LLM(I_m, D). \tag{3}$$

4.3 Experience-Tuned Optimization

In the optimization phase, it is necessary to tune the expansion and acceptance of prompt candidates to quickly improve the task performance as evaluated on the validation set V and thus reduce optimization steps. Inspired by previous research (**?**), we intend to make the best of past failure cases to generate promising prompt candidates and filter out unnecessary optimization attempts. In each optimization step, we maintain a failure case set $F_k = \{(x_k^f, y_k^f)\}$ containing the examples from V where the initial prompt p_{k-1} fails to predict the ground truth in the acceptance stage, *i.e.*, $\hat{y}_k^f \neq y_k^f$.

```
meta-instruction for initialization

You gave me an instruction on a certain task
and some example inputs with chain-of-thought.
I read the instruction carefully and wrote an
output with chain-of-thought for every input
correctly. Here are some correct input-output
pairs which strictly meet all your
requirements:

{example_pairs}


The instruction given contains the following
parts. Based on the input-output pairs
provided, give me the final complete
instruction in English without any explanation:

###Task type###
Task type: This is a <...> task.

###Task detailed description###
Task detailed description: <Task detailed
description>

###Your output must satisfy the following
format and constraints###
Output format(type): <Output format or its
type>
Output constraints: <constraints on output>

###You must follow the reasoning process###
<add several reasoning steps if it's
necessary>

###Tips###
<add several useful tips from a professional
point of view to accomplish this task better>
```

Figure 3: Meta-instruction used in our initialization phase to generate high-quality initial prompts.

Expansion. In the expansion stage, since the initial prompts are long prompts with at least four sentences, we aim to improve the expansion efficiency by segmenting them into individual sentences for sentence-level expansion following LongPO (**?**). Moreover, since different sentences in the initial prompts contain different task-related information and may have different impacts on the task performance, we devise sentence weights w^k to estimate the impact of each sentence on the performance improvement, which is updated leveraging the past failure cases. We first split the initial prompt p_0 into M sentences, and initialize the weight w^1 for each sentence as 1.

$$\begin{aligned} p_0 &= [s_1^1, s_2^1, ..., s_M^1], \\ w_t^1 &= 1, t \in [1, M]. \end{aligned} \tag{4}$$

In the k-th optimization step, we compute the acceptance probability Pr^k for each sentence:

$$\mathrm{Pr}_i^k = \frac{\exp(w_i^k)}{\sum_{j=1}^{M} \exp(w_j^k)}. \tag{5}$$

After that, we sample a sentence for expansion based on the probability distribution $\mathrm{Pr}^k = [\mathrm{Pr}_1^k, \cdots, \mathrm{Pr}_M^k]$, where the sampled sentence is denoted as $s_o^k, o \in [1, M]$. For expansion of s_o^k, we design a meta-instruction I_e (*cf.* Figure **??**) to instruct LLM to generate a revised sentence considering the past experience.

$$\hat{s}_o^k = LLM(I_e, p_{k-1}, F_k, s_o^k). \tag{6}$$

```
meta-instruction for optimization

I'm trying to write a zero-shot prompt which
consists of four parts.
My current prompt is:
[{prompt_to_revise}]

But it gets the following outputs that fail to
match the expected outputs:
{failed_cases}

The sentence I want to revise is:
{sentences[chosen_sentence]}

Comparing the wrong outputs with their
corresponding expected answers under the same
input, optimize the above sentence to help AI
understand the task more comprehensively and
accomplish this task better.
Your response format is as follows.
The given sentence
'{sentences[chosen_sentence]}' should be
revised as:
```

Figure 4: Meta-instruction used in the optimization phase.

Before passing $\hat{s}_o^k$ to the acceptance stage, we design additional strategies to further guarantee the effectiveness of the generated sentence leveraging F_k. Firstly, to ensure $\hat{s}_o^k$ can actually improve over s_o^k, we replace s_o^k in p_{k-1} with $\hat{s}_o^k$, denoted as $\hat{p}_k$, and evaluate whether $\hat{p}_k$ outperforms p_{k-1} on F_k. We accept $\hat{s}_o^k$ only when $\hat{p}_k$ has improved the performance over p_{k-1} larger than a threshold H_F.

$$f_S(\hat{p}_k, F_k) - f_S(p_{k-1}, F_k) > H_F. \qquad (7)$$

Besides, to avoid repeatedly generating the same ineffective $\hat{s}_o^k$, we build a collection $\mathcal{G}$ of undesired sentence revisions and check whether $\hat{s}_o^k$ has appeared in $\mathcal{G}$. If the above two criteria are not met, we abandon $\hat{s}_o^k$ and regenerate starting from Eq. **??**.

