Unleashing the Potential of Large Language Models as Prompt Optimizers: Analogical Analysis with Gradient-based Model Optimizers

Xinyu Tang[1*], **Xiaolei Wang**[1*], **Wayne Xin Zhao**[1†], **Siyuan Lu**[2], **Yaliang Li**[3], **Ji-Rong Wen**[1]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]International School, Beijing University of Posts and Telecommunications
[3]Alibaba Group
txy20010310@163.com, wxl1999@foxmail.com, batmanfly@gmail.com

Abstract

Automatic prompt optimization is an important approach to improving the performance of large language models (LLMs). Recent research demonstrates the potential of using LLMs as prompt optimizers, which can generate improved task prompts via iterative refinement. In this paper, we propose a novel perspective to investigate the design of LLM-based prompt optimizers, by drawing an analogy with gradient-based model optimizers. To connect these two approaches, we identify two pivotal factors in model parameter learning: *update direction* and *update method*. By systematically analyzing a rich set of improvement strategies on the two aspects, we further develop a capable **G**radient-inspired LLM-based **P**rompt **O**ptimizer called **GPO**. At each step, it first retrieves relevant prompts from the optimization trajectory as the update direction. Then, it utilizes the generation-based refinement strategy to perform the update, while controlling the edit distance through a cosine-based decay strategy. Extensive experiments demonstrate the effectiveness and efficiency of GPO. In particular, GPO brings an additional improvement of up to 56.8% on Big-Bench Hard and 62.6% on MMLU compared to baseline methods.

Introduction

Nowadays, prompting has become the pivotal approach to unleashing the power of large language models (LLMs) (Zhao et al. 2023). However, prompt engineering is not easy and requires extensive trial-and-error efforts since LLMs are sensitive to prompts (Lu et al. 2022; Wang et al. 2024; Tang et al. 2024). Although general guidelines for high-quality prompts exist (Kojima et al. 2022; Amatriain 2024), they cannot always lead to optimal task performance.

To improve the task performance of LLMs, *automatic prompt optimization* has been proposed (Zhou et al. 2023). Early work either performs discrete optimization through methods like reinforcement learning (Zhang et al. 2023) or performs continuous optimization in the embedding space (Chen et al. 2023; Wang et al. 2022a, 2023). However, they often require access to the logits or internal states of LLMs, which is infeasible for those only accessible through APIs. In addition, they need to be specially trained for each task. Considering these issues, recent work proposes to model the optimization problem in natural language and using LLMs as *prompt optimizers* due to their strong understanding (Zhan et al. 2024) and generation capabilities (Schnabel and Neville 2024). In this approach, LLMs perform optimization with only outputs from the task model and quickly adapt to various tasks without training. However, such a method raises a new challenge for the design of *meta-prompt*, which is the prompt for LLMs to perform prompt optimization.

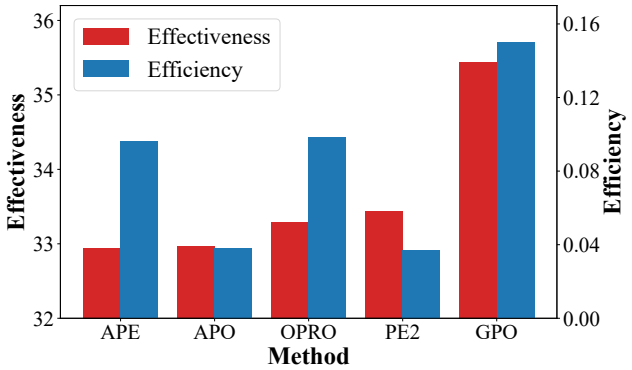

Figure 1: Comparisons of GPO to existing LLM-based prompt optimizers in terms of effectiveness (Accuracy) and efficiency (improvement per dollar spent on API) on BBH.

To tackle this issue, we aim to investigate the design of meta-prompts. Existing methods for creating meta-prompts typically involve either manual human effort (Yang et al. 2023) or heuristic algorithms (Fernando et al. 2023). Despite the flexibility, these studies still lack principled guidelines about their designs. Our work is inspired by the great success of gradient-based optimizers in model optimization, which have been systemically studied in both theory and practice (Sun et al. 2020). Since both optimizers aim at enhancing model performance through iterative optimization, it is feasible to connect the two different approaches via analogical analysis. In this way, we can borrow the theoretical framework and extensive research of gradient-based model optimizers to enhance LLM-based prompt optimizers.

Therefore, in this paper, we propose a comprehensive analogy framework for the two key factors (*i.e., update direction*

[*] Equal contribution.
[†] Corresponding author.

