Prompt Engineering a Prompt Engineer

Qinyuan Ye[1†] **Maxamed Axmed**[2] **Reid Pryzant**[2] **Fereshte Khani**[2]
[1]University of Southern California [2]Microsoft
qinyuany@usc.edu fkhani@microsoft.com

Abstract

Prompt engineering is a challenging yet crucial task for optimizing the performance of large language models on customized tasks. It requires complex reasoning to examine the model's errors, hypothesize what is missing or misleading in the current prompt, and communicate the task with clarity. While recent works indicate that large language models can be meta-prompted to perform automatic prompt engineering, we argue that their potential is limited due to insufficient guidance for complex reasoning in the meta-prompt. We fill this gap by infusing into the meta-prompt three key components: detailed descriptions, context specification, and a step-by-step reasoning template. The resulting method, named PE2, exhibits remarkable versatility across diverse language tasks. It finds prompts that outperform "let's think step by step" by 6.3% on MultiArith and 3.1% on GSM8K, and outperforms competitive baselines on counterfactual tasks by 6.9%. Further, we show that PE2 can make targeted and highly specific prompt edits, rectify erroneous prompts, and induce multi-step plans for complex tasks.

1 Introduction

Large language models (LLMs) are powerful tools for many natural language processing tasks, when provided with the right prompts.[1] However, LLMs are also notoriously sensitive to prompt design (Jiang et al., 2020; Zhao et al., 2021; Reynolds and McDonell, 2021; Lu et al., 2022), and finding the right prompts often requires extensive manual efforts referred to as "prompt engineering." Non-AI experts, in particular, may struggle to effectively communicate the task of interest to LLMs, resulting in prompt engineering being performed opportunistically rather than systematically (Zamfirescu-Pereira et al., 2023). Adding to this challenge, once a high-quality prompt is found and deployed into production, unforeseen edge cases can arise, necessitating more rounds of manual efforts. All these challenges give rise to an emerging research field of *automatic* prompt engineering. Within this field, a notable line of methods involves leveraging the capabilities of LLMs themselves (Zhou et al., 2023b; Pryzant et al., 2023). This entails meta-prompting LLMs with instructions such as "inspect the current prompt and a batch of examples, provide feedback, then propose a new prompt." (See Figure 1)

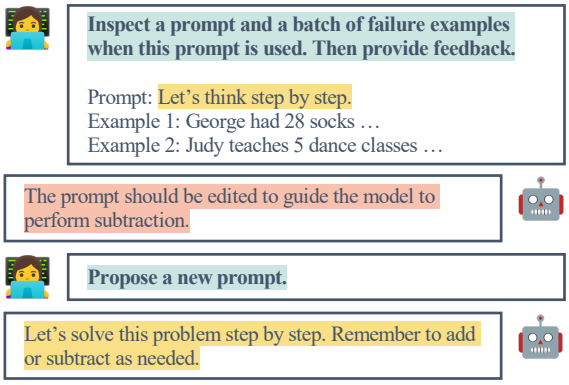

Figure 1: LLM-powered automatic prompt engineering methods typically use a meta-prompt that guides an LLM to inspect the current prompt, provide feedback (sometimes refered to as textual "gradients") and then generate an updated prompt. In this paper, we design and investigate meta-prompt variants to guide LLMs to perform automatic prompt engineering more effectively.

While these methods achieve impressive performance, a subsequent question arises: What makes a good meta-prompt for automatic prompt engineering? To answer this question, we connect two key observations: (1) Prompt engineering, at its core, is a language generation task that requires complex reasoning: it involves closely examining the model's errors, hypothesizing what is missing

[†]Work done while interning at Microsoft.

[1]In this paper, we focus on textual prompts of task description (*e.g.*, "Translate English to French") or instruction (*e.g.*, "Let's think step by step").

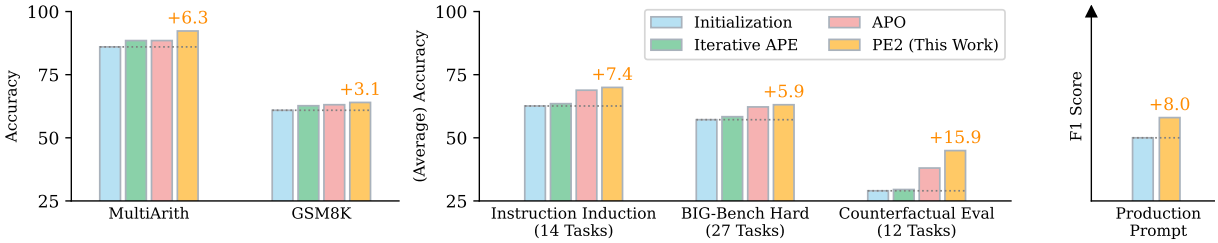

Figure 2: **Results Overview.** Our method PE2 consistently brings improvements over the prompt initialization (marked with orange text). It matches or outperforms prompt optimization baselines Iterative APE (Zhou et al., 2023b) and APO (Pryzant et al., 2023) across a wide range of language tasks, with most significant performance gain observed on Conterfactual Eval (Wu et al., 2024). See detailed performance breakdown in Fig. 10-13.

or misleading in the current prompt, and communicating the task more clearly to LLMs. (2) Complex reasoning capabilities in LLMs can be elicited by prompting the model to "think step by step" (Wei et al., 2022a; Kojima et al., 2022) and can be further improved by instructing them to reflect on their outputs (Madaan et al., 2023; Chen et al., 2024).

Bridging these two observations, in this work, we prompt engineer a prompt engineer—we aim to construct a meta-prompt that guide LLMs to perform automatic prompt engineering more effectively. We argue that prior works do not provide sufficient guidance in the meta-prompt, thereby limiting the potential of LLMs for automatic prompt engineering. To address this, we introduce new meta-prompt components such as detailed two-step task descriptions, context specification, and a step-by-step reasoning template, to better equip LLMs throughout the process (§3; Fig. 3).

The resulting method, named PE2, achieves strong empirical performance (§5.1). When using `text-davinci-003` as the task model, the prompts produced by PE2 surpass the zero-shot chain-of-thought prompt "let's think step by step" (Kojima et al., 2022) by 6.3% on MultiArith and 3.1% on GSM8K. Moreover, PE2 matches or outperforms two automatic prompt engineering baselines, Iterative APE (Zhou et al., 2023b) and APO (Pryzant et al., 2023) in multiple settings (Fig. 2). PE2 is most effective on counterfactual tasks (Wu et al., 2024), where the automatic prompt engineer is anticipated to reason about non-standard situations (*e.g.*, do addition in base-8 instead of base-10) and explain such situation to the task model through the prompt. Beyond academic datasets, we show that PE2 can improve an expert-written production prompt consisting of over 5,000 tokens, resulting in an 8.0% increase in F1 score.

We further provide a detailed analysis on the behaviors, advantages, and limitations of PE2. Upon examining the prompt edit history (§5.3), we find that PE2 consistently offers meaningful prompt edits (Table 6). It is able to amend erroneous or incomplete prompts and enrich the prompts with additional details, which leads to improved final performance. It is also able to devise multi-step plans for complex tasks. For example, in the task of movie recommendation, PE2 makes the plan to "consider factors such as genre, plot and style" in the prompt. Interestingly, when uninformed about performing addition in base-8, PE2 formulates partially-correct arithmetic rules from examples by itself: "If both numbers are less than 50, add 2 to the sum. If either number is 50 or greater, add 22 to the sum.[2]" This demonstrates PE2's remarkable ability to reason and adapt in non-standard situations, while also raises concerns of "shortcut learning" in prompt optimization.

2 Background

In this section, we provide a formal formulation of the prompt engineering problem (§2.1), and describe a general framework of automatic prompt engineering using LLMs and meta-prompts (§2.2). Building on this foundation, we introduce new meta-prompt components used in PE2 in §3.

2.1 Prompt Engineering

The goal of prompt engineering is to find the textual prompt p^* that achieves the best performance on a given dataset D when using a given LLM $\mathcal{M}_{task}$ as the task model. More specifically, we assume all datasets can be formatted as textual input-output pairs, *i.e.*, $D = \{(x, y)\}$. We are given a training set D_{train} for optimizing the prompt, D_{dev} for validation, and D_{test} for final evaluation. Following the notations in Zhou et al. (2023b), the prompt

[2]Both the base-8 addition rules and the model-induced rules hold true for examples like 75+7=104 and 5+6=13.

engineering problem can be described as:

$$p^* = \arg\max_p \sum_{(x,y)\in D_{dev}} f(\mathcal{M}_{task}(x;p), y) \quad (1)$$

where $\mathcal{M}_{task}(x;p)$ is the output generated by the task model when conditioning on the prompt p, and f is a per-example evaluation function. For example, if the evaluation metric is exact match, $f(\mathcal{M}_{task}(x;p), y) = \mathbb{1}[\mathcal{M}_{task}(x;p) = y]$.

2.2 Automatic Prompt Engineering with LLMs

To alleviate the intensive efforts of manual prompt engineering, recent works explore automating this process by meta-prompting LLMs to paraphrase the prompt (Zhou et al., 2023b) or refine the prompt by inspecting failure examples (Pryzant et al., 2023). In the following, we describe a framework that encapsulates these prior works and is employed in our development of PE2 in later sections. It has three parts: prompt initialization, new prompt proposal, and the search procedure.

Prompt Initialization. To start the prompt engineering process, a set of initial prompts $P^{(0)}$ is needed. We consider two initialization methods: **(1) Manual initialization** is applicable for tasks that has pre-existing prompts written by humans experts. For example, "Let's think step by step" is effective on mathematical reasoning tasks and can be used as the initialization for prompt optimization. In **(2) Induction Initialization**, we follow Zhou et al. (2023b) by using a batch of examples $\{(x, y)\}$ from D_{train} and a prompt p^{init} ("Here are the input-output pairs. What is the instruction?") to generate a set of initial prompts $P^{(0)}$.

New Prompt Proposal. Given a set of initial prompts, the automatic prompt engineer will continuously propose new and potentially better prompts. At timestamp t, the prompt engineer is given a prompt $p^{(t)}$ and expected to write a new prompt $p^{(t+1)}$. Optionally, a batch of examples $B = \{(x, y, y')\}$ may be inspected in the new prompt proposal process. Here $y' = \mathcal{M}_{task}(x;p)$ represents output generated by the task model and y represents the ground-truth label. We use p^{meta} to denote a meta-prompt that is used to instruct the prompt proposal model $\mathcal{M}_{proposal}$ to propose new prompts. Therefore,

$$p^{(t+1)} = \mathcal{M}_{proposal}(p^{(t)}, B; p^{meta}) \quad (2)$$

	Iter. APE	APO	PE2
Inspect Model Failures	✗	✓	✓
(a) Task Description			
- Length	Short	Short	Long
- # Steps	One-step	Two-step	Two-step
- Position	Beginning	On-the-fly	Both
(b) Context Specification	✗	✗	✓
(c) Step-by-step Template	✗	✗	✓

Table 1: Comparison of Meta-prompt Components Used in Baseline Methods and PE2.