Acceptance. In addition to evaluating $\hat{p}_k$'s performance on the entire failure case F_k, we also evaluate its performance on the validation set V. We accept $\hat{p}_k$ as the next initial prompt p_k only when $\hat{p}_k$ has improved the performance over p_{k-1} larger than a threshold H_V. Otherwise, we abandon $\hat{p}^k$ and restart from sampling s_o^k.

$$f_S(\hat{p}_k, V) - f_S(p_{k-1}, V) > H_V. \qquad (8)$$

If $\hat{p}^k$ is accepted, we update its sentence weights. We calculate the mixed evaluation result $f_R(\cdot)$ and update the w^{k+1} as follows, where α and the learning rate η are adjusting hyperparameters.

$$f_R(\hat{p}_k) = \alpha f_S(\hat{p}_k, V) + (1-\alpha) f_S(\hat{p}_k, F_k). \qquad (9)$$

$$w_i^{k+1} = w_i^k \exp\left(\frac{\eta f_R(\hat{p}_k)}{\mathrm{Pr}_i^k M}\right).$$

When the number of times that Eq. **??** or Eq. **??** is not satisfied accumulates to 5, we consider the algorithm to have converged.

The weight formula is designed to adaptively update the importance of each sentence in the prompt based on its impact on overall performance improvement. $f_R(\hat{p}_k)$ modulates the magnitude of the weight adjustment: a higher $f_R(\hat{p}_k)$ leads to larger updates. Pr_i^k determines the weight's contribution, while M is used for normalization to ensure balanced weight updates. The learning rate η controls the extent of weight adjustments based on the evaluation feedback. Inspired by the EXP3 algorithm (**?**), these components facilitate a dynamic and adaptive optimization process, tuned by empirical performance data. The who process is summarized in Algorithm **??**.

Algorithm 1
Dual-Phase Accelerated Prompt Optimization

```
Require: Input-output exemplars $D$, validation set $V$, meta-instruction $I_m$ and $I_e$.
Ensure: Optimized prompt $p^*$
 1: Initialize $p_0$ (Eq. ??), derive failure case set $F_1$
 2: Split $p_0$ into $M$ sentences $[s_1^1, s_2^1, \ldots, s_M^1]$, initialize sentence weights $\{w_i^1\}_{i=1}^M \leftarrow 1$, $k \leftarrow 1$
 3: while not converged do
 4:     ▷ Expansion
 5:     Sample a sentence $s_o^k$ based on $\mathrm{Pr}^k$ (Eq. ??)
 6:     Generate revised sentence $\hat{s}_o^k$ (Eq. ??)
 7:     Replace $s_o^k$ in $p_{k-1}$ with $\hat{s}_o^k$ to get $\hat{p}_k$
 8:     if $\hat{s}_o^k \in \mathcal{G}$ or (Eq. ??) is not satisfied then
 9:         Add $\hat{s}_o^k$ to $\mathcal{G}$
10:         Regenerate $\hat{s}_o^k$ from line ??
11:     end if
12:     ▷ Acceptance
13:     if (Eq. ??) is not satisfied then
14:         Restart from line ??
15:     end if
16:     $p_k \leftarrow \hat{p}_k$, update $w_i^{k+1}$, $k \leftarrow k+1$
17:     Update $F_k$ with new failure cases
18: end while
19: return optimized prompt $p^* = p_k$
```

5 Experiments

In this section, we begin by detailing datasets, baselines, and the implementation of the experiments. Following this, we conduct comprehensive and controlled experiments on our method.