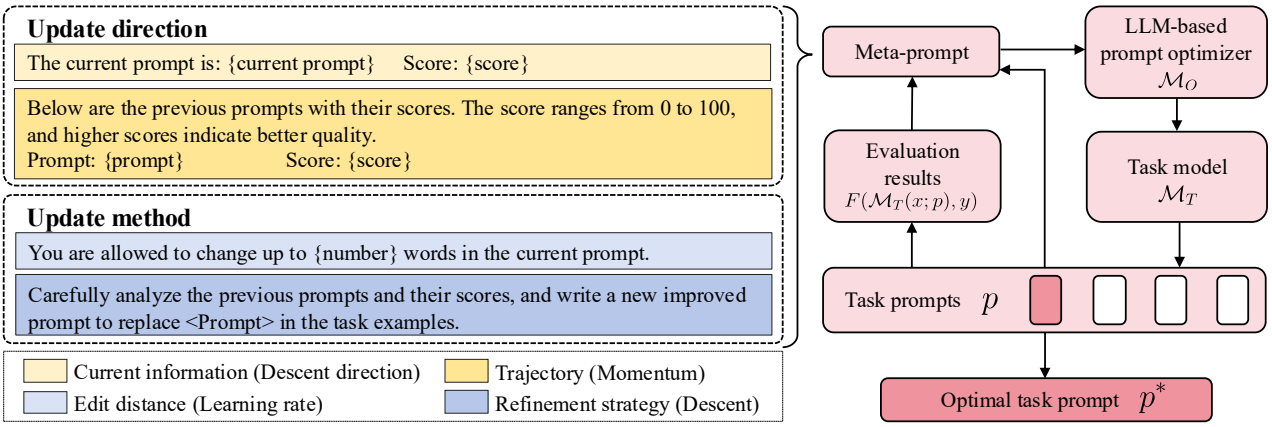

Figure 2: The overview of the GPO framework. "Current information", "Trajectory", "Edit distance", and "Refinement strategy" are concepts of LLM-based prompt optimizers, which can correspond to "Descent direction", "Momentum", "Learning rate", and "Descent" in gradient-based model optimizers.

and *update method*) in LLM-based optimizers. As illustrated in Table 1, for update direction, we analogize descent direction to the information of the current prompt and momentum to the information of the historical prompt, while for the update method, we analogize the learning rate to edit distance and gradient descent to the refinement strategy. Based on such a framework, we conduct systematic empirical studies on various implementations for each component and report their experimental results with detailed analysis.

Based on the findings from our systematic analysis, we further develop a capable **G**radient-inspired LLM-based **P**rompt **O**ptimizer called **GPO**, with the best implementation for each component. Figure 2 illustrates the overall framework of GPO. We evaluate its effectiveness across various LLMs, tasks, and evaluation settings. When using `Llama-2-7b-chat` as the task model, the prompts produced by GPO surpass the instruction "Let's think step by step" by 18.5% on Big-Bench Hard (BBH) and 7.6% on MMLU. Furthermore, GPO produces an additional improvement of up to 56.8% on BBH and 62.6% on MMLU compared with baseline methods while using fewer tokens.

Our contributions can be summarized as follows:

- To the best of our knowledge, this is the first time that a systematic analogy has been conducted for LLM-based prompt optimizers with gradient-based model optimizers.
- We conduct a comprehensive experimental analysis on the two key factors (*i.e.,* update direction and update method) and report several key findings.
- Based on the findings of the systematic analysis, we develop a more effective and efficient LLM-based prompt optimizer, GPO, which surpasses competitive baseline methods across various LLMs, tasks, and evaluation settings while incurring lower costs.

Related Work

Prompt Engineering and Optimization. Prompt engineering aims to find suitable prompts for LLMs to perform various tasks. To reduce human efforts, researchers have explored automatic prompt optimization, which can be categorized into continuous and discrete optimization methods. Discrete methods directly optimize the natural language prompts through methods like reinforcement learning (Deng et al. 2022; Zhang et al. 2023) and editing (Xu et al. 2022; Prasad et al. 2023). In contrast, continuous methods perform optimization in the embedding space of LLMs, allowing for optimization through gradient (Li and Liang 2021). We focus on discrete methods, especially LLM-based prompt optimizers.

LLM-based Prompt Optimizers. Due to the unprecedented capabilities of LLMs, recent work starts to utilize them as prompt optimizers. One line of work (Guo et al. 2023; Yang and Li 2023) combines LLMs with evolutionary algorithms to perform prompt optimization. Another line of work (Yang et al. 2023) aims to adapt concepts and techniques from gradient-based model optimizers (*e.g.,* gradient (Pryzant et al. 2023) and momentum (Yang et al. 2023)) to LLM-based prompt optimizers. However, no comprehensive guidelines exist for using LLMs as prompt optimizers. We aim to tackle this with a systematic investigation, which is conducted by analogy with gradient-based model optimizers.

Analogical Analysis

In this section, we present an analogical analysis between model optimization and prompt optimization to build connections and improve existing LLM-based prompt optimizers.

Background

In this part, we first introduce the task definition of LLM-based prompt optimization, and then establish connections with gradient-based optimizers.

Task Formulation. Prompt optimization aims to find the optimal *task prompt* p^* in the format of natural language that maximizes the performance on a specific task dataset $\mathcal{D}$ when using an LLM as the task model $\mathcal{M}_T$. To perform such optimization, our idea is to develop a prompt optimizer, which can be built upon some search algorithm (*e.g.,* evolutionary

Factor	Gradient-based model optimizer	LLM-based prompt optimizer
Update direction	Descent direction Momentum	Current information Trajectory
Update method	Learning rate Descent	Edit distance Refinement strategy

Table 1: Analogy between glossaries in model optimizer and prompt optimizer.

algorithms (Guo et al. 2023)) or an LLM (Yang et al. 2023). In this paper, we focus on using an LLM as the prompt optimizer $\mathcal{M}_O$. Formally, the problem of prompt optimization can be formulated as:

$$p^* = \arg\max_{p \sim \mathcal{M}_O} \mathbb{E}_{\langle x,y \rangle \in \mathcal{D}} \left[F(\mathcal{M}_T(x;p), y) \right], \quad (1)$$

where p is the prompt generated by the prompt optimizer $\mathcal{M}_O$, $\mathcal{M}_T(x;p)$ represents the output from the task model for input x conditioned on the prompt p, and the function $F(\cdot)$ calculates the task performance based on some measurement.

