LARGE LANGUAGE MODELS AS OPTIMIZERS

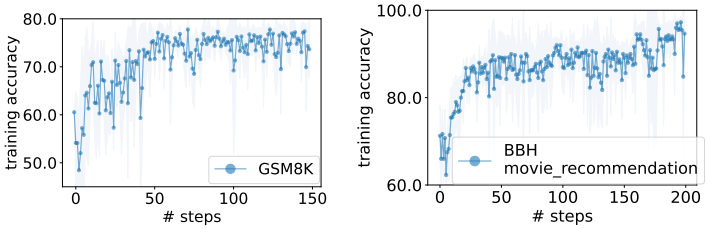
Chengrun Yang* **Xuezhi Wang** **Yifeng Lu** **Hanxiao Liu**
Quoc V. Le **Denny Zhou** **Xinyun Chen***

{chengrun, xuezhiw, yifenglu, hanxiaol}@google.com
{qvl, dennyzhou, xinyunchen}@google.com

Google DeepMind * Equal contribution

ABSTRACT

Optimization is ubiquitous. While derivative-based algorithms have been powerful tools for various problems, the absence of gradient imposes challenges on many real-world applications. In this work, we propose Optimization by PROmpting (OPRO), a simple and effective approach to leverage large language models (LLMs) as optimizers, where the optimization task is described in natural language. In each optimization step, the LLM generates new solutions from the prompt that contains previously generated solutions with their values, then the new solutions are evaluated and added to the prompt for the next optimization step. We first showcase OPRO on linear regression and traveling salesman problems, then move on to our main application in prompt optimization, where the goal is to find instructions that maximize the task accuracy. With a variety of LLMs, we demonstrate that the best prompts optimized by OPRO outperform human-designed prompts by up to 8% on GSM8K, and by up to 50% on Big-Bench Hard tasks. Code at `https://github.com/google-deepmind/opro`.



(a) GSM8K (b) BBH movie_recommendation

Figure 1: Prompt optimization on GSM8K (Cobbe et al., 2021) and BBH (Suzgun et al., 2022) movie_recommendation. The optimization on GSM8K has pre-trained `PaLM 2-L` as the scorer and the instruction-tuned `PaLM 2-L` (denoted `PaLM 2-L-IT`) as the optimizer; the optimization on BBH movie_recommendation has `text-bison` as the scorer and `PaLM 2-L-IT` as the optimizer. Each dot is the average accuracy across all (up to 8) generated instructions in the single step, and the shaded region represents standard deviation. See Section 5 for more details on experimental setup.

Table 1: Top instructions with the highest GSM8K zero-shot test accuracies from prompt optimization with different optimizer LLMs. All results use the pre-trained `PaLM 2-L` as the scorer.

Source	Instruction	Acc
Baselines		
(Kojima et al., 2022)	Let's think step by step.	71.8
(Zhou et al., 2022b)	Let's work this out in a step by step way to be sure we have the right answer.	58.8
	(empty string)	34.0
Ours		
`PaLM 2-L-IT`	Take a deep breath and work on this problem step-by-step.	**80.2**
`PaLM 2-L`	Break this down.	79.9
`gpt-3.5-turbo`	A little bit of arithmetic and a logical approach will help us quickly arrive at the solution to this problem.	78.5
`gpt-4`	Let's combine our numerical command and clear thinking to quickly and accurately decipher the answer.	74.5

1 INTRODUCTION

Optimization is critical for all areas. Many optimization techniques are iterative: the optimization starts from an initial solution, then iteratively updates the solution to optimize the objective function (Amari, 1993; Qian, 1999; Kingma & Ba, 2015; Bäck & Schwefel, 1993; Rios & Sahinidis, 2013; Reeves, 1993). The optimization algorithm typically needs to be customized for an individual task to deal with the specific challenges posed by the decision space and the performance landscape, especially for derivative-free optimization.

In this work, we propose Optimization by PROmpting (OPRO), a simple and effective approach to utilize large language models (LLMs) as optimizers. With the advancement of prompting techniques, LLMs have achieved impressive performance in various domains (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022; Zhou et al., 2022a; Madaan et al., 2023; Bai et al., 2022; Chen et al., 2023e). Their ability to understand natural language lays out a new possibility for optimization: instead of formally defining the optimization problem and deriving the update step with a programmed solver, we describe the optimization problem in natural language, then instruct the LLM to iteratively generate new solutions based on the problem description and the previously found solutions. Optimization with LLMs enables quick adaptation to different tasks by changing the problem description in the prompt, and the optimization process can be customized by adding instructions to specify the desired properties of the solutions.

To demonstrate the potential of LLMs for optimization, we first present case studies on linear regression and the traveling salesman problem, which are two classic optimization problems that underpin many others in mathematical optimization, computer science, and operations research. On small-scale optimization problems, we show that LLMs are able to find good-quality solutions simply through prompting, and sometimes match or surpass hand-designed heuristic algorithms.