5.1 Experimental Settings

Datasets. Our experiments are first conducted on general natural language understanding tasks across four datasets to validate our method, specifically focusing on sentiment classification (SST-2 (**?**)), topic classification (AG's News (**?**), TREC (**?**)) and subjectivity classification (Subj (**?**)). Then we perform our approach to the challenging BBH tasks (**?**), which include manually provided few-shot Chain-of-Thought (CoT) prompts containing task descriptions and demonstrations.

Baselines. We compare our method with three popular prompt optimization methods for zero-shot black-box prompting and the well-crafted prompts manually provided in BBH tasks: **APO** (**?**): Generating natural language "gradients" to criticize and improve the current prompts. **APE** (**?**): Proposing both a naive and an iterative Monte Carlo search methods to approximate the solution to the prompt optimization problem. **PromptAgent** (**?**): Automating expert-level prompt generation by treating it as a strategic planning problem using Monte Carlo tree search and error feedback to refine and optimize prompts. **Manual Prompt** (**?**): The few-shot CoT version of human-designed prompts with teaching examples developed in BBH tasks.

Implementation Details. In line with (**?**), since BBH tasks lack an official train-test split, we shuffle the data and allocate approximately half for testing. The rest is used for training, prompt generation, and optimization. For datasets with predefined test sets, we use those directly.

Unless otherwise stated, we evaluate performance (*i.e.,* accuracy) on GPT-3.5-Turbo using the OpenAI API[1] (currently gpt-3.5-turbo-0125) in a zero-shot prompt setting. The temperature is set to 0 for prediction and 0.5 for prompt generation to enhance diversity. To accelerate prompt optimization, we limit the maximum optimization steps to **four** for all methods, while keeping other baseline parameters and settings at default. At the beginning of prompt initialization, eight exemplars are obtained by concatenating unique input-output pairs from the shuffled training data until the desired amount is reached, ensuring no duplicate inputs. Due to limited computational resources, our approach generates and optimizes only one initial prompt. By default, we set $H_F = 0.3$, $H_V = 0.1$, $\alpha = 0.4$, and $\eta = 0.055$ in Algorithm **??** to accelerate the optimization phase.

5.2 Main Results & Analysis

	Few-shot	**Zero-shot**			
Task	**Manual**	**APO**	**APE**	**PA**	**Ours**
SST-2	/	0.89	0.92	0.443	**0.978**
AG's News	/	0.88	0.819	0.785	**0.928**
TREC	/	**0.795**	0.513	0.687	0.785
Subj	/	0.64	0.593	0.494	**0.72**
Logical Five	0.388	0.392	0.404	0.443	**0.48**
Hyperbaton	0.744	0.808	0.865	0.823	**0.88**
Disambiguation	0.580	0.688	0.645	0.696	**0.74**
Salient Translation	0.544	0.456	0.538	0.468	**0.548**
Avg.	0.564	0.694	0.662	0.605	**0.757**

Table 1: Accuracy on eight tasks on GPT-3.5-Turbo. PA indicates PromptAgent. Bold and underlined text indicate the best and second-best results, respectively.

Overall Results. Table **??** demonstrates the effectiveness of our accelerated dual-phase approach across 8 NLP tasks compared to classic prompt optimization methods. Our method significantly outperforms all baselines, achieving an average improvement of approximately **10.7%** over APO, **16.4%** over APE, and **29.7%** over PromptAgent across the given tasks.

Our method also surpasses few-shot CoT human-crafted prompts with an approximately **17.6%** average improvement on selected BBH tasks, indicating its ability to produce high-quality prompts that enhance the black-box LLM's capabilities in logical deduction, grammar, language understanding, and multilingual tasks without teaching examples.

Analysis. To understand this result, we analyzed the prompt expansion and acceptance processes: In prompt expansion, our method leverages past experience, filters out unnecessary optimization attempts, and collects undesired revisions. This contrasts with baseline methods that inefficiently explore prompt space and underutilize past iterations. APE lacks reflection on past iterations, slowing its Monte Carlo-based search. APO uses error feedback to guide beam search but is slowed by evaluating many paths. PromptAgent's Monte Carlo