Revisiting Gradient-based Optimizers. Similar to LLM-based prompt optimizer, gradient-based model optimizer aims to find the optimal values of model parameters that minimize the loss function. In the basic form of gradient descent (Boyd and Vandenberghe 2014), a single optimization step can be formulated as follows:

$$\Theta_{k+1} = \Theta_k - \tau_k g_k, \quad (2)$$

where Θ_k and Θ_{k+1} are the values of model parameters at the last and current steps, τ_k and g_k are the learning rate and gradient at the current step. Gradient descent can be improved by focusing on two elements in the formula: τ_k and g_k. For τ_k, learning rate schedulers (Gotmare et al. 2019) are proposed to dynamically adjust the learning rate. For g_k, the concept of momentum (Sutskever et al. 2013) is introduced to include historical gradients, and its computation can be expressed as follows: $v_{k+1} = \beta v_k + g_k = \sum_{i=1}^{k} \beta^{k-i} g_i$, where β represents the momentum coefficient.

Despite various gradient-based optimizers, they mainly model two key factors, namely *update direction* (*e.g.,* gradient g_k) and *update method* (*e.g.,* subtract $\tau_k g_k$). Our approach is inspired by the observation that existing LLM-based prompt optimization methods also implicitly employ the two aspects (see Table 1). However, existing work only initially explores the design of the two key factors, we aim to conduct more in-depth and systematic investigations from these two novel perspectives in the following subsection.

Update Direction

The update direction refers to the adjustments based on the information of previous or current prompts to determine the best direction for improving them. We apply the descent direction and momentum concepts to design the meta-prompts.

Analogical Descent Direction. Descent direction determines the direction of parameter updates based on the model performance. We analogize two similar forms that determine how to improve new prompts according to the current information.

• *Prompt+Performance.* One straightforward method is to include the last-round task prompt and the corresponding model performance into the meta-prompt for the LLM-based optimizer $\mathcal{M}_O$. It leverages the capacity of LLMs to reason about how to improve prompting optimization.

• *Prompt+Performance+Reflection.* Another way to solve the barrier of the descent direction is to leverage the reflection capability of LLMs (Pryzant et al. 2023). With the reflection mechanism, LLMs can generate feedback from past failures to improve performance. Such feedback can be seen as a form of "semantic" gradient signals.

Analogical Momentum. Inspired by the momentum in gradient descent, we aim to make LLMs aware of the prompts used in the optimization process and their corresponding results (*i.e.,* descent direction), thereby achieving a more global update direction. A straightforward way is to directly include all intermediate results into the meta-prompt. However, it might be limited by the context length of LLMs and affected by the accumulated noise. To better utilize the optimization trajectory, we propose two alternative methods.

• *Summarization-based trajectory.* One simple approach is to summarize the intermediate results from the optimization trajectory. Specifically, at each step, we use an instruction (*i.e.,* "Your task is to summarize the problems in the previous prompt and the current prompt.") to let the LLM perform summarization using the summary in the last step and the result in the current step.

• *Retrieval-based trajectory.* Another approach is to select k pieces of intermediate results from the optimization trajectory. Specifically, we consider three strategies: (1) Recency: selecting k nearest intermediate results; (2) Relevance: selecting k most relevant intermediate results, which are measured by the semantic similarity based on the BGE model (Xiao et al. 2023); (3) Importance: selecting k most important intermediate results, which are measured by the performance.

Update Method

The update method can refer to how the LLM utilizes such information to improve task prompts. Accordingly, we explore how to analogize the learning rate and the descent process into the design of meta-prompts.

Analogical Learning Rate. In the model optimization, the learning rate controls the extent of gradient updates at each step. Similarly, we aim to control the variation degree of prompt optimization. Specifically, we limit the number of words that can be modified in the meta-prompt (*i.e., edit distance*). Accordingly, we propose two methods of controlling edit distance as follows:

• *Decay-based constraint.* To avoid overshooting the optimal solution, the decay strategy is proposed to gradually reduce the learning rate (Izmailov et al. 2018). Here, we borrow the idea of controlling the maximum edit distance and consider gradually reducing its value following either a linear

Prompt Optimizer		GPT-3.5-turbo		GPT-4		GPT-4	
Task Model		Llama-2-7b-chat		Llama-2-7b-chat		GPT-3.5-turbo	
Gradient		P+M	P+M+R	P+M	P+M+R	P+M	P+M+R
Momentum	None	41.07	40.34	40.32	39.56	64.76	64.55
	Summarization	41.03	40.63	40.58	40.41	64.62	64.55
	Recency	41.93	41.55	42.02	41.34	65.26	64.97
	Relevance	**42.53**	40.81	**42.89**	41.97	**65.87**	65.39
	Importance	41.84	39.73	41.73	41.04	65.26	65.11

Table 2: The performance comparison of various update directions based on different prompt optimizers and task models. "P" denotes prompt, "M" denotes performance, and "R" denotes reflection.

or cosine curve. In particular, we reduce the constraint to approximately 20% of its maximum value until convergence.