Constructing a better meta-prompt p^{meta} to improve the quality of the proposed prompt $p^{(t+1)}$ is the main focus of this study. We will describe in more details in §3.

Search Procedure. As LLMs are sensitive to trivial prompt variations, it is possible that the newly proposed prompt $p^{(t+1)}$ under-performs the original prompt $p^{(t)}$. Therefore, automatic prompt engineering is typically combined with a back-tracking enabled search procedure. At timestamp t, we select n best-performing prompts from *all* prompt candidates obtained in previous timestamps (*i.e.*, $P^{(0)} \cup P^{(1)} \cup ... \cup P^{(t)}$). For *each* of these n prompts, we sample m different batches B of model errors, and run the meta-prompt in Eq. 2 to produce m new prompts. This results in $m \times n$ new prompts, which we denote as $P^{(t+1)}$ collectively and are used at the next timestamp $t + 1$. The search algorithm is described more formally in Algorithm 1.

3 Prompt Engineering a Prompt Engineer

Much like how the prompt plays an important role for the end task performance, the meta-prompt p^{meta} introduced in Eq. 2 plays an important role in the quality of newly proposed prompts, and the overall quality of automatic prompt engineering. In this work, we focus on prompt engineering the meta-prompt p^{meta}—we develop meta-prompt components that can potentially help improve LLMs' prompt engineering quality.

In the following, we reflect on the limitations of prior works and subsequently introduce three meta-prompt components targeting these limitations. We visualize these components in Fig. 3 and provide a summary in Table 1. We name our method using these three components as **PE2**, a prompt engineered prompt engineer.

(a) Two-step Task Description. In APO (Pryzant et al., 2023) the task of prompt engineering is decomposed into two steps (Fig. 1):

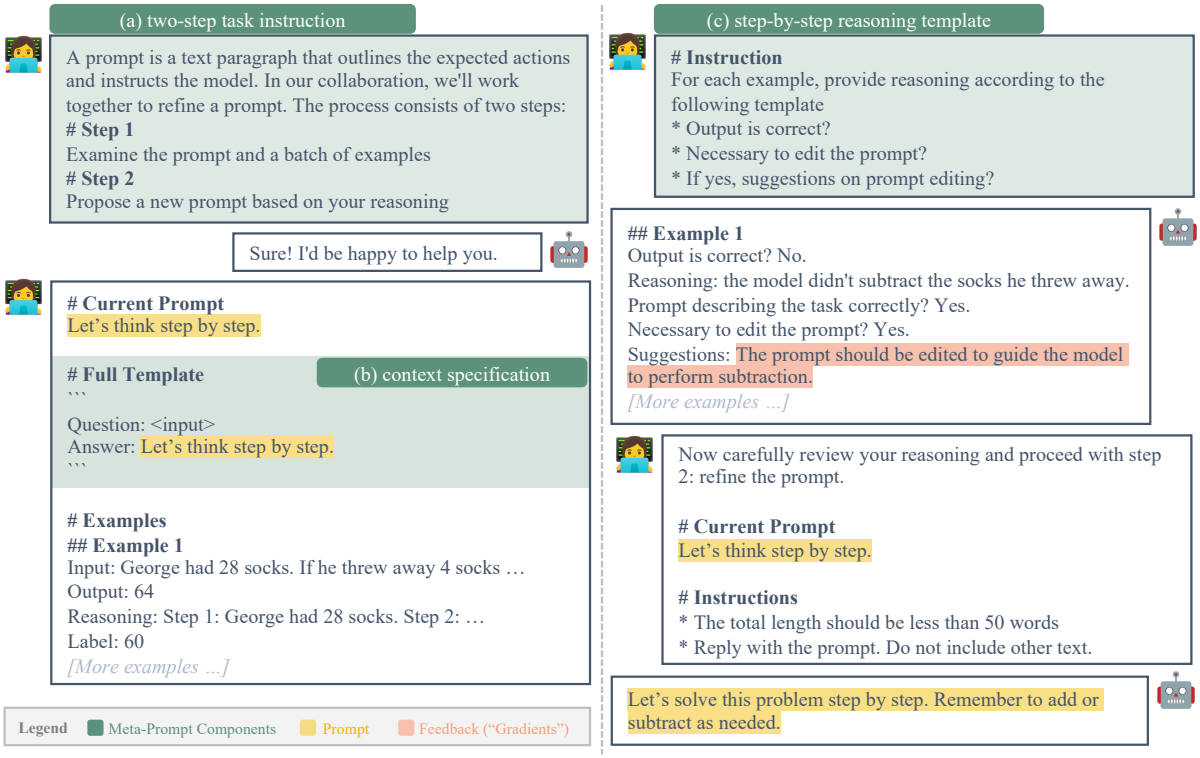

Figure 3: An redacted example to illustrate the meta-prompt components introduced in §3.

In step 1, the model is expected to inspect the current prompt and a batch. In step 2, the model is expected to generate an improved prompt.

However, in APO, each step is explained *briefly* and *on the fly*. In contrast, we consider clarifying the steps and expectations *upfront* in the meta-prompt, and also guiding the model with specific steps *on-the-fly*.

(b) Context Specification. In practice, how the prompt and the input text are formatted together is flexible. It may appear *before* the input text to describe the task, *e.g.*, "Translate English to French." It may appear *after* the input text, *e.g.*, "Let's think step by step" to elicit reasoning capabilities. Recognizing these varying contexts, we explicitly specify the layout of the prompt and the input.

(c) Step-by-step Reasoning Template. To encourage the model to examine *each* example in the batch B closely and reflect on the limitations in the current prompt, we guide the prompt proposal model $\mathcal{M}_{proposal}$ with a list of questions. For example: Is the prompt correctly describing the task? Is it necessary to edit the prompt? If yes, provide actionable suggestions on prompt editing.

Other Meta-prompt Components We Tried. Inspired by optimization concepts such as batch size, step size and momentum, we considered adding their verbalized counterparts to the meta-prompt and investigate their effects. We also considered adding a prompt engineering tutorial in the meta-prompt to help the LLM better understand the task. Our observations on these components are mixed. We report these results in Appendix B.

4 Experiment Setting

4.1 Tasks

We use the following five groups of tasks to evaluate the effectiveness of PE2. Details of the used datasets (*e.g.*, dataset sizes, train-test splits, license) are deferred in Appendix D.4.

(1) Math Reasoning. We use MultiArith (Roy and Roth, 2015) and GSM8K (Cobbe et al., 2021), which contain grade school math problems that requires multiple steps of arithmetic operations.

(2) Instruction Induction. Instruction Induction (Honovich et al., 2023) is a benchmark for inferring the underlying instruction from few-shot examples. We use 14 selected tasks that cover a wide range of use cases, *e.g.*, "Formality" is a task that aims at rephrasing a sentence to be more formal.

(3) BIG-bench Hard. BIG-bench Hard (Suzgun et al., 2023) is a collection of 23 tasks (27 subtasks)

that are challenging to LLMs but the performance may be improved with advanced prompting techniques (Wei et al., 2022b). Some of BIG-bench Hard tasks are closely related to real-world applications (*e.g.*, movie recommendation).

(4) Counterfactual Evaluation. We use the arithmetic, chess, and syntax tasks and their counterfactual variants introduced in Wu et al. (2024). For arithmetic, the original task is base-10 addition, and the counterfactual tasks are base-8/9/11/16 addition. For chess, the starting positions for knights and bishops are swapped in the counterfactual task. We use this set of tasks to investigate whether PE2 can reason about non-standard situations and communicate them to the task model.

(5) Production Prompt. Lastly, we optimize an internal production prompt for a hierarchical, multi-label classification task. The task is to classify a user query into domains, intents and slots, and then output a nested dictionary as the result. The initialization prompt is carefully designed by experienced engineers and has more than 5,000 tokens.

4.2 Compared Methods

We compare PE2 with the following automatic prompt engineering methods. **(a) APE** (Zhou et al., 2023b): The base version of APE is an initialization-only method and does not involve new prompt proposal steps. It uses an initialization prompt p^{init} to generate multiple prompt candidates from a few examples, and select the best one among them based on D_{dev} performance. **(b) Iterative APE** (Zhou et al., 2023b): After initialization, p^{meta} instructs the model to produce a paraphrase of $p^{(t)}$ and use it as $p^{(t+1)}$. **(c) APO** (Pryzant et al., 2023): p^{meta} contains short instructions on inspecting the batch B, generating textual "gradients" (feedback), and producing a new prompt $p^{(t+1)}$. We include the p^{init} and p^{meta} used in these baseline methods in Appendix F.

4.3 Expeirment Details

LLMs. By *default*, we use `gpt-4` (OpenAI, 2023) as prompt proposal model $\mathcal{M}_{proposal}$ and use `text-davinci-003` (Ouyang et al., 2022) as the task model $\mathcal{M}_{task}$ performing the underlying task. Experiments on BIG-bench Hard are conducted at a later stage, and we use `gpt-4-turbo` and `gpt-3.5-turbo-instruct` to save costs and demonstrate the compatibility of our methods. To ensure fair comparison, we always use the same set of LLMs ($\mathcal{M}_{proposal}$ and $\mathcal{M}_{task}$) when evaluating PE2 against other prompt optimization methods.

Prompt Initialization. For Math Reasoning and BIG-bench Hard tasks, we use "Let's think step by step." (Kojima et al., 2022) as the initialization prompt, which can elicit multi-step reasoning in LLMs to perform these tasks. For Instruction Induction, we follow the setting in prior works (Zhou et al., 2023b) and use induction initialization. For Counterfactual Eval, we experiment with both. For the production task, we use the prompt written by an experienced engineer.

Search Budget. We use the same search budget for all prompt optimization methods. For experiments using induction initialization, 30 prompts are generated by p^{init} and form the initial candidate set $P^{(0)}$. Due to budget constraints, the number of optimization steps T is set to be 3. At each timestamp, we select $n = 4$ best-performing prompts, and propose $m = 4$ prompts from each of them.

We defer other experiment details in Appendix D.

5 Results and Analysis

5.1 Main Results

Improved baselines with more recent LLMs. In Zero-shot CoT (Kojima et al., 2022) and APE (Zhou et al., 2023b), the results were obtained with a earlier `text-davinci-002` model. We first rerun the prompts in these two works with `text-davinci-003`, an upgraded model. In Table 2, we observe a significant performance boost by using `text-davinci-003`, suggesting that it is more capable of solving math reasoning problems with Zero-shot CoT. Moreover, the gaps between the two prompts are narrowed (MultiArith: $3.3\% \rightarrow 1.0\%$, GSM8K: $2.3\% \rightarrow 0.6\%$), indicating `text-davinci-003` has a reduced sensitivity to prompt paraphrasing. Given this, methods that rely on simple paraphrasing, such as Iterative APE, may not improve the final accuracy as effectively. More specific and targeted edits are necessary to improve the performance.