[1] https://chat.openai.com/

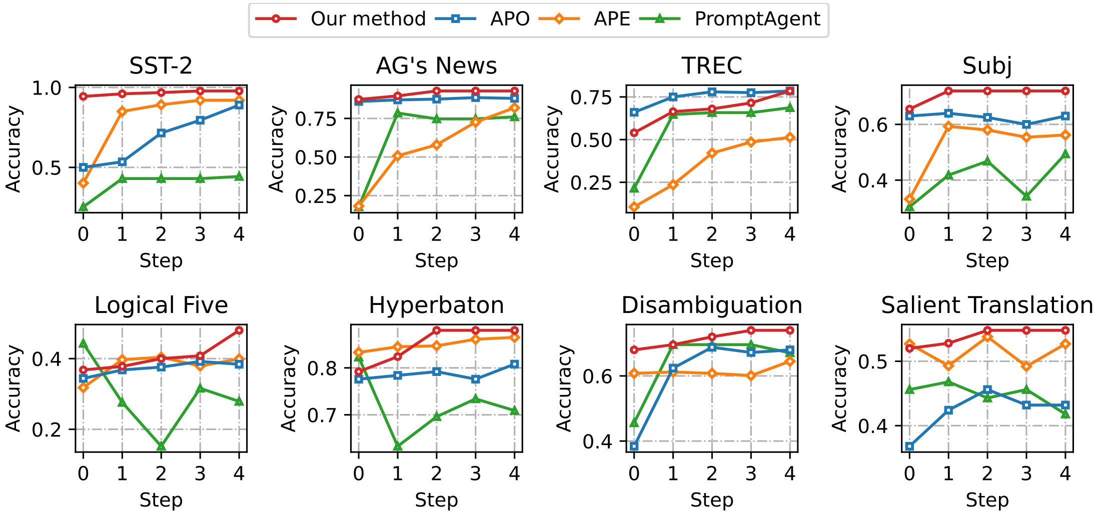

Figure 5: Performance (accuracy) over 4 steps across 8 tasks on GPT-3.5-Turbo.

Search Tree explores prompt optimization through simulations, but limited steps lead to suboptimal results.

In the acceptance process, inspired by the EXP3 algorithm, our method uses weighted sentences and modifications to enhance prompt quality, making it superior in identifying promising candidates and optimizing directions.

Convergence Analysis. To evaluate our method's convergence within four steps compared to others, we examine how quickly each method achieves peak performance across datasets. Figure **??** shows the performance (accuracy) variation of four prompt optimization methods across eight datasets, with each subfigure representing a different dataset. While APO, APE, and PromptAgent experience fluctuations or plateau at lower accuracy, our method demonstrates the fastest convergence across most datasets, often reaching near-peak performance within the first two steps. This rapid convergence highlights our method's efficiency in optimizing prompts quickly and effectively, making it promising for tasks requiring prompt optimization within a few steps.

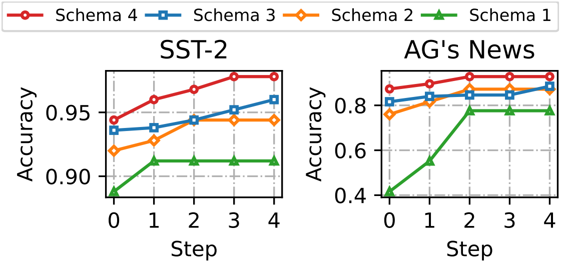

Figure 6: Results on GPT-3.5-Turbo with different initial prompt schemas.

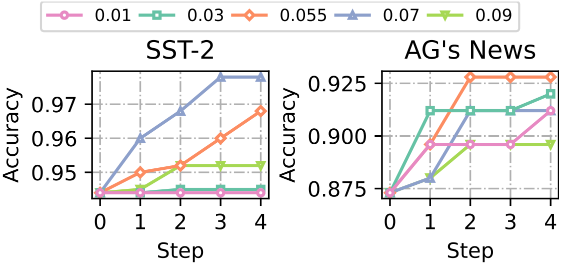

Figure 7: Results on GPT-3.5-Turbo with different optimization learning rates.

5.3 Ablation Study

We conduct several ablation experiments to assess the efficacy of our method.

5.3.1 Different Initial Prompt Schemas

Our method uses a meta-instruction to generate a prompt with four components: a) task type and description, b) output format and constraints, c) suggested reasoning process, and d) professional tips. We define: *Schema 4*: All four components *Schema 3*: First three components *Schema 2*: First two components *Schema 1*: Task type and description only (common in current techniques). We vary the meta-instructions for these schemas and conduct four-step prompt optimization experiments on SST-2 and AG's News to assess their impact on optimization.