Next, we demonstrate the ability of LLMs to optimize prompts: the goal is to find a prompt that maximizes the task accuracy. Specifically, we focus on natural language tasks where both the task input and output are texts. LLMs are shown to be sensitive to the prompt format (Zhao et al., 2021; Lu et al., 2021; Wei et al., 2023; Madaan & Yazdanbakhsh, 2022); in particular, semantically similar prompts may have drastically different performance (Kojima et al., 2022; Zhou et al., 2022b; Zhang et al., 2023), and the optimal prompt formats can be model-specific and task-specific (Ma et al., 2023; Chen et al., 2023c). Therefore, prompt engineering is often important for LLMs to achieve good performance (Reynolds & McDonell, 2021). However, the large and discrete prompt space makes it challenging for optimization, especially when only API access to the LLM is available. Following prior work on continuous and discrete prompt optimization (Lester et al., 2021; Li & Liang, 2021; Zhou et al., 2022b; Pryzant et al., 2023), we assume a training set is available to compute the training accuracy as the objective value for optimization, and we show in experiments that optimizing the prompt for accuracy on a small training set is sufficient to reach high performance on the test set.

The prompt to the LLM serves as a call to the optimizer, and we name it the *meta-prompt*. Figure 3 shows an example. The meta-prompt contains two core pieces of information. The first piece is previously generated prompts with their corresponding training accuracies. The second piece is the optimization problem description, which includes several exemplars randomly selected from the training set to exemplify the task of interest. We also provide instructions for the LLM to understand the relationships among different parts and the desired output format. Different from recent work on using LLMs for automatic prompt generation (Zhou et al., 2022b; Pryzant et al., 2023), each optimization step in our work *generates* new prompts that aim to increase the test accuracy based on a trajectory of previously generated prompts, instead of *editing* one input prompt according to natural language feedback (Pryzant et al., 2023) or requiring the new prompt to follow the same semantic meaning (Zhou et al., 2022b). Making use of the full optimization trajectory, OPRO enables the LLM to gradually generate new prompts that improve the task accuracy throughout the optimization process, where the initial prompts have low task accuracies.

We conduct comprehensive evaluation on several LLMs, including `text-bison` and `Palm 2-L` in the PaLM-2 model family (Anil et al., 2023), as well as `gpt-3.5-turbo` and `gpt-4` in the GPT model family. We optimize prompts on GSM8K (Cobbe et al., 2021) and Big-Bench Hard (Suzgun et al., 2022), which are reasoning benchmarks where prompting techniques have achieved remarkable performance breakthrough (Wei et al., 2022; Kojima et al., 2022; Suzgun et al., 2022). Starting from initial prompts with low task accuracies, we show that all LLMs in our evaluation are able to

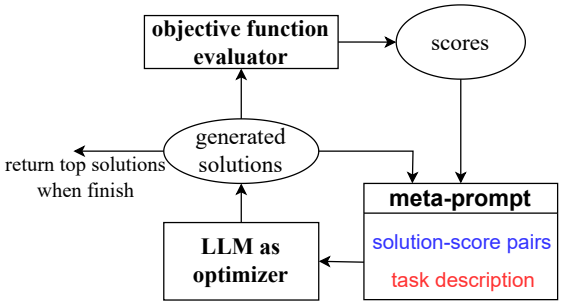


Figure 2: An overview of the OPRO framework. Given the meta-prompt as the input, the LLM generates new solutions to the objective function, then the new solutions and their scores are added into the meta-prompt for the next optimization step. The meta-prompt contains the solution-score pairs obtained throughout optimization, a natural language description of the task, and (in prompt optimization) a few task exemplars. Figure 3 shows a sample meta-prompt for prompt optimization.

serve as optimizers, which consistently improve the performance of the generated prompts through iterative optimization until convergence (see Figure 1). In particular, while these LLMs generally produce instructions of different styles (see Table 1), with zero-shot prompting, their best generated instructions match the few-shot chain-of-thought prompting performance when applied to `PaLM 2-L`, outperforming the zero-shot performance with human-designed prompts by up to 8% on GSM8K. Additionally, we observe that the OPRO-optimized prompts transfer to other benchmarks of the same domain and also deliver notable performance gain.

2 OPRO: LLM AS THE OPTIMIZER

Figure 2 illustrates the overall framework of OPRO. In each optimization step, the LLM generates candidate solutions to the optimization task based on the optimization problem description and previously evaluated solutions in the meta-prompt. Then the new solutions are evaluated and added to the meta-prompt for the subsequent optimization process. The optimization process terminates when the LLM is unable to propose new solutions with better optimization scores, or a maximum number of optimization steps has reached. We first outline the desired features of LLMs for optimization, then describe the key design choices based on these desirables.

2.1 DESIRABLES OF OPTIMIZATION BY LLMS

Making use of natural language descriptions. The main advantage of LLMs for optimization is their ability of understanding natural language, which allows people to describe their optimization tasks without formal specifications. For instance, in prompt optimization where the goal is to find a prompt that optimizes the task accuracy, the task can be described with a high-level text summary along with input-output examples.

Trading off exploration and exploitation. The exploration-exploitation trade-off is a fundamental challenge in optimization, and it is important for LLMs serving as optimizers to balance these two competing goals. This means that the LLM should be able to exploit promising areas of the search space where good solutions are already found, while also exploring new regions of the search space so as to not miss potentially better solutions.

2.2 META-PROMPT DESIGN

As the input to the optimizer LLM, the meta-prompt contains the following two essential parts.

Optimization problem description. The first part is the text description of the optimization problem, including the objective function and solution constraints. For example, for prompt optimization, the LLM can be instructed to "generate a new instruction that achieves a higher accuracy", and we denote such instructions in the meta-prompt as *meta-instructions*. We can also provide customized