PE2 outperforms Iterative APE and APO on various tasks. PE2 is able to find a prompt that achieves 92.3% accuracy on MultiArith ($+6.3\%$ compared to Zero-shot CoT) and 64.0% on GSM8K ($+3.1\%$). Additionally, we demonstrate the wide applicablity of PE2 on a wide range of language tasks. In Fig. 2 we summarize the results and

Method	Task Model	Proposal Model	MultiArith Test	GSM8K Test
Fixed Prompt, Reported by Zhou et al. (2023b)				
Zero-shot CoT	TD002	-	78.7	40.7
APE	TD002	TD002	82.0	43.0
Fixed Prompt, Reproduced				
Zero-shot CoT	TD003	-	86.0	60.9
APE	TD003	-	87.0	61.5
Prompt Optimization				
Iterative APE	TD003	GPT-4	88.5	62.7
APO	TD003	GPT-4	88.5	63.1
PE2 (this work)	TD003	GPT-4	**92.3**	**64.0**

Table 2: Performance Comparison on Math Reasoning Tasks. TD002/003 stand for `text-davinci-002/003`. See Table 4 for the final prompts.

Method	GSM8k	MultiA.	Date	Hyper.	Temp.	Word
Init.	48.1	71.5	36	52	50	4
Iter. APE	49.7	73.5	48	48	42	20
APO	51.0	73.5	48	72	52	16
PE2	50.5	74.3	56	74	62	28

Tasks from BIG-Bench Hard: Date = Date Understanding; Hyper. = Hyperbaton; Temp. = Temporal Sequence, Word = Word Sorting.

Table 3: Results on six selected tasks when using `Mistral-7B-Instruct-v0.2` (Jiang et al., 2023) as the task model and `gpt-4-turbo` as the prompt proposal model. See Table 26 for the final prompts.

show that PE2 outperforms Iterative APE (Zhou et al., 2023b) and APO (Pryzant et al., 2023) in multiple cases. Most notably, when induction initialization is used, PE2 outperforms APO on 11 out of 12 counterfactual tasks (Fig. 11), exhibiting a 6.9% average increase in accuracy. This highlights PE2's capability in reasoning about contradictions and unconventional situations. We defer experiment details and performance breakdown for these benchmarks in Appendix E.

PE2 generates targeted prompt edits and high-quality prompts. In Fig. 4 we plot the quality of prompt proposals over the course of prompt optimization. We observe very distinct patterns for the three prompt optimization methods: Iterative APE is based on paraphrasing, so the newly generated prompts have smaller variance. APO makes drastically large prompt edits and thus the performance drops in the first step. Among the three methods, PE2 has a better balance between exploration and stability. In Table 4, we list the optimal prompts found by these methods. Both APO and PE2 are able to provide instructions on "considering all parts / details". In addition, PE2 is designed to inspect the batch closely, enabling it make very specific prompt edits such as "remember to add or subtract as needed".

Method	MultiArith Prompt
Fixed Prompt	
Zero-shot CoT	Let's think step by step.
APE	Let's work this out in a step by step way to be sure we have the right answer.
Prompt Optimization	
Iterative APE	Let's proceed in a methodical, step-by-step manner.
APO	Given the scenario, perform the necessary calculations step by step to find the final result. Consider all parts of the input and the sequence of events.
PE2 (this work)	Let's solve this problem by considering all the details. Pay attention to each piece of information, remember to add or subtract as needed, and perform the calculations step by step.

Table 4: MultiArith prompts found by compared prompt optimization methods. We use `text-davinci-003` as the task model and `gpt-4` as the prompt proposal model.

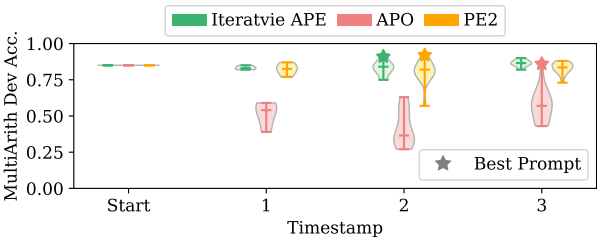

Figure 4: Prompt optimization dynamics of Iterative APE, APO and PE2 on MultiArith. The violins represent the quality distributions of newly proposed prompts at each optimization step. PE2 has a better balance between exploration and stability.

PE2 is widely applicable to various LLMs, including open-weight models. Our previous experiment setting employs different combinations of *closed-source* models as the task model $\mathcal{T}_{task}$ and the prompt proposal model $\mathcal{T}_{proposal}$ (§4.3). To further demonstrate PE2's model-agnostic nature, we introduce `Mistral-7B-Instruct-v0.2`, a recent open-weight model, as the task model, and uses `gpt-4-turbo` as the prompt proposal model. We report the results in Table 3 and the final optimized prompts in Table 26. Consistent with our experiments with closed-source models, results in Table 3 demonstrate that PE2 performs competitively with or surpasses other automated prompt engineering methods. As recent research suggests, LLMs still exhibit surprising sensitivity to prompt design and formatting (Sclar et al., 2024; Mizrahi et al., 2023), highlighting the importance of investigating why certain prompts succeed or fail. We hope PE2 em-

powers researchers to discover effective and ineffective prompts, which lay empirical foundations for future exploration into prompt sensitivity with open-weight models.

5.2 Ablation Study

To demonstrate the effectiveness of the three new meta-prompt components introduced in PE2, we run ablation experiments by removing these components during prompt optimization on MultiArith and GSM8K. In these experiments, we make sure that the meta-prompt still contains sufficient information about the task of prompt engineering. From the results in Table 5, we show that these three components contribute significantly to the final accuracy. In Fig. 5, we visualize the optimization dynamics of these ablation experiments. We find that the exclusion of any one of these components results in a higher variance in the quality distribution of newly-proposed prompts. Moreover, without these components, the proposal model more frequently suggests low-quality prompts.

We also conduct an ablation study on back-tracking (*i.e.*, at timestamp t, select top-performing prompts from $\cup_{i=0}^{t} P^{(i)}$ versus only $P^{(t)}$) and hard negative sampling (*i.e.*, the batch B is sampled from the model's errors, versus being randomly sampled from D_{train}). Since both techniques show slightly positive effects on PE2's performance, they are retained in the final version of PE2.

We encourage readers to refer to §B for additional meta-prompt components that we explored during PE2's development, such as verbalized "momentum", "step size", and a tutorial on prompt engineering. Although these elements were not included in the final version of PE2, we document them to encourage further exploration as more capable language models emerge in the future.

5.3 Case Study

PE2 amends erroneous or incomplete instructions, and devises multi-step plans for complex tasks. Tables 6 and 16 present notable prompt edits made by PE2. In the task of finding rhyming words (*e.g.* "car" rhymes with "bar"), induction initialization mistakenly suggests the task is about changing the first letter of a word. PE2 successfully correct this after one optimization step. In the task of movie recommendation, PE2 is able to decompose the complex task into concrete criteria, such as genre, plot and actor, when determining movie similarities. In date understanding, PE2 identifies the crucial step of referencing information about "today". These examples demonstrate PE2's ability to learn by summarizing key steps from failures and incorporating them into improved prompts, aligning with recent work (Zhang et al., 2024).

Method	MultiArith Dev	GSM8K Dev
PE2 (default)	92.0	68.0
Baselines		
Iterative APE	89.0	66.0
APO	86.0	60.0
Ablation: Meta-prompt Components		
- two-step task description	89.0	66.0
- step-by-step reasoning template	87.0	61.0
- context specification	93.0	63.0
Ablation: Search Algorithm Configurations		
- back-tracking	90.0	66.0
- hard negative sampling	90.0	68.0

Table 5: Ablation study on meta-prompt components.

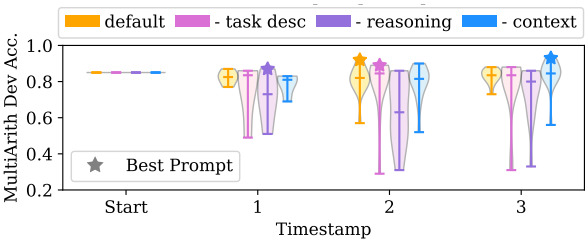

Figure 5: Prompt optimization dynamics on MultiArith when removing meta-prompt components. By removing one component, the new prompts have larger variance in their quality.

Limitations in meta-prompt following and hallucination. Despite the successes made by PE2, we note several factors that's limiting its performance when applied to challenging counterfactual tasks. We provide representative cases in Table 7. Occasionally, PE2 refuses to propose a new prompt and insists that "the prompt is correct, but the label is incorrect," even when we explicitly state "the labels are absolutely correct" within the meta-prompt. In another example, we attempt to guide it with hints (*e.g.*, suggesting a different numerical base), but this can unfortunately lead PE2 to generate incorrect solutions (*e.g.*, base-80) and even create justifications for these imagined solutions. These observations highlight the importance of improving LLMs' abilities to follow (meta-)instructions accurately and addressing hallucination issues.

Discussion on "shortcut learning." We find interesting yet concerning prompt edits on the coun-

Task	t	Prompt	Dev Acc.
Correct wrong or incomplete task instructions			
Rhymes	0	Remove the first letter from each input word and then replace that first letter with a similar sounding letter or group of letters to form a new word.	0.35
	1	Generate a word that rhymes with the input word.	0.45
Lay out tailored multi-step plans for complex problems			
Movie Recommendation	0	Let's think step by step.	0.58
	1	Consider the genre, plot, and style of the input movies. Using this information, think step by step to identify which of the following options is most similar to the given movies.	0.74
	2	Considering factors such as genre, director, actors, release period, audience target, animation style, and humor, analyze the similarities among the given movies and identify the movie from the options that shares the most similarities.	0.82
Date Understanding	0	Let's think step by step.	0.39
	2	Analyzing the given information, let's calculate the solution. Remember to consider the context provided, such as references to 'today' or specific dates.	0.54
Produce shortcut solutions in counterfactual tasks			
Base-8 Addition (Induction Init.)	0	Add the two numbers given as input to get the output.	0.0
	3	Add the two numbers provided in the input. Then, adjust this sum based on the following rule: if both numbers are less than 50, add 2 to the sum. If either number is 50 or greater, add 22 to the sum. The final result is the output.	0.35

Table 6: Notable prompt edits made by PE2. See §5.3 for discussion. See Table 16 for additional examples.

terfactual task of base-8 addition. When induction initialization is used (*i.e.*, PE2 is uninformed with the information of base-8 and must infer it from the examples), PE2 is able to devise its own heuristic that is partially correct ("... if both numbers are less than 50, add 2 to the sum. If either number is 50 or greater, add 22 to the sum"; see Table 6). This heuristic holds true for a subset of test cases like 75+7=104 and 5+6=13, but is ultimately not the intended solution.