• *Warmup-based constraint.* To avoid instability in the early stage of optimization, the warmup strategy is proposed to gradually increase the learning rate at the beginning (Goyal et al. 2017). Similarly, we adopt the widely used linear warmup schedule to gradually increase the constraint for the maximum edit distance in the initial 5% steps.

Analogical Gradient Descent. By analogy with the subtraction operation in gradient descent (*i.e.,* $-\tau_k g_k$ in Eq. (2)), we introduce two methods to update the task prompt accordingly.

• *Editing-based refinement.* The first method directly edits the last-round task prompt to improve performance. Specifically, we add an instruction (*i.e.,* "Modify the current prompt") into the meta-prompt, which requires the LLM to edit the task prompt from the last step according to the update direction. This method allows for effectively exploiting the existing prompt.

• *Generation-based refinement.* In addition to direct edits, we can also leverage the in-context learning capability of LLMs to generate refined task prompts. Specifically, we present the information regarding the updated direction in the format of demonstrations. Then, We add an instruction (*i.e.,* "Write a new improved prompt") to let the LLM follow the demonstration to directly generate a new task prompt. Compared with the editing-based method, the generation-based approach explores a wider range of prompt variations.

Empirical Experiments

In this part, we describe the experiment setting for our analogical analysis and detail our findings from the experiment.

Experiment Setup. We select a dataset from each type of task in BBH (Suzgun et al. 2023) to create a lite BBH benchmark for our analysis: i) Navigate (binary choice); ii) Movie Recommendation (multiple choice); iii) Object Counting (numeric response); iv) Word Sorting (free response). We adopt exact match as the metric for performance evaluation. We use three different model combinations of prompt optimizers and task models (*i.e.,* `GPT-3.5-turbo` and `Llama-2-7b-chat`, `GPT-4` and `Llama-2-7b-chat`, `GPT-4` and `GPT-3.5-turbo`). The optimization process lasts for at most 3 epochs, under which the task prompt can reach the plateau.

Findings for Update Direction. The results for the update direction are presented in Table 2. Here are the main findings:

• *Reflection Leads to Performance Drop.* Regarding the analogy to descent direction, *prompt+performance* achieves better performance than *prompt+performance+reflection*, which brings an improvement of up to 31% compared with the initial task prompt. The improvement brought by prompts can be attributed to their rich semantic information about the task, which can activate the task-specific knowledge of LLMs for optimization. In contrast, LLMs are limited in their capabilities of reflection (Huang et al. 2023), which may lead to inaccurate updates.

• *Prompt Optimizers can Learn More from Contextually Relevant Prompts.* The analogical concept of momentum can further improve the performance. Among various designs, *relevance-based trajectory* emerges as the most effective one, which brings an additional 15% improvement. This can be attributed to the fact that LLMs can learn more from contextually relevant prompts, while it might be challenging for LLMs to fully understand the signal of recency or importance. By contrast, the summarization-based trajectory proves to be less helpful. One possible reason is that summarization only captures common aspects of the trajectory while neglecting details that may be crucial.

Findings for Update Method. To investigate the update method for prompt optimization, we conduct experiments using the best configuration found in the previous experiments. The results for the update method are presented in Table 3.

• *Generation-based Refinement is Better.* In general, *generation-based refinement* outperforms *editing-based refinement*, which brings an improvement of up to 36% compared with the initial task prompt. This can be partly attributed to the significance of exploration in prompt optimization. Generation-based strategy is not confined to the current prompt, allowing the model to better leverage the in-context examples. Therefore, this approach can demonstrate great flexibility to enable LLMs to explore a larger search space.

• *Decay Strategy is Helpful.* Among various controlling methods for prompt variation, *cosine decay-based constraint*

Prompt Optimizer		GPT-3.5-turbo		GPT-4		GPT-4	
Task Model		Llama-2-7b-chat		Llama-2-7b-chat		GPT-3.5-turbo	
Learning rate		Editing	Generation	Editing	Generation	Editing	Generation
Descent	No control	42.53	42.61	42.89	43.17	65.87	66.37
	Fixed	42.91	43.09	43.38	43.66	66.48	66.91
	+Warmup	41.76	40.08	42.53	42.95	65.79	65.52
	Linear decay	42.68	42.86	43.55	44.03	66.56	67.10
	+Warmup	41.47	41.12	41.47	42.91	66.03	66.18
	Cosine decay	42.95	**43.75**	43.98	**44.97**	66.74	**67.80**
	+Warmup	40.19	41.29	42.68	43.13	65.94	66.37

Table 3: The performance comparison of various update methods based on different prompt optimizers and task models.