As shown in Figure **??**, initial prompts from Schema 4 yield the highest evaluation results. In contrast, Schema 1 has the lowest metrics and often falls into suboptimal local minima, a common issue with current methods. This comparison validates our meta-instruction design and underscores that

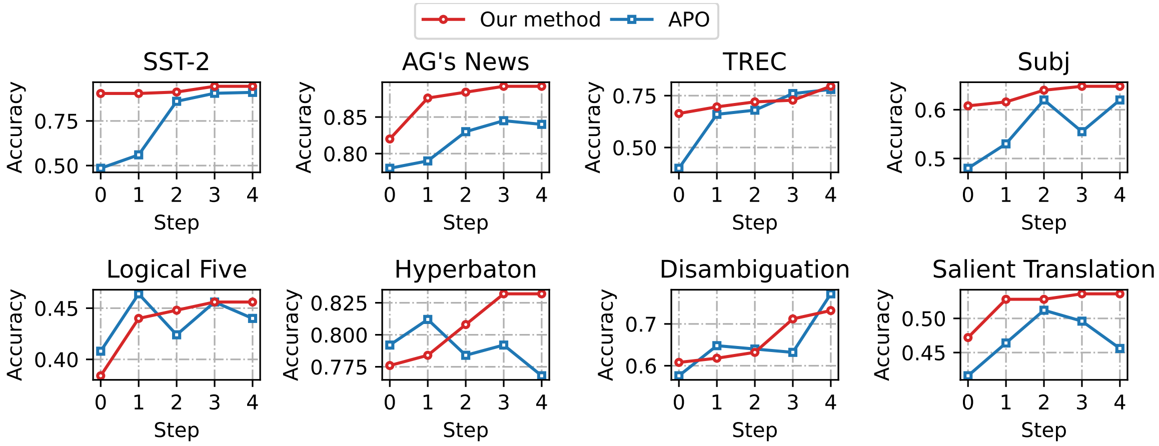

Figure 8: Accuracy over 4 steps across 8 tasks on Baichuan2-Turbo.

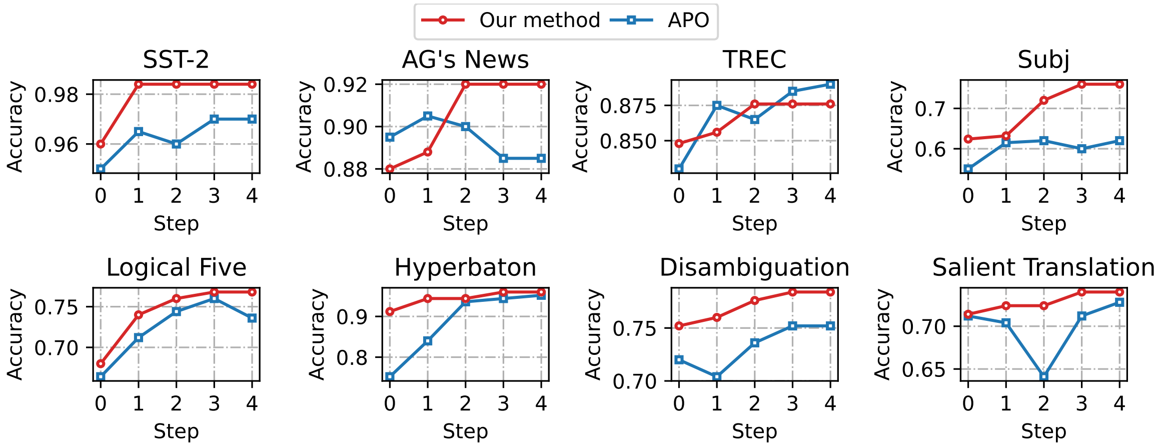

Figure 9: Accuracy over 4 steps across 8 tasks on GPT-4.

a high-quality initial prompt is crucial for quickly identifying the optimal prompt.