On the positive side, it demonstrates PE2's ability to adapt in unseen scenarios and engage in sophisticated counterfactual reasoning. However, it is concerning that models prompted with these self-induced shortcut solutions achieve a test accuracy of 37% (average over 5 runs), which outperform models explicitly prompted to perform base-8 addition (test acc: 17%/28% before/after PE2 optimization). Shortcut learning (Geirhos et al., 2020) has been studied extensively for gradient-based optimization. Our experiments suggest that similar failure modes may be present in automatic prompt optimization.

5.4 Additional Analysis

Due to space limit, we summarize our other findings below and defer the details to Appendix A.

Effect of Initialization. (§A.1) (1) PE2 is able to recover from misleading or irrelevant prompt initializations, however the final prompt after optimization is worse than when using an instructive initialization. (2) We experiment with induction initialization. In this case, PE2 is able to discover a high quality prompt *from scratch* that matches with "Let's think step by step" on MultiArith.

Effect of Task Format/Difficulty. (§A.2) We experiment with using a generative format (*i.e.*, generating the answer string) and a multi-choice format (*i.e.*, selecting from given choices A/B/C/D) on the Date Understanding task in BIG-bench Hard. We observe that automatic prompt engineering methods has limited effect on the multi-choice format, but bring significant gains on the generative format.

Do optimized prompts generalize to other LLMs? (§A.3) We evaluate prompts optimized for `text-davinci-003` on other models such as `mpt-7b-instruct`, `yi-6b` and `mistral-7b-instruct`. We do not observe consistent cross-model generalization trends. This suggests that, although PE2 is a model-agnostic prompt optimization method (*i.e.*, can be applied to various $\mathcal{M}_{task}$), the final optimized prompts are specific to the underlying task model.

6 Related Work

Automatic Prompt Engineering. To alleviate the intensive trial-and-error efforts in manual prompt engineering, the research community has developed various strategies to automate this process with techniques such as incremental editing (Prasad et al., 2023), reinforcement learning (Deng

Task	Meta-prompt and Reasoning Snippets
Neglecting instructions in the meta-prompt	
Base-9 Addition	**Meta-prompt:** ... Note that the ground-truth labels are __absolutely correct__, but the prompts (task descriptions) may be incorrect and need modification. ...
	Reasoning: No, it is not necessary to edit the prompt. The prompt is correct, but the label is incorrect. ... The issue seems to be with the label, not the prompt.
Hallucination (when hints are provided in the meta-prompt)	
Base-8 Addition	**Hint:** The calculation may be performed in a different numerical base.
	Reasoning: ... Given this, it's possible that the numbers are being added in base 80, not base 10. In base 80, adding 20 to the sum of two numbers would be equivalent to adding 1 in base 10.
	New Prompt: The inputs are two numbers separated by a '+'. Add these two numbers together in base 80, then add 1 to give the output in base 10.

Table 7: Limitations and failure cases of PE2. See §5.3 for discussion.

et al., 2022; Zhang et al., 2022), algorithmic search (Xu et al., 2022), generating in-context demonstrations adaptively (Wan et al., 2023a,b), among others. A line of work focus on meta-prompting LLMs themselves for automatic prompt engineering (Honovich et al., 2023; Zhou et al., 2023b; Pryzant et al., 2023). In our work, we discuss potential limitations in these methods and subsequently introduce new meta-prompt components in PE2.

Prompting LLMs for Complex Reasoning Tasks. Recent research works suggest that LLMs can perform complex reasoning tasks, *e.g.*, grade-school math problems (Cobbe et al., 2021). There are two major techniques to boost LLMs' performance on this: **(1) prompting methods** that guide the model to produce intermediate reasoning steps, either with few-shot demonstrations (Nye et al., 2021; Wei et al., 2022a; Yao et al., 2023) or with zero-shot prompts (Kojima et al., 2022); **(2) self-reflection methods** that progressively guide the model to inspect its current output and refine it (Chen et al., 2024; Madaan et al., 2023; Paul et al., 2023; Kim et al., 2023a). At its core, prompt engineering is a language generation task requiring complex reasoning. Human prompt engineers usually examine the failure cases produced by the current prompt closely, make hypotheses, and compose a new prompt. In this work, we explore various prompting strategies when building an LLM-powered automatic prompt engineer.

Self-training and Self-improving for LLMs. Self-training refers to the technique of using a weak model to annotate input-label pairs and using them for further training (Rosenberg et al., 2005). In the context of LLMs, STaR (Zelikman et al., 2022) and Self-Improve (Huang et al., 2023) show that employing LLMs to generate high-quality reasoning chains, followed by model fine-tuning on these chains, can significantly improve the model's reasoning capabilities. In this work, we consider textual prompts as the "parameters" of LLMs, and we optimize these "parameters" with LLMs. This may be categorized as a case of self-improving (Goodman, 2023). More discussion in Appendix C.1.

7 Conclusion

In this paper, we introduced three meta-prompt components that lead to improved performance on automatic prompt engineering. The resulting method PE2 refines prompts written by human experts and surpasses established automatic prompt engineering baselines across various scenarios, notably on counterfactual tasks and a production application. Through comprehensive analysis and case studies, we illustrate PE2's ability to make targeted prompt edits and generate high-quality prompts, and demonstrate its general applicability with different LLMs.

The challenge of prompt engineering a prompt engineer remains ongoing. As highlighted in our case study, we believe improving the LLM's instruction following abilities and mitigating hallucination issues will be crucial for improving automatic prompt engineering. As the capabilities of LLM continue to evolve, their potential involvement in optimization or feedback loops necessitates a deeper empirical understanding of their failure modes (Pan et al., 2024), including shortcut learning discovered in this study. Looking ahead, we are also excited about applying PE2 to optimize its own meta-prompt in a self-referential way, in line with Metz et al. (2020); Irie et al. (2022); Fernando et al. (2023); Zelikman et al. (2023).

Limitations

Firstly, we opt for a relatively small prompt search budget ($T = 3$, $m = 4$, $n = 4$; see §4.3) due to cost considerations. In most of our experiments, the performance tends to plateau after $T = 3$ optimization steps. However, it's important to consider that the use of the initialization prompt "let's think step by step" in many cases might introduce a potential confounding factor. This prompt could be already near-optimal, leading to fast convergence during prompt optimization. Given the stochastic nature of natural language generation sampling and prompt optimization dynamics, it is possible that a larger prompt search budget or different experimental settings could yield new insights and conclusions.

Secondly, our study uses proprietary models such as `gpt-4` and `text-davinci-003`. (1) It raises reproducibility concerns as proprietary models may undergo upgrades or discontinuation over time. However, we believe the core concepts introduced in this paper is model-agnostic. This is supported by our experiments where we use two different sets of ($\mathcal{M}_{proposal}$, $\mathcal{M}_{task}$) (see §4.3) and experiments using `Mistral-7B-Instruct-v0.2` model as the task model. (2) It also raises concerns on data contamination, as the tasks and prompts included in our study may or may not have been part of the model's training data.

Lastly, apart from the three translation tasks in the Instruction Induction benchmark, our study predominantly focuses on tasks in English. We recognize the importance of inclusivity in language technology and acknowledge the need to extend our research to a multilingual setting in the future.

Acknowledgments

We thank anonymous reviewers, members of USC NLP, and members of Microsoft Office of Applied Research for their valuable feedback. In particular, Qinyuan Ye would like to thank Mayee Chen, Zhuoran Lu, Onkar Kulkarni, Jakob Schoeffer, Connor Lawless and Marios Papachristou for the insightful conversations and for making her internship a truly memorable experience. Qinyuan Ye was supported by a USC Annenberg Fellowship.

References

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. 2022. Controllable text generation with language constraints. *arXiv preprint arXiv:2212.10466*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Noah Goodman. 2023. Meta-prompt: A simple self-improving language agent.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, Toronto, Canada. Association for Computational Linguistics.

Xinyu Hu, Pengfei Tang, Simiao Zuo, Zihan Wang, Bowen Song, Qiang Lou, Jian Jiao, and Denis X Charles. 2024. Evoke: Evoking critical thinking abilities in LLMs via reviewer-author prompt editing. In *The Twelfth International Conference on Learning Representations*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. 2022. A modern self-referential weight matrix that learns to modify itself. In *Proc. Int. Conf. on Machine Learning (ICML)*, Baltimore, MD, USA.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023a. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.

Seungone Kim, Se Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023b. The CoT collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12685–12708, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Moxin Li, Wenjie Wang, Fuli Feng, Yixin Cao, Jizhi Zhang, and Tat-Seng Chua. 2023. Robust prompt optimization for large language models against distribution shifts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1539–1554, Singapore. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Luke Metz, Niru Maheswaranathan, C Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. 2020. Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves. *arXiv preprint arXiv:2009.11243*.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2023. State of what art? a call for multi-prompt llm evaluation. *arXiv preprint arXiv:2401.00595*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. 2024. Feedback loops with language models drive in-context reward hacking. *arXiv preprint arXiv:2402.06627*.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, 1:29–36.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3155–3168, Singapore. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan Arik, and Tomas Pfister. 2023a. Better zero-shot reasoning with self-adaptive prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3493–3514, Toronto, Canada. Association for Computational Linguistics.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, Hanjun Dai, Julian Eisenschlos, Sercan Arik, and Tomas Pfister. 2023b. Universal self-adaptive prompting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7437–7462, Singapore. Association for Computational Linguistics.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022a. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, Mexico City, Mexico. Association for Computational Linguistics.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. 2022. Gps: Genetic prompt search for efficient few-shot learning. *arXiv preprint arXiv:2210.17041*.

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*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA. Association for Computing Machinery.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2023. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. 2024. In-context principle learning from mistakes. *arXiv preprint arXiv:2402.05403*.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. 2022. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed H Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *arXiv preprint arXiv:2402.03620*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023a. Controlled text generation with natural language instructions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42602–42613. PMLR.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023b. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A Additional Results and Analysis

A.1 Effect of Initialization

Previously, we use "Let's think step by step" as the initialization for math reasoning tasks. We further experiment with using a *misleading* prompt, an *irrelevant* prompt and *induction* initialization (induction from a few examples). The results are presented in Table 8 and the optimization dynamics are visualized in Fig. 6.

Initialization	MultiArith Dev	GSM8K Dev
default (Let's think step by step.)	92.0	68.0
misleading† (Don't think. Just feel.)	81.0	50.0
irrelevant† (It's a beautiful day.)	73.0	49.0
induction from few-shot examples	84.0	43.0
no-op (Let's think step by step.)	85.0	57.0

Table 8: Effect of Initialization. † The prompts are originally from (Kojima et al., 2022).