Prompt optimizer	Update direction		Update method	
	Currect information	Trajectory	Edit distance	Refinement strategy
APE	P	None	None	Generation
APO	P+R	None	None	Editing
OPRO	P+M	Recency	None	Generation
PE2	P+M+R	Recency	Fixed	Generation
GPO	P+M	Relevance	Decay	Generation

Table 4: Comparisons of GPO with existing LLM-based prompt optimizers. "P" refers to prompt, "M" refers to performance, and "R" refers to reflection.

achieves the best performance, bringing an additional 10% improvement. However, unlike gradient-based model optimization, the warmup strategy does not yield improvement. This finding suggests that, at the early stage of prompt optimization, exploration plays a crucial role in conducting large-scale prompt searches, while stability becomes more important in the later stage for more refined adjustments.

Method

In this section, we present our novel gradient-inspired LLM-based prompt optimizer called **GPO**, which leverages the insights gained from our systematic study. GPO adopts an iterative prompt optimization framework. At each step, it first retrieves relevant prompts from the optimization trajectory as the update direction. Then, it utilizes the generation-based refinement strategy to perform the update, while controlling the edit distance through a cosine-based decay strategy.

Iterative Prompt Optimization. GPO performs prompt optimization through a multi-step iterative process. At each step, it first generates multiple candidate task prompts based on a meta-prompt. The meta-prompt serves as the input that guides the LLM in optimizing its prompts. Subsequently, we select the task prompt with the best performance for the next iteration. This iterative process continues until either the performance of the task prompt reaches a plateau or the predefined maximum number of optimization steps is reached.

The Design of Meta-Prompt. As the input to the LLM, our meta-prompt consists of two key components: update direction and update method.

• *Update direction.* To determine the update direction, we leverage the retrieval-based optimization trajectory in Section . This trajectory consists of past task prompts, along with their model performance. They are selected using a *relevance-based strategy* and sorted in ascending order based on their similarity to the current prompt.

• *Update method.* After the update direction is determined, we further employ the *generation-based refinement* strategy to update the task prompt. Specifically, we present the trajectory in the format of demonstrations in the meta-prompt. Then, the LLM can follow these demonstrations to generate a new task prompt via in-context learning. Additionally, we implement the *cosine-based decay strategy* to control the edit distance between task prompts at consecutive iterations, ensuring gradual and controllable changes.

Furthermore, we enhance the meta-prompt by incorporating a few task examples. These examples provide additional context to aid the LLM in understanding the task effectively.

Comparison of LLM-Based Prompt Optimizers. Existing LLM-based prompt optimizers can be divided into two main classes by their update direction. Some work (*i.e.,* APO (Pryzant et al. 2023) and PE2 (Ye et al. 2023)) leverages the reflection capability of LLMs to produce textual "gradients" as the update direction, while others (*i.e.,* OPRO (Yang et al. 2023) and APE (Zhou et al. 2023)) directly use task prompts as the update direction. Our approach is based on the systematic investigation of the update direction and the update method. In particular, we propose several novel designs for the meta-prompt: relevance-based trajectory as the update direction and decay-based constraint for edit distance in the update method. Table 4 presents a detailed comparison.

Experiments

In this section, we first set up the experiments and then report the results and detailed analysis.

Task	Complex reasoning task		Knowledge intensive task					Common NLP task	
Dataset	BBH	GSM8K	MMLU					WSC	WebNLG
			STEM	Human.	Social.	Other	Average		
Empty	30.51	22.00	31.05	36.54	41.75	37.20	35.96	60.67	32.14
CoT	29.91	24.00	32.53	37.05	41.05	36.94	36.36	59.33	31.11
SGDM	33.30	27.33	32.88	38.36	41.88	38.02	37.20	64.00	38.01
APE	32.94	25.00	33.51	38.69	42.02	37.96	37.50	62.00	36.49
APO	32.97	25.33	33.17	37.94	44.94	38.23	37.71	62.00	34.92
OPRO	33.29	26.67	34.76	38.72	43.55	37.11	38.05	63.33	37.89
PE2	33.43	25.33	33.77	37.95	44.80	38.25	38.07	62.67	39.10
GPO	**35.43**	**28.33**	**35.00**	**38.84**	**46.61**	**38.60**	**39.14**	**65.33**	**42.51**

Table 5: Performance comparison using only the task prompts obtained from different methods. "Human." and "Social." denote the datasets classified as Humanities and Social Science in MMLU.

Experimental Setup

Tasks and Datasets. We select datasets from three groups of tasks for the experiment: Big-Bench Hard (BBH) (Suzgun et al. 2023) and GSM8K (Cobbe et al. 2021) for complex reasoning tasks, MMLU (Hendrycks et al. 2021) for knowledge-intensive tasks, and WSC (Levesque, Davis, and Morgenstern 2012) and WebNLG (Gardent et al. 2017) for common NLP tasks. Due to computational limitations, we sample a subset from each dataset for the main experiment. In addition, we use the lite BBH benchmark for detailed analysis.