5.3.2 Sensitivity to Learning Rate

During the optimization phase, the learning rate η controls the extent of sentence weight updates after each round. A higher η results in significant updates and responsiveness to recent performance changes, while a lower η promotes stability with gradual adjustments. This balance is crucial for navigating the trade-off between exploration and exploitation.

We conduct prompt optimization experiments on SST-2 and AG's News within four steps, testing η values from 0.01 to 0.1. As shown in Figure **??**, $\eta = 0.055$ and $\eta = 0.07$ are the most and second most effective in accelerating optimization.

5.3.3 Performance on Different LLMs

As Table **??** indicates, APO is the best baseline method. Therefore, we compare our method with APO using Baichuan2 (**?**) and GPT-4 accessed via the APIs. We conduct prompt optimization experiments on eight NLP datasets across four optimization steps.

Figure **??** and **??** illustrate the performance variation of both methods across different datasets as optimization steps progress. APO fails to converge within four steps and shows greater performance volatility compared to Baichuan2-Turbo and GPT-4. In contrast, our method demonstrates rapid convergence and strong optimization acceleration. Except for the generalizability to other models, we also find that stronger LLM can achieve more effective prompt optimization with our method.

5.3.4 Performance on Specialized Domain-Specific Task

To evaluate our method performance on specialized tasks that require domain knowledge, we conduct experiments on the Geometric Shapes task (**?**), which involves interpreting SVG paths to determine the geometric figures they represent, a task that requires specific domain knowledge.

Model	APO	Ours
GPT-3.5-Turbo	0.36	0.392
GPT-4	0.448	0.488

Table 2: Accuracy on Geometric Shapes task on GPT-3.5-Turbo and GPT-4.

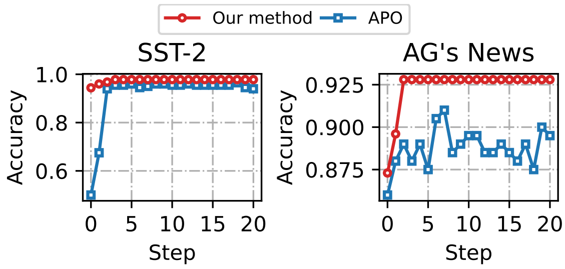

Figure 10: Accuracy over 20 steps on GPT-3.5-Turbo.

As shown in Table **??**, our approach demonstrates consistent performance improvement over the best baseline APO, revealing the effectiveness of our method in specialized task.

5.3.5 Results without Step Constraint

We report the results of prompt optimization with a maximum of 20 steps on two general NLU tasks. As shown in Figure **??**, the strongest baseline, APO, converges on the SST-2 task with slightly lower accuracy than our method. However, on the AG's News task, APO's performance fluctuates significantly and lags behind our method. Thus, our method demonstrates superior performance and faster convergence compared to existing methods, even with fewer optimization steps.

5.3.6 Computational Complexity

Since the running time is related to the number of API calls and may be affected by the network condition, we mainly present the number of API calls, which is an important metric for cost comparison on black-box LLMs.

We conduct our experiments with GPT-4 on nine tasks. As shown in Table **??**, our method requires approximately 1/7 of the number of API calls compared to the strongest baseline method, APO.

6 Conclusion

In this paper, we addressed the issue of low convergence rates in gradient-free prompt optimization methods for LLMs. Our proposed dual-phase approach effectively accelerates prompt optimization by generating high-quality initial prompts and leveraging tuning experience to navigate the optimization process. Extensive experiments on several LLMs across diverse datasets demonstrated the superiority of our method in achieving satisfactory performance within few optimization steps. Our approach not only enhances the efficiency of prompt optimization but also improves the overall performance of LLMs in various NLP tasks. Future work will focus on further refining the optimization strategies and exploring their applications in more diverse and complex scenarios.

Task	APO	Ours
SST-2	12,520	1,708
AG's News	12,733	2,089
TREC	9,739	1,486
Subj	12,790	1,848
Logical Five	9,631	1,512
Hyperbaton	9,934	1,626
Disambiguation	9,471	1,187
Salient Translation	10,190	1,451
Geometric Shapes	9,648	1,496
Avg.	10,739	1,600

Table 3: API calls consumed on nine tasks on GPT-4.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (62272437,62402470), the University Synergy Innovation Program of Anhui Province (GXXT-2023-071), and the Fundamental Research Funds for the Central Universities of China (WK2100000053, PA2024GDSK0107). This research is also supported by the advanced computing resources provided by the Supercomputing Center of the USTC.