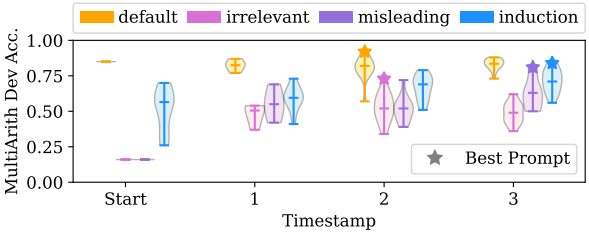

Figure 6: Prompt optimization dynamics on MultiArith when different prompt initializations are used.

In general, performance drops when alternative initialization methods are used, which highlights the importance of high-quality initialization. Still, PE2 is able to override the irrelevant or misleading prompts and gradually improve the performance (Fig. 6). Remarkably, PE2 is able to discover a high quality prompt[3] by itself using induction initialization (84% on MultiArith-Dev) that almost matches with "Let's think step by step" (85%) designed by highly-experienced human prompt engineers. This demonstrates the impressive prompt engineering capability of PE2 and suggests its potential for finding even better prompts when given additional computational resources.

[3]MultiArith prompt found by PE2 using induction initialization: "Analyze the problem and perform the calculations. Consider addition, subtraction, division, multiplication and perform them in the order they appear. If required, round up results to the nearest whole number. Subtract done tasks from total when necessary."

A.2 Effect of Task Format

For Date Understanding from BIG-bench Hard, we experiment with both a generative format (*i.e.*, generating the answer string; used in Gao et al. (2023)) and a discriminative/multi-choice format (*i.e.*, selecting from given choices A/B/C/D; used in Suzgun et al. (2023)). For Movie Recommendation, we experiment with two different multi-choice formats. See Table 9 for the formats that we used.

Task	Example
Date Understanding (multi-choice)	Today is 9/7. Jane is watching NFL 2003. What is the date tomorrow in MM/DD/YYYY? Options: (A) 09/08/1916 (B) 09/13/2003 (C) 08/18/2003 (D) 09/08/2003 (E) 09/15/2003 (F) 09/01/2003 **(D)**
Date Understanding (generative)	May 6, 1992 is like yesterday to Jane, but that is actually ten years ago. What is the date a month ago in MM/DD/YYYY? **04/06/2002**
Movie Recommendation (multi-choice 1)	Find a movie similar to Rocky, Star Wars Episode IV - A New Hope, Toy Story, The Terminator: Options: (A) Dracula Dead and Loving It (B) Independence Day (C) The Extraordinary Adventures of Adèle Blanc-Sec (D) The American President **(B)**
Movie Recommendation (multi-choice 2)	What movie is simlar to Apollo 13, Jurassic Park, Die Hard With a Vengeance, Forrest Gump? Choose from the following: Killer Movie, Stealth, The Last Man on Earth, True Lies. **True Lies**

Table 9: Different Task Formats for Date Understanding and Movie Recommendation. The correct answer is marked in **blue**.

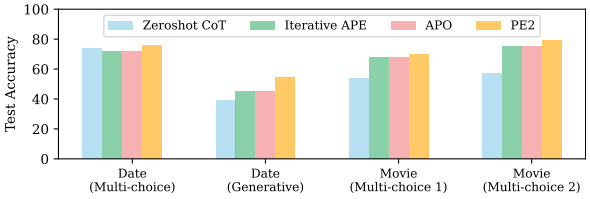

Figure 7: Effect of Task Format. See Table 9 for the formats used.

We report the results in Fig. 7. For Date Understanding, the multi-choice format narrows the output space and thus lower the difficulty of the task. We hypothesize that in combination with Zero-shot CoT, the task performance is close to saturation and automatic prompt engineering method does not provide extra benefit. The task is more challenging in the generative format and the detailed instructions in the optimized prompt bring significant performance gains.

For Movie Recommendation, we found that prompt optimization methods bring significant performance gains in both cases. The optimized prompts contain multi-step plans (*e.g.*, consider genre, director, ...), which boost the task performance. Minor formatting decisions such as outputting the option letter or the option string can still mildly affect the final accuracy.

Overall, the question of "when is automatic prompt optimization most effective" is dependent on many factors, including task format, task difficulty, output space size, task model's instruction following abilities, etc.

A.3 Do optimized prompts generalize to other LLMs?

We evaluate 5 GSM8K prompts for our prompt generalization study (see Table 10). Note that the APO and PE2 prompts are optimized for `text-davinci-003`. The two prompts reported in OPRO (Yang et al., 2024) are optimized for `PaLM 2-L`. We investigate the generalization of optimized prompts by evaluating them on four models: `gpt-3.5-turbo-instruct`, `mistral-7b-instruct-v0.2`, `yi-6b`, and `mpt-7b-instruct`. We report the results in Fig. 8.

Method	GSM8K Prompt
Zero-shot CoT	Let's think step by step.
APO	Given the scenario, perform necessary calculations and provide a step-by-step explanation to arrive at the correct numerical answer. Consider all information provided.
PE2	Let's solve the problem step-by-step and calculate the required total value correctly.
OPRO (1)	Take a deep breath and work on this problem step-by-step.
OPRO (2)	Let's combine our numerical command and clear thinking to quickly and accurately decipher the answer.

Table 10: Prompts used in transferability study in Fig. 8.

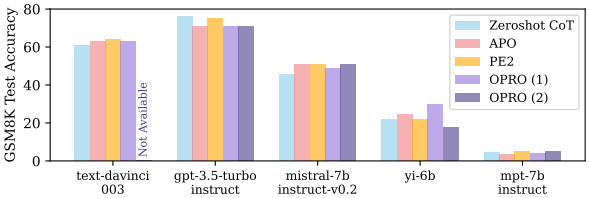

Figure 8: Analysis on generalizability of prompts across models. See Table 10 for the prompts used in this study.

Our results do not exhibit consistent generalization trends. Optimized prompts generally outperforms the original zero-shot CoT prompt on `text-davinci-003`, `mistral-7b-instruct-v0.2`, `yi-6b`. However, with `gpt-3.5-turbo-instruct`, the original CoT prompt outperforms all optimized prompts. Our hypothesis is that "Let's think step by step" is included in public instruction tuning

collections (Kim et al., 2023b) and thus models trained on these collections may perform better with this special prompt. However the instruction tuning mixture used for training `gpt-3.5-turbo-instruct` are not disclosed, and thus we cannot further investigate this.

Overall, our results suggest that current automatic prompt optimization methods tend to find model-specific prompts that *do not* reliably generalize to alternative models. This conclusion contrasts with the findings in PromptAgent (Wang et al., 2024), which we attribute to discrepancies in experimental setup. To maintain consistency with prior research (Zhou et al., 2023b), we have limited the prompt length to be 50 or 200 tokens. The conclusion may differ when this constraint is removed.

Future work may develop robust prompt optimization methods that operate across multiple task models, in a way similar to Li et al. (2023) which operates across domains. This may help identify high-quality prompts invariant to the underlying task model, so that when new and more powerful models (*e.g.*, GPT-5) are released, the optimized prompt may be used directly.

(Continued on next page)

B Other Meta-Prompt Components We Tried

In addition to the meta-prompt components studies in the main paper, we also tried other components in the early stage of PE2's development. As the results are mixed and inconclusive on these components, we report them here in the appendix. We illustrate these components in Fig. 9.

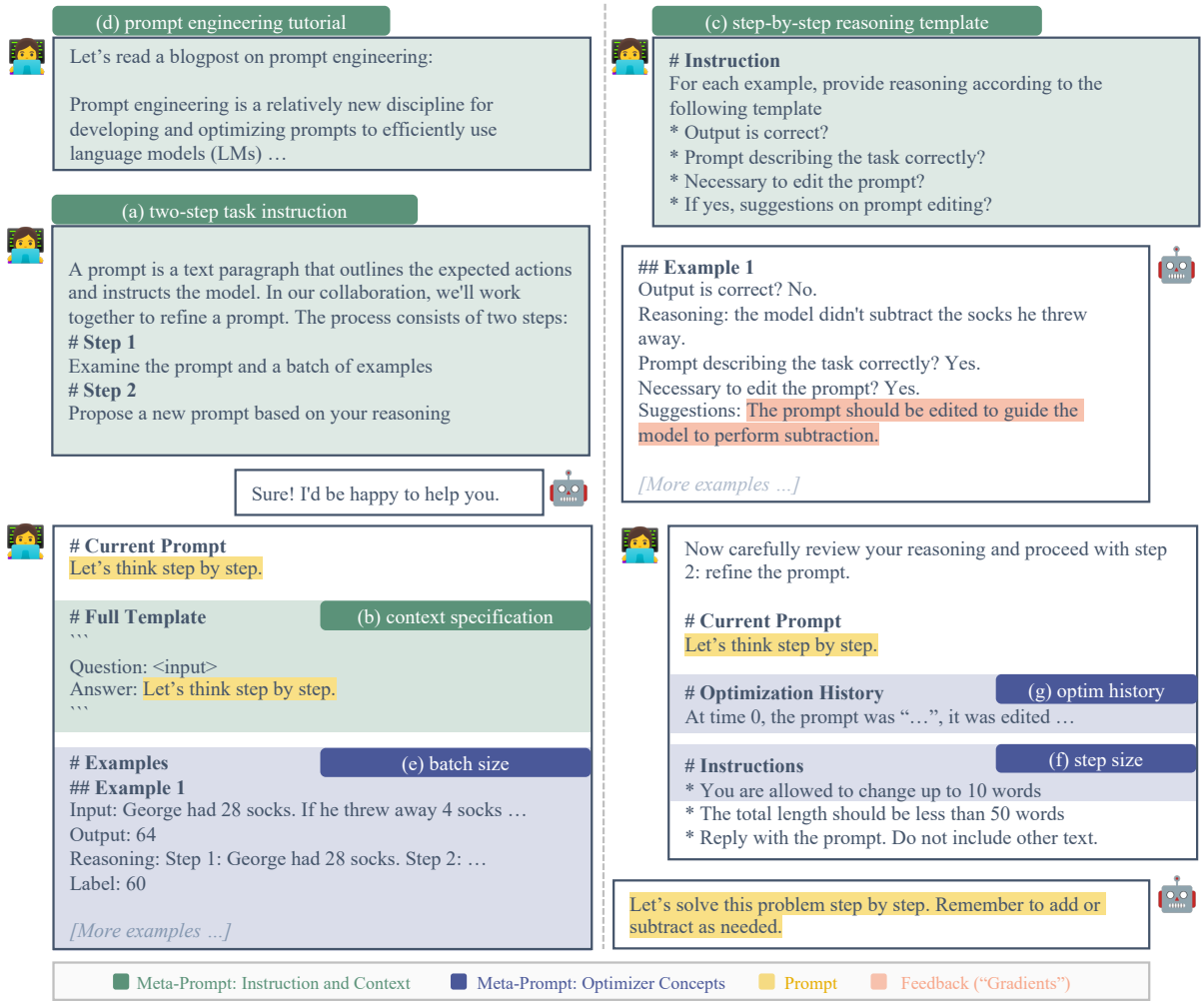

Figure 9: Illustration of meta-prompt components discussed in Appendix B.