Baselines. We select several representative methods for comparison, including existing LLM-based prompt optimizers and one from gradient-based model optimizers. (1) *SGDM* (Sutskever et al. 2013) is a momentum-based model optimizer. We adapt it for prompt optimization using the summarization-based trajectory and the editing-based refinement strategy. (2) *APE* (Zhou et al. 2023) utilizes LLMs to generate semantically similar variants of task prompts and selects one with the best performance. (3) *APO* (Pryzant et al. 2023) first uses reflection to obtain the gradient and then edits the task prompt accordingly. (4) *OPRO* (Yang et al. 2023) incorporates historical prompts with their scores into the meta-prompt. (5) *PE2* (Ye et al. 2023) adds historical prompts and reflection into the meta-prompt and controls the edit distance with a fixed constraint. In addition, we consider the baseline without an instruction ("Empty") and the instruction "Let' think step by step." from chain-of-thought prompting (Kojima et al. 2022) for performance comparison.

Evaluation Metrics. We report the average accuracy of all the subtasks for BBH and MMLU following Suzgun et al. (2023) and Hendrycks et al. (2021), accuracy for GSM8K following Cobbe et al. (2021), ROUGE-L for WSC and WebNLG following Wang et al. (2022b).

Implementation Details. We use both the base model (*i.e.,* `Llama-2-7b` and `Llama-3-8b`) and the instruction-tuned models (*i.e.,* `Baichuan2-7b-chat`, `Llama-2-7b-chat`, `Llama-2-13b-chat`, `Llama-3-8b-instruct` (Dubey et al. 2024), `GPT-3.5-turbo`, and `GPT-4`) as the task model. For the prompt optimizer, we use `GPT-3.5-turbo` and `GPT-4`. Unless otherwise specified, we use `Llama-2-7b-chat` as task model and `GPT-3.5-turbo` as prompt optimizer throughout experiments. We repeat all the experiments three times and report the average of the results.

Task model	Llama-2-7b	Llama-3-8b	Llama-2-7b-chat		Llama-3-8b-instruct	
Setting	I + D		I	I + D	I	I + D
Empty	40.28	60.42	32.29	36.63	48.26	54.86
CoT	36.46	51.39	31.25	34.20	52.43	54.34
SGDM	42.19	62.50	40.63	35.77	60.42	58.51
APE	42.54	61.28	42.01	36.29	59.43	58.51
APO	42.19	61.45	40.34	36.29	59.38	57.99
OPRO	42.02	62.68	42.14	36.46	62.33	59.02
PE2	42.88	63.20	42.01	36.81	62.67	58.68
GPO	**45.48**	**65.11**	**43.75**	**38.02**	**63.89**	**61.11**

Table 6: Performance comparison under different evaluation settings. "I" denotes instruction and "D" denotes demonstration.

Main Results

Table 5 and 6 show the results of different methods for prompt optimization across various tasks and evaluation settings.

First, when only considering the task prompt, we can see that trajectory-based methods (*i.e.,* SGDM, OPRO, PE2, and GPO) perform very well. One possible reason is that the trajectory helps the prompt optimizer pay more attention to the important information instead of the noise in the current step. Furthermore, our prompt optimizer GPO achieves the best performance across all tasks. Our relevance-based trajectory provides semantically similar demonstrations that can be ef-

Prompt optimizer	GPT-3.5-turbo					GPT-4				
Task model	Baichuan2-7b-chat	Llama-2-7b-chat	Llama-2-13b-chat	GPT-3.5-turbo	GPT-4	Baichuan2-7b-chat	Llama-2-7b-chat	Llama-2-13b-chat	GPT-3.5-turbo	GPT-4
Empty	15.75	32.29	40.97	60.48	72.87	15.75	32.29	40.97	60.48	72.87
CoT	19.45	31.25	40.28	62.15	73.61	19.45	31.25	40.28	62.15	73.61
SGDM	20.87	40.63	42.24	64.18	75.23	21.85	40.41	42.03	64.55	75.82
APE	20.61	42.01	41.65	63.63	74.71	20.29	39.98	42.96	64.38	74.13
APO	19.97	40.34	41.79	63.96	74.13	22.09	39.56	43.67	64.55	74.56
OPRO	19.86	42.14	42.89	64.56	75.06	21.28	41.77	43.35	64.91	76.67
PE2	21.11	42.01	42.68	64.95	75.27	22.43	42.03	44.91	65.23	76.55
GPO	**23.35**	**43.75**	**44.83**	**67.02**	**76.56**	**25.34**	**44.97**	**46.17**	**67.80**	**78.65**

Table 7: Performance of different prompt optimization methods with various models.

fectively learned by the LLM, while the cosine-based decay strategy can control the optimization process in a fine-grained manner through edit distance.

Second, under various evaluation settings for the lite BBH benchmark, it can be observed that GPO not only excels in the "Instruction" setting but also yields substantial gains in the "Instruction + Demonstration" setting for both the base model and the instruction-tuned variant. Even in the scenario that is challenging for baselines (*i.e.,* `Llama-2-7b-chat` with "Instruction + Demonstration"), our approach still demonstrates strong improvement. This showcases the versatility of our approach in both zero-shot and few-shot evaluation settings.

Detailed Analysis

Next, we conduct a detailed analysis of our prompt optimizer GPO from the following aspects.

The Impact of Model Selection. To confirm the effectiveness of GPO across different models, we explore the impact of different model combinations compared with other baseline methods. Table 7 presents the results on the lite BBH benchmark. In general, GPO demonstrates remarkable capabilities for prompt optimization across various scenarios, including strong-to-weak optimization, self-optimization, and weak-to-strong optimization. This indicates the versatility of our framework. Notably, `GPT-4` can consistently find better task prompts than `GPT-3.5-turbo`, which suggests the need for a capable model as the prompt optimizer.