Limitations

We acknowledge some limitations despite the promising results of our research that could pave the way for future studies:

1) Our experiments were limited to general NLP tasks and one domain-specific task, more performance assessment on specialized tasks remains to be included. 2) Our method relies on labeled task data for prompt generation and evaluation, raising concerns about its robustness in personalized or scenarios lacking labeled data. 3) Our experiments were confined to GPT-3.5-Turbo, Baichuan2-Turbo and GPT-4, leaving the effectiveness of our method on other large language models to be validated in future studies.

Further study may be needed to address these limitations so as to improve the generalizability and robustness of our approach in broader and more complex real-world applications.

References

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 1995. Gambling in a rigged casino: The adversarial multi-arm bandit problem. In *IEEE Annual Symposium on Foundations of Computer Science*.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2023. Instructzero: Efficient instruction optimization for black-box large language models. *ArXiv*, abs/2306.03082.

Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, LIN Yong, Xiao Zhou, and Tong Zhang. 2023a. Black-box prompt learning for pre-trained language models. *Transactions on Machine Learning Research*.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023b. Active prompting with chain-of-thought for large language models. *ArXiv*, abs/2302.12246.

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

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *ArXiv*, abs/2209.12356.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, Yujiu Yang, Tsinghua University, and Microsoft Research. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *ArXiv*, abs/2309.08532.

Cho-Jui Hsieh, Si Si, Felix X. Yu, and Inderjit S. Dhillon. 2023. Automatic engineering of long prompts. *ArXiv*, abs/2311.10117.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, and Kian Hsiang Low. 2024. Prompt optimization with human feedback. *ArXiv*, abs/2405.17346.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. 2023. Use your instinct: Instruction optimization using neural bandits coupled with transformers. *ArXiv*, abs/2310.02905.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55:1 – 35.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*.

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

Rui Pan, Shuo Xing, Shizhe Diao, Xiang Liu, Kashun Shum, Jipeng Zhang, and Tong Zhang. 2023. Plum: Prompt learning using metaheuristic. *ArXiv*, abs/2311.08364.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *ArXiv*, cs.CL/0409058.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2022. Grips: Gradient-free, edit-based instruction search for prompting large language models. *ArXiv*, abs/2203.07281.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Conference on Empirical Methods in Natural Language Processing*.

Libo Qin, Qiguang Chen, Xiachong Feng, Yang Wu, Yongheng Zhang, Yinghui Li, Min Li, Wanxiang Che, and Philip S. Yu. 2024. Large language models meet nlp: A survey. *ArXiv*, abs/2405.12819.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Eliciting knowledge from language models using automatically generated prompts. *ArXiv*, abs/2010.15980.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing*.

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Huai hsin Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Annual Meeting of the Association for Computational Linguistics*.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. 2023. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *ArXiv*, abs/2310.16427.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *ArXiv*, abs/2302.11382.

Ai Ming Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Hai Zhao, Hang Xu, Hao-Lun Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kuncheng Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Pei Guo, Ruiyang Sun, Zhang Tao, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yan-Bin Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023a. Baichuan 2: Open large-scale language models. *ArXiv*, abs/2309.10305.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023b. Large language models as optimizers. *ArXiv*, abs/2309.03409.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. 2023. Prompt engineering a prompt engineer. *ArXiv*, abs/2311.05661.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023a. Extractive summarization via chatgpt for faithful summary generation. In *Conference on Empirical Methods in Natural Language Processing*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Neural Information Processing Systems*.

Zhihan Zhang, Shuo Wang, W. Yu, Yichong Xu, Dan Iter, Qingkai Zeng, Yang Liu, Chenguang Zhu, and Meng Jiang. 2023b. Auto-instruct: Automatic instruction generation and ranking for black-box language models. In *Conference on Empirical Methods in Natural Language Processing*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed Huai hsin Chi, Quoc V. Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *ArXiv*, abs/2310.06117.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *ArXiv*, abs/2211.01910.