Providing Detailed Instructions and Context.

(d) **Prompt Engineering Tutorial.** To help the LLM better understand the task of prompt engineering, we provide an online tutorial of prompt engineering in the meta-prompt.[4]

Incorporating Common Optimizer Concepts. The prompt engineering problem described in Eq. 1 is essentially an optimization problem, and the prompt proposal in Eq. 2 can be considered as doing one optimization step. Thus, we consider the following concepts commonly used in gradient-based optimization and develop their *verbalized counterparts* to be used in our meta-prompt.

(e) **Batch Size.** Batch size is the number of (failure) examples that is used in each prompt proposal step (Eq. 2). By default PE2 uses a batch size of 2. We experiment with batch sizes of $\{1, 4, 8\}$ additionally in this section.

(f) **Step Size.** In gradient-based optimization, the step size determines the extent to which the model's weights are updated. In prompt engineering, the counterpart would be the number of words (tokens)

[4] https://www.promptingguide.ai/introduction. Published under MIT license.

Method	MultiArith Dev	GSM8K Dev
PE2 (default)	92.0	68.0
Meta-prompt Components		
+ prompt engineering tutorial	90.0	63.0
+ tune batch size $\{1, 2, 4, 8\}$	92.0	68.0
+ tune step size $\{5, 10, 15, \text{None}\}$	95.0	68.0
+ optim history and momentum	93.0	67.0

Table 11: Investigation on meta-prompt components and configurations (Appendix).

that can be modified. We directly specify that "You are allowed to change up to s words in the original prompt" in the meta-prompt, where $s \in \{5, 10, 15, \text{None}\}$.[5]

(g) **Optimization History and Momentum.** Momentum (Qian, 1999) is a technique to accelerate optimization and avoid oscillations by maintaining the moving average of past gradients. To develop the verbalized counterpart of momentum, we include all past prompts (at timestamp $0, 1, ..., t-1$), their performance on the dev set, and a summary of prompt edits.

Results and discussion. We report the results when these components are used in Table 11. We do not observe significant improvement by incorporating prompt engineering tutorial. As the tutorial is excessively long (2500+ tokens) and slows down the runtime, we do not include it in the final version of PE2. The optimizer-inspired concepts can improve the performance occasionally, but the current experiments do not give a consistent conclusion regarding their utilities.

Similar to the challenges encountered in gradient-based optimization, the process of hyperparameter selection is inherently noisy and often varies depending on the task at hand. For discrete prompt optimization, this complexity is further compounded by factors such as the task model's sensitivity to prompts and the proposal model's capability to follow instructions in the meta-prompt. For example, Sun et al. (2023) point out that LLMs struggled at meeting fine-grained constraints such as "generate exactly 5 words," which could potentially diminish the effectiveness of the (f) step size component. Additionally, (g) momentum requires multiple optimization steps to accumulate, yet our experiments are restricted to three steps due to cost constraints.

Although these meta-prompt components do not currently take effect consistently with existing LLMs and experimental settings, their potential justifies re-examination in the future, particularly as models become more capable, and the efficiency and scalability of automatic prompt engineering methods improve.

C Additional Discussion

C.1 Recent Works

A recent work (Yang et al., 2024) introduced the concept of large language models as optimizers and proposed optimization by prompting (OPRO). In the following, we discuss the differences and connections between OPRO and our work.

(1) Focus of the work. Both OPRO and our work conduct experiments on prompt optimization; the focus of the two works differ. OPRO can be applied to general optimization problems, including linear regression and traveling salesman problem. In our work we limit the scope to prompt optimization, with a specific focus on proposing and investigating different components in the meta-prompt.

(2) Optimization strategy. The optimization strategies of the two works are different. PE2 is largely inspired by the concepts in APO (Pryzant et al., 2023), instructing the model to produce textual feedback ("gradient") *explicitly*. It is more analogous to gradient descent. OPRO uses the execution accuracy as rewards to guide the optimization *indirectly*, which, in our understanding, is more analogous to in-context

[5] Chen et al. (2022) and Zhou et al. (2023a) showed that LLMs could follow text generation constraints specified in natural language.

RL methods (Shinn et al., 2023). For future work, it would be interesting to compare the effectiveness and efficiency of both methods in a controlled setup.

(3) Challenges in making direct comparison. Yang et al. (2024) mainly uses `PaLM 2-L` model and `text-bison` model as the task model (scorer), and optimizes the prompt for up to 200 steps. In our work, we mainly use `text-davinci-003` and `GPT-4`, and optimize the prompt for 3 steps by default. Due to access and budget constraints, we are unable to make direct comparison with OPRO.

In addition to OPRO, several recent works have explored automatic prompt engineering using diverse strategies. PromptBreeder (Fernando et al., 2023) adopts a self-referential prompt evolution framework, employing mutation prompts (similar to the concept of "meta-prompts" discussed in this paper) to edit task prompts, and hyper-mutation prompts to edit mutation prompts. PromptAgent (Wang et al., 2024) adopts the Monte Carlo Tree Search algorithm that iteratively performs selection, expansion, simulation and back-propagation for strategic prompt editing. Evoke (Hu et al., 2024) introduces a collaborative approach where an LLM-reviewer and an LLM-author work together to refine the prompt using critical thinking. In parallel to these works, we focus on the design and evaluation of the *meta-prompt* in LLM-powered automatic prompt engineering in this paper.

Our work is also related to Self-Discover (Zhou et al., 2024), a framework for LLMs to compose reasoning structures, such as "break down into sub-tasks" for complex tasks. PE2 demonstrates task decomposition behaviors as discussed in §5.3 and Table 6, which can be seen as presence of rudimentary meta-reasoning capabilities in LLMs.

C.2 Discussion on using PE2 to optimize its own meta-prompt

Conceptually, PE2 may be applied to not only optimize prompts, but also meta-prompts. We can replace $p^{(t)}$ and $p^{(t+1)}$ in Eq. 2 with the meta-prompt p^{meta} directly to enable PE2 to optimize the meta-prompt. We believe this is an exciting direction to pursue. However, three challenges (and broader questions) arise if we pursue this direction, and we look forward to addressing these challenges in the future:

1. How to collect data for such a study? To ensure this meta-prompt is general we may need a large collection of tasks along with prompt optimization history associated with them. Creating a resource like this will be a large effort.
2. How to automatically optimize the meta-prompt when there are no ground truth labels for prompt engineering? Math problems have ground-truth answers so that PE2 can inspect them and provide feedback for prompt refinement. The task of prompt engineering does not have ground truth labels, and this potentially makes the meta-prompt optimization process more noisy.
3. It would be very costly to run and even evaluate a system like this. To *evaluate* one meta-prompt candidate and show it outperforms other meta-prompt candidates, we will need to use it for prompt optimization on various tasks. We would expect the *optimization* process of the meta-prompt to be a magnitude more costly.

D Additional Experiment Details

D.1 Prompt Search Algorithm

See Algorithm 1.

D.2 Controlling Prompt Length

By default the max length of prompts is set to be 50 tokens, following Zhou et al. (2023b). For counterfactual tasks, to allow more space to explain the counterfactual situations, the max length is set to be 200 tokens.

D.3 Infrastructure and Runtime

Infrastructure. We use OpenAI API[6] to access `text-davinici-003`, `gpt-3.5-turbo-instruct`, `gpt-4`, `gpt-4-turbo`. For prompt generalization experiments using `mistral-7b-instruct`,

[6] https://openai.com/blog/openai-api

Algorithm 1 Search Procedure

$P^{(0)} = P_{init}$ or $P^{(0)} = \mathcal{M}_{init}(x_1, y_1, ..., x_n, y_n; p^{init})$ ▷ Manual init. or induction init.
for $t = 0, ..., T-1$ **do**
  $P^{(t+1)} = \emptyset$
  for $p^{(t)} \in$ Select-Best$(\cup_{i=0}^{t} P^{(i)}, n)$ **do** ▷ Select best n prompts based on D_{dev}
    for $j = 1...m$ **do**
      $B =$ Sample(D_{train}) ▷ Sample a batch (random or failure examples)
      $p^{(t+1)} = \mathcal{M}_{optim}(p^{(t)}, B; p^{meta})$ ▷ New prompt proposal
      $P^{(t+1)} = P^{(t+1)} \cup \{p^{(t+1)}\}$
    end for
  end for
end for
return Select-Best$(\cup_{i=0}^{T} P^{(i)}, 1)$ ▷ Return the final best prompt based on D_{dev}

`mpt-7b-instruct` and `yi-6b`, we run experiments locally using one Nvidia RTX A6000 GPU and the vLLM toolkit (Kwon et al., 2023).

Runtime. One prompt optimization experiment using `gpt-4/text-davinici-003` as task/proposal model takes about 90 minutes. This is also subject to API rate limits.

Costs. When using `gpt-4/text-davinci-003` it costs about $25 USD for one prompt optimization experiment. In the later stage of this project, we use `gpt-4-turbo`/`gpt-3.5-turbo-instruct` which are newer and cheaper, and the cost is reduced to about $3 USD per experiment.

D.4 Tasks and Data

We summarize the dataset size and data split information in Table 12. We summarize the source and license information of the datasets in Table 13. To the best of our knowledge, our usage of these datasets are consistent with their intended use; the data we use do not contain personal or sensitive information. Most of the datasets are in English and not domain-specific.

Dataset	Subtasks	$\lvert T_{train}\rvert$	$\lvert T_{dev}\rvert$	$\lvert T_{test}\rvert$	# Random Samples
MultiArith (Roy and Roth, 2015)	-	100	100	400	1
GSM8K (Cobbe et al., 2021)	-	100	100	1319	1
Instruction Induction (Honovich et al., 2023)	14 Subtasks	100	20	100	5
Counterfactual Eval (Wu et al., 2024)	12 Subtasks	100	20	100	5
BIG-Bench Hard (BBH format used in Suzgun et al. (2023))	27 Subtasks	100	100	50	1
BIG-Bench Hard (Alternative format; see §A.2)	2 Subtasks	100	100	500	1

Table 12: Dataset sizes and data splits.