The Efficiency of Optimization. LLM-based prompt optimization requires multiple interactions with the LLM. In this part, we investigate the efficiency of LLM-based prompt optimizers by examining the optimization curve over the first 12 steps. Figure 3 shows that on the movie recommendation dataset, compared to other methods, GPO demonstrates rapid enhancement of the task prompt in the early stage, followed by steady and consistent performance improvement in the later stage of optimization. Since GPO only utilizes task prompts to derive the update direction and performs fine-grained control over the variation, it can achieve better performance with high efficiency.

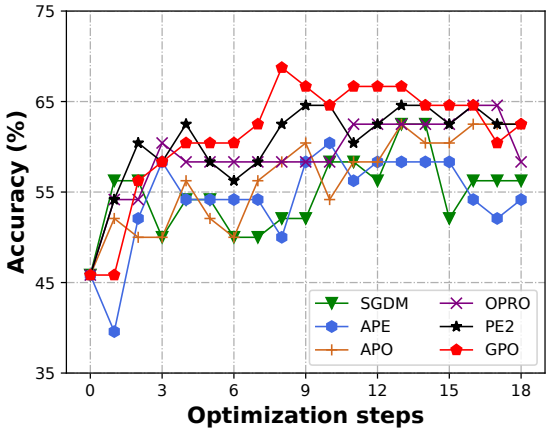

Figure 3: The efficiency of our approach GPO w.r.t. optimization steps.

Conclusion and Discussion

In this paper, we conduct a systematic analogy between gradient-based model optimizers and LLM-based prompt optimizers. Based on existing work and our newly proposed approach, we conduct an experimental analysis of the two key factors (*i.e.,* update direction and update method) to determine the best configuration. According to this configuration, we further propose a novel prompt optimization framework GPO. At each step, it retrieves relevant prompts from the optimization trajectory as the update direction. Then, it utilizes the generation-based refinement strategy to perform the update, while controlling the edit distance through a cosine-based decay strategy. Extensive experiments demonstrate the effectiveness and efficiency of GPO.

One limitation of our work is that we only draw an analogy with the most widely used gradient-based optimizers. More advanced model optimizers like Newton's method and its application to meta-prompts remain to be investigated. Additionally, our approach relies on textual update directions, future research could explore more direct and fine-grained numerical update signal methods (Nie et al. 2024).

Acknowledgements

This work was partially supported by National Natural Science Foundation of China under Grant No. 92470205 and 62222215. Xin Zhao is the corresponding author.

References

Amatriain, X. 2024. Prompt Design and Engineering: Introduction and Advanced Methods. *CoRR*, abs/2401.14423.

Boyd, S. P.; and Vandenberghe, L. 2014. *Convex Optimization*. Cambridge University Press. ISBN 978-0-521-83378-3.

Chen, L.; Chen, J.; Goldstein, T.; Huang, H.; and Zhou, T. 2023. InstructZero: Efficient Instruction Optimization for Black-Box Large Language Models. *CoRR*, abs/2306.03082.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *CoRR*, abs/2110.14168.

Deng, M.; Wang, J.; Hsieh, C.; Wang, Y.; Guo, H.; Shu, T.; Song, M.; Xing, E. P.; and Hu, Z. 2022. RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 3369–3391. Association for Computational Linguistics.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Fernando, C.; Banarse, D.; Michalewski, H.; Osindero, S.; and Rocktäschel, T. 2023. Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution. *CoRR*, abs/2309.16797.

Gardent, C.; Shimorina, A.; Narayan, S.; and Perez-Beltrachini, L. 2017. Creating Training Corpora for NLG Micro-Planners. In Barzilay, R.; and Kan, M., eds., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 179–188. Association for Computational Linguistics.

Gotmare, A.; Keskar, N. S.; Xiong, C.; and Socher, R. 2019. A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Goyal, P.; Dollár, P.; Girshick, R. B.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR*, abs/1706.02677.

Guo, Q.; Wang, R.; Guo, J.; Li, B.; Song, K.; Tan, X.; Liu, G.; Bian, J.; and Yang, Y. 2023. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers. *CoRR*, abs/2309.08532.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Huang, J.; Chen, X.; Mishra, S.; Zheng, H. S.; Yu, A. W.; Song, X.; and Zhou, D. 2023. Large Language Models Cannot Self-Correct Reasoning Yet. *CoRR*, abs/2310.01798.

Izmailov, P.; Podoprikhin, D.; Garipov, T.; Vetrov, D. P.; and Wilson, A. G. 2018. Averaging Weights Leads to Wider Optima and Better Generalization. In Globerson, A.; and Silva, R., eds., *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, 876–885. AUAI Press.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large Language Models are Zero-Shot Reasoners. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Levesque, H. J.; Davis, E.; and Morgenstern, L. 2012. The Winograd Schema Challenge. In Brewka, G.; Eiter, T.; and McIlraith, S. A., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, 4582–4597. Association for Computational Linguistics.