Dataset	License	Source
MultiArith (Roy and Roth, 2015)	Unknown	`https://github.com/wangxr14/Algebraic-Word-Problem-Solver/`
GSM8K (Cobbe et al., 2021)	MIT	`https://github.com/openai/grade-school-math`
Instruction Induction (Honovich et al., 2023)	Apache-2.0	`https://github.com/orhonovich/instruction-induction`
Counterfactual Eval (Wu et al., 2024)	Unknown	`https://github.com/ZhaofengWu/counterfactual-evaluation`
BIG-bench Hard (Suzgun et al., 2023)	Apache-2.0	`https://github.com/google/BIG-bench` (original) `https://github.com/suzgunmirac/BIG-Bench-Hard` (reformatted)

Table 13: License and Source of the datasets used in this study.

(1) Mathematical Reasoning. The MultiArith dataset (Roy and Roth, 2015) contains 600 examples. As our prompt optimization method requires a training set, we randomly split into 100/100/400 for train/dev/test. This creates a slight discrepancy when comparing the results with past reported results. We ensure our reproduction is fair across different methods by using this fixed split. The GSM8K dataset (Cobbe et al., 2021) has a provided test split (1319 examples). We randomly selected 200 examples for the original train split, and use 100 as D_{train} and 100 as D_{dev}.

Task	Instruction	Demonstration
Subtasks used in this work (14)		
Second Letter	Extract the first letter of the input word.	cat → a
Starting With	Extract the words starting with a given letter from the input sentence.	The man whose car I hit last week sued me. [m] → man, me
Negation	Negate the input sentence.	Time is finite → Time is not finite.
Antonyms	Write a word that means the opposite of the input word.	won → lost
Synonyms	Write a word with a similar meaning to the input word.	alleged → supposed
Membership	Write all the animals that appear in the given list.	cat, helicopter, cook, whale, frog, lion → frog, cat, lion, whale
Rhymes	Write a word that rhymes with the input word.	sing → ring
Informal to Formal	Rephrase the sentence in formal language.	Please call once you get there → Please call upon your arrival.
Translation EN-DE	Translate the word into German.	game → spiel
Translation EN-ES	Translate the word into Spanish.	game → jeugo
Translation EN-FR	Translate the word into French.	game → jeu
Sentiment	Determine whether a movie review is positive or negative.	The film is small in scope, yet perfectly formed. → positive
Sentence Similarity	Rate the semantic similarity of two sentences on a scale of 0 to 5	Sentence 1: A man is smoking. Sentence 2: A man is skating. → 0 - definitely not
Word in Context	Determine whether an input word has the same meaning in the two sentences.	Sentence 1: Approach a task. Sentence 2: To approach the city. Word: approach → not the same
Subtasks removed due to near-perfect accuracy (95%+) with baseline method (8)		
First Letter	Extract the first letter of the input word.	cat → c
List Letters	Break the input word into letters, separated by spaces.	cat → c a t
Singular to Plural	Convert the input word to its plural form.	cat → cats
Active to Passive	Write the input sentence in passive form.	The artist introduced the scientist. → The scientist was introduced by the artist.
Larger Animal	Write the larger of the two given animals.	koala, snail → koala
Sum	Sum the two given numbers.	22 10 → 32
Diff	Subtract the second number from the first.	32 22 → 10
Number to Word	Write the number in English words.	26 → twenty-six
Subtask removed due to small dataset size (2)		
Cause and Effect	Find which of the two given cause and effect sentences is the cause.	Sentence 1: The soda went flat. Sentence 2: The bottle was left open. → The bottle was left open.
Common Concept	Find a common characteristic for the given objects.	guitars, pendulums, neutrinos → involve oscillations.

Table 14: Details of Instruction Induction dataset. Adapted from Table 4 in Honovich et al. (2023).

(2) Instruction Induction. We closely follow the settings in Zhou et al. (2023b). For each subtask, we randomly sample 5 different $D_{train}/D_{dev}/D_{test}$ of size 100/20/100. We list the sub-tasks in Instruction Induction benchmark in Table 14. We removed 8 tasks (active to passive, diff, first word letter, letters list, num to verbal, singular to plural, sum), because our baseline method APE (Zhou et al., 2023b) already achieves near perfect accuracies (95%+) on these tasks. We also removed 2 tasks (cause and effect, common concept) because they have less than 50 examples in total, and it is challenging to create train/dev/test split from these examples.

(3) BIG-bench Hard Tasks. We mainly experiment with the BBH task format used in Suzgun et al. (2023). As the public BBH repository have 250 examples per task, we randomly split them into 100/100/50 for $D_{train}/D_{dev}/D_{test}$. For Date Understanding and Movie Recommendation, we consider using alternative tasks formats to study the their effect (see §A.2). We obtain the data from the original BIG-bench repository which contains more examples per task. Hence we randomly sample 100/100/500 examples for $D_{train}/D_{dev}/D_{test}$ in these two experiments.

(4) Counterfactual Evaluation. We use three subtasks in this evaluation suite: arithmetic, chess and syntax. For each subtask, we randomly sample 5 different $D_{train}/D_{dev}/D_{test}$ of size 100/20/100. We list the sub-tasks in Table 15.

Task	Category	Demonstration
Arithmetic - Two-digit addition		
Base-10	Original	22+10 → 32
Base-8	Counterfactual	76+76 → 174
Base-9	Counterfactual	76+14 → 101
Base-11	Counterfactual	76+14 → 8A
Base-16	Counterfactual	EC+DD → 1C9
Chess - Legality of a 4-move opening		
Normal Rules	Original	1. g3 Ng6 2. b3 Kf8 * → illegal
Swapping bishops and knights	Counterfactual	1. g3 Ng6 2. b3 Kf8 * → legal
Syntax - Identify the main subject and the main verb of a sentence		
SVO	Original	he has good control . → he has
SOV	Counterfactual	he good control has . → he has
VSO	Counterfactual	has he good control . → he has
VOS	Counterfactual	has good control he . → he has
OVS	Counterfactual	good control has he . → he has
OSV	Counterfactual	good control he has . → he has

Table 15: Details of Couterfactual Evaluation dataset (Wu et al., 2024).

(5) Production Prompt. We use a randomly sampled subset of human annotated queries and labels (> 150), which are derived from user reported errors. The data is divided between training (50%), validation (25%) and testing (25%). We use the F1-score for evaluating model outputs and report the absolute change in score with the initialization prompt.

E Additional Result Figures and Tables

Notable Prompt Edits. Additional examples on notable prompt edits made by PE2 are in Table 16.

Task	t	Prompt	Dev Acc.
Correct wrong or incomplete task instructions			
Antonyms	0	Write the opposite of the given word by adding an appropriate prefix.	0.3
	1	Find the opposite of the given word. If applicable, add or remove an appropriate prefix to form the opposite.	0.6
Provide more specific context and details			
Second Word Letter	0	Find the second letter in each word.	0.9
	1	Identify the second character in the provided word.	0.95
	2	Identify the second character from the start of the given word.	1.0
Sentence Similarity	0	Rate the similarity between Sentence 1 and Sentence 2 on a scale from 1 to 5, with 1 being 'probably not similar' and 5 being 'perfectly similar'.	0.0
	1	Rate the similarity between Sentence 1 and Sentence 2 as '1 - probably not similar', '2 - possibly', '3 - moderately', '4 - almost perfectly', or '5 - perfectly similar'.	0.15
Lay out tailored multi-step plans for complex problems			
Date Understanding	0	Let's think step by step.	0.39
	2	Analyzing the given information, let's calculate the solution. Remember to consider the context provided, such as references to 'today' or specific dates.	0.54
Produce short-cut solutions in counterfactual tasks			
Base-9 Addition (Induction Init.)	0	Add the numbers in each input together to get the output.	0.0
	1	Add the numbers in each input together and then add 11 to get the output.	0.2

Table 16: Notable prompt edits made by PE2 (Part 2; Continued from Table 6).

Results Breakdown. We report the results on each subtask in Instruction Induction in Fig. 10 and Table 17. For counterfactual tasks, results using induction initialization are in Fig. 11 and Table 18; results using manual initialization are in Fig. 12 and Table 19. For BIG-bench Hard tasks, we report the results in Fig. 13 and Table 20. We report the results on Date Understanding and Movie Recommendation when alternative task formats are used in Table 25.

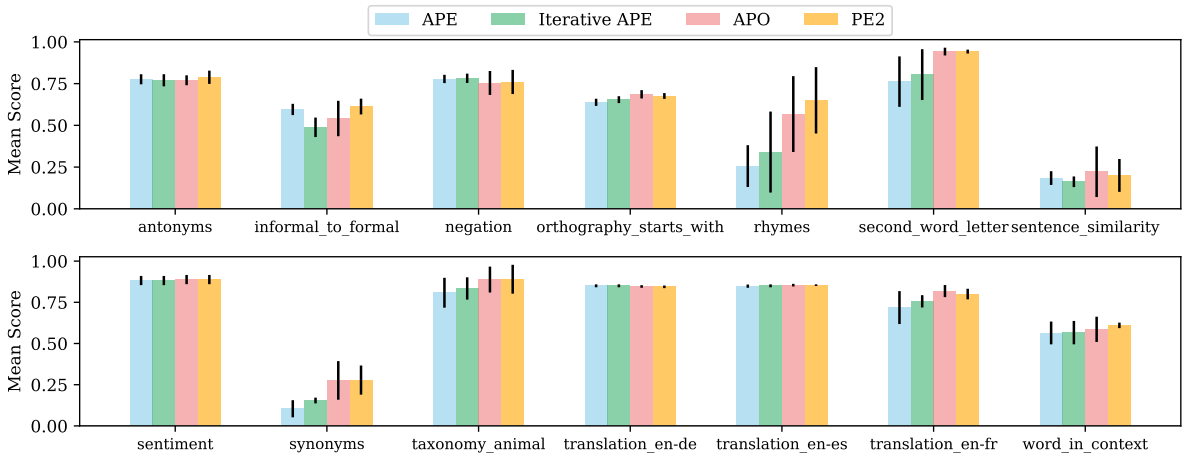

Figure 10: Results on the Instruction Induction Benchmark. The performance of APO and PE2 are close to each other on most tasks. Our hypothesis is that tasks in Instruction Induction Benchmark are relatively easier compared to the other benchmarks, leading to performance saturation. Raw results in Table 17.