Lu, Y.; Bartolo, M.; Moore, A.; Riedel, S.; and Stenetorp, P. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 8086–8098. Association for Computational Linguistics.

Nie, A.; Cheng, C.; Kolobov, A.; and Swaminathan, A. 2024. The Importance of Directional Feedback for LLM-based Optimizers. *CoRR*, abs/2405.16434.

Prasad, A.; Hase, P.; Zhou, X.; and Bansal, M. 2023. GrIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models. In Vlachos, A.; and Augenstein, I., eds., *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, 3827–3846. Association for Computational Linguistics.

Pryzant, R.; Iter, D.; Li, J.; Lee, Y. T.; Zhu, C.; and Zeng, M. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, 7957–7968. Association for Computational Linguistics.

Schnabel, T.; and Neville, J. 2024. Prompts As Programs: A Structure-Aware Approach to Efficient Compile-Time Prompt Optimization. *arXiv preprint arXiv:2404.02319*.

Sun, S.; Cao, Z.; Zhu, H.; and Zhao, J. 2020. A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Trans. Cybern.*, 50(8): 3668–3681.

Sutskever, I.; Martens, J.; Dahl, G. E.; and Hinton, G. E. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, 1139–1147. JMLR.org.

Suzgun, M.; Scales, N.; Schärli, N.; Gehrmann, S.; Tay, Y.; Chung, H. W.; Chowdhery, A.; Le, Q. V.; Chi, E. H.; Zhou, D.; and Wei, J. 2023. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In Rogers, A.; Boyd-Graber, J. L.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, 13003–13051. Association for Computational Linguistics.

Tang, X.; Wang, X.; Zhao, W. X.; and Wen, J. 2024. DAWN-ICL: Strategic Planning of Problem-solving Trajectories for Zero-Shot In-Context Learning. *CoRR*, abs/2410.20215.

Wang, X.; Zhou, K.; Tang, X.; Zhao, W. X.; Pan, F.; Cao, Z.; and Wen, J. 2023. Improving Conversational Recommendation Systems via Counterfactual Data Simulation. In Singh, A. K.; Sun, Y.; Akoglu, L.; Gunopulos, D.; Yan, X.; Kumar, R.; Ozcan, F.; and Ye, J., eds., *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, 2398–2408. ACM.

Wang, X.; Zhou, K.; Wen, J.; and Zhao, W. X. 2022a. Towards Unified Conversational Recommender Systems via Knowledge-Enhanced Prompt Learning. In Zhang, A.; and Rangwala, H., eds., *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 1929–1937. ACM.

Wang, Y.; Mishra, S.; Alipoormolabashi, P.; Kordi, Y.; Mirzaei, A.; Naik, A.; Ashok, A.; Dhanasekaran, A. S.; Arunkumar, A.; Stap, D.; Pathak, E.; Karamanolakis, G.; Lai, H. G.; Purohit, I.; Mondal, I.; Anderson, J.; Kuznia, K.; Doshi, K.; Pal, K. K.; Patel, M.; Moradshahi, M.; Parmar, M.; Purohit, M.; Varshney, N.; Kaza, P. R.; Verma, P.; Puri, R. S.; Karia, R.; Doshi, S.; Sampat, S. K.; Mishra, S.; A, S. R.; Patro, S.; Dixit, T.; and Shen, X. 2022b. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. In *EMNLP*, 5085–5109. Association for Computational Linguistics.

Wang, Y.; Ren, R.; Li, J.; Zhao, X.; Liu, J.; and Wen, J. 2024. REAR: A Relevance-Aware Retrieval-Augmented Framework for Open-Domain Question Answering. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, 5613–5626. Association for Computational Linguistics.

Xiao, S.; Liu, Z.; Zhang, P.; and Muennighof, N. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. *CoRR*, abs/2309.07597.

Xu, H.; Chen, Y.; Du, Y.; Shao, N.; Wang, Y.; Li, H.; and Yang, Z. 2022. GPS: Genetic Prompt Search for Efficient Few-Shot Learning. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 8162–8171. Association for Computational Linguistics.

Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q. V.; Zhou, D.; and Chen, X. 2023. Large Language Models as Optimizers. *CoRR*, abs/2309.03409.

Yang, H.; and Li, K. 2023. InstOptima: Evolutionary Multi-objective Instruction Optimization via Large Language Model-based Instruction Operators. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, 13593–13602. Association for Computational Linguistics.

Ye, Q.; Axmed, M.; Pryzant, R.; and Khani, F. 2023. Prompt Engineering a Prompt Engineer. *CoRR*, abs/2311.05661.

Zhan, Y.-L.; Lu, Z.-Y.; Sun, H.; and Gao, Z.-F. 2024. Over-parameterized Student Model via Tensor Decomposition Boosted Knowledge Distillation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhang, T.; Wang, X.; Zhou, D.; Schuurmans, D.; and Gonzalez, J. E. 2023. TEMPERA: Test-Time Prompt Editing via Reinforcement Learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.; and Wen, J. 2023. A Survey of Large Language Models. *CoRR*, abs/2303.18223.

Zhou, Y.; Muresanu, A. I.; Han, Z.; Paster, K.; Pitis, S.; Chan, H.; and Ba, J. 2023. Large Language Models are Human-Level Prompt Engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.