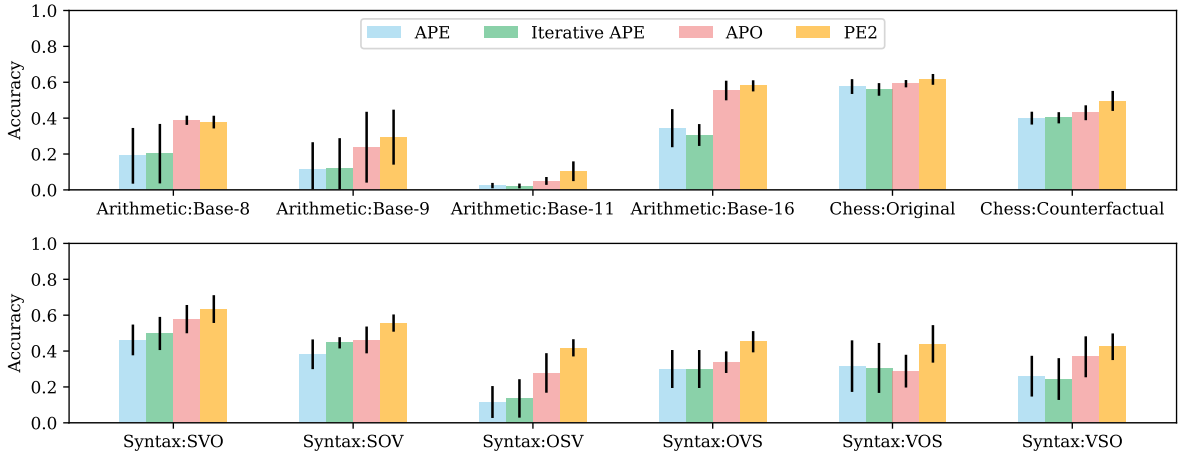

Figure 11: Results on Counterfactual Eval (Induction Initialization). Raw results in Table 18.

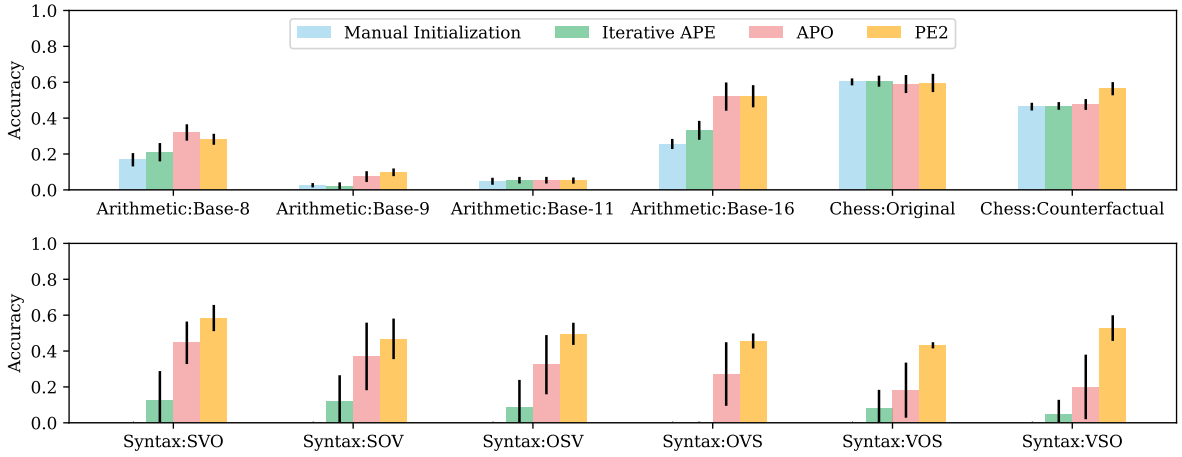

Figure 12: Results on Counterfactual Eval (Manual Initialization). Raw Results in Table 19.

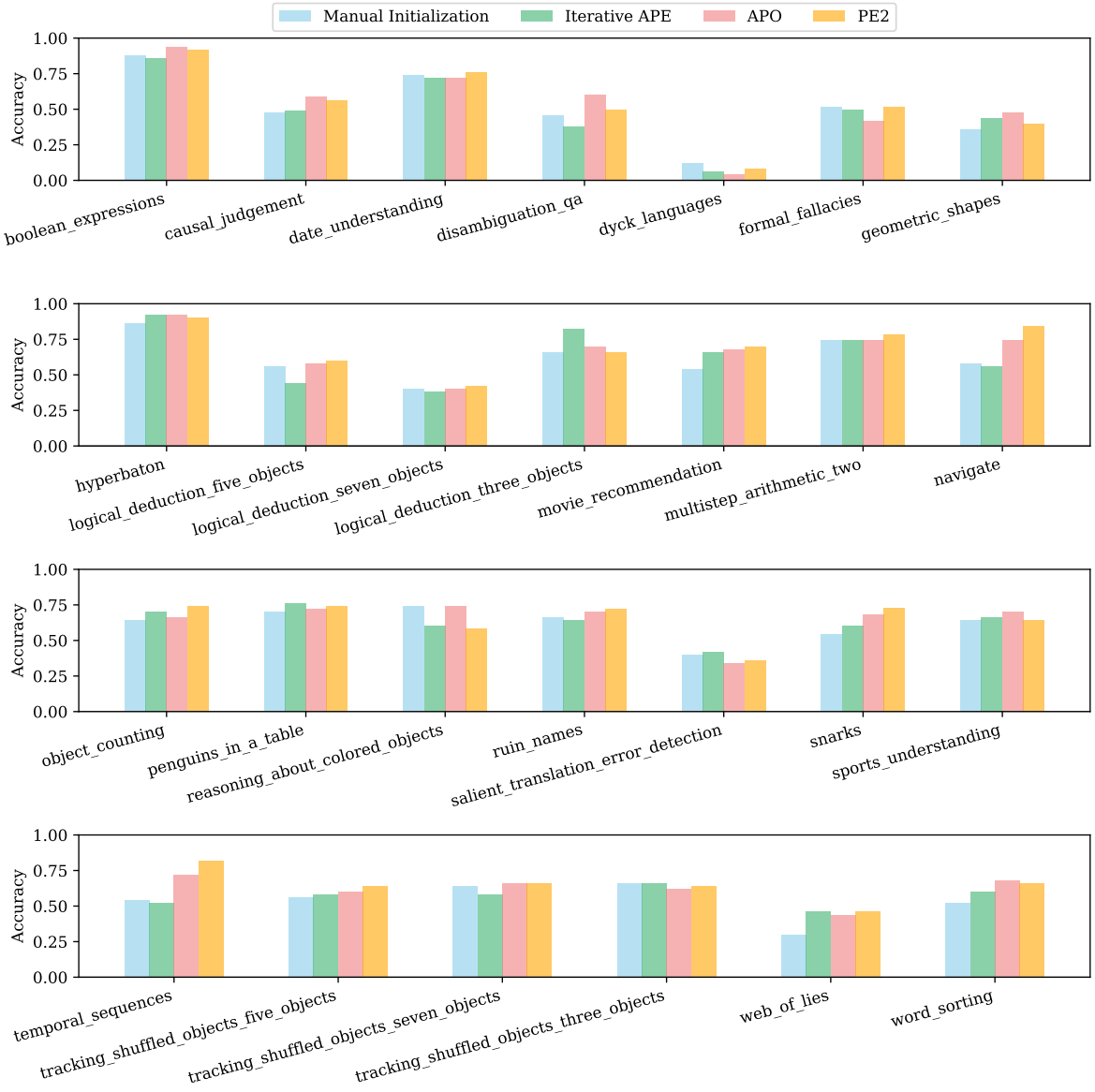

Figure 13: Results on the BIG-bench Hard (Suzgun et al., 2023). Raw Results in Table 20.

	APE		Iter. APE		APO		PE2	
	mean	std	mean	std	mean	std	mean	std
antonyms	77.60	3.01	77.00	3.63	77.00	2.97	78.80	3.97
informal_to_formal	59.53	3.37	48.83	5.83	54.10	10.61	61.26	4.73
negation	77.80	2.48	78.20	2.79	75.40	7.17	76.00	7.24
orthography_starts_with	63.80	2.14	65.40	2.06	68.60	2.50	67.60	1.74
rhymes	25.60	12.52	34.00	24.26	56.75	22.72	65.00	19.88
second_word_letter	76.20	15.12	80.40	15.23	94.20	2.32	94.20	1.17
sentence_similarity	18.40	4.13	16.20	3.19	22.20	15.14	20.00	9.84
sentiment	88.20	2.79	88.20	2.79	88.80	2.79	88.80	2.79
synonyms	10.40	5.20	15.40	1.74	27.60	11.71	27.80	8.84
taxonomy_animal	80.80	9.06	83.40	6.77	88.80	7.86	89.00	8.76
translation_en-de	85.00	0.89	85.00	0.89	84.60	0.80	84.40	0.80
translation_en-es	84.80	0.98	85.00	0.89	85.40	0.80	85.40	0.49
translation_en-fr	71.80	9.99	75.60	3.72	81.80	3.66	80.00	3.22
word_in_context	56.40	6.92	56.60	7.09	58.60	7.66	61.00	1.67
14-task average	62.60	-	63.52	-	68.85	-	69.95	-
(Δ with APE)	(+0.00)	-	(+0.92)	-	(+6.25)	-	(+7.35)	-

Table 17: Raw Results on Instruction Induction Benchmark. We report mean and standard deviation across 5 runs (5 different random samples of train and dev sets). The results for each task are visualized in Fig. 10 and the average results for 14 tasks are visualized in Fig. 2.

	APE		Iter. APE		APO		PE2	
	mean	std	mean	std	mean	std	mean	std
arithmetic_base8	19.00	15.53	20.20	16.57	38.80	2.56	37.80	3.54
arithmetic_base9	11.20	15.38	12.00	16.84	23.80	19.74	29.40	15.32
arithmetic_base11	2.40	1.50	2.20	1.33	5.00	2.19	10.40	5.50
arithmetic_base16	34.40	10.61	30.60	6.09	55.40	5.46	58.00	3.10
chess_original	57.60	4.13	56.00	3.52	59.20	2.04	61.60	3.01
chess_cf	40.00	3.58	40.20	3.12	43.00	4.15	49.60	5.57
syntax_svo	46.20	8.57	49.80	9.24	57.80	7.88	63.40	7.74
syntax_sov	38.20	8.28	44.60	3.14	46.20	7.47	55.60	4.80
syntax_osv	11.60	8.89	13.60	10.71	27.80	11.02	41.80	4.79
syntax_ovs	30.00	10.58	30.00	10.58	33.80	6.01	45.20	5.91
syntax_vos	31.60	14.37	30.60	13.92	28.80	9.13	44.00	10.45
syntax_vso	26.00	11.35	24.40	11.64	36.80	11.41	42.40	7.42
12-task average	29.02	-	29.52	-	38.03	-	44.93	-
(Δ with APE)	(+0.00)	-	(+0.50)	-	(+9.01)	-	(+15.91)	-

Table 18: Raw Results on Counterfactual Eval (Induction Initialization). We report mean and standard deviation across 5 runs (5 different random samples of train and dev sets). The results for each task are visualized in Fig. 11 and the average results for 12 tasks are visualized in Fig. 2.

	Initialization		Iter. APE		APO		PE2	
	mean	std	mean	std	mean	std	mean	std
arithmetic_base8	16.80	3.71	21.00	5.10	32.00	4.56	28.20	3.06
arithmetic_base9	2.60	1.20	2.00	2.19	7.40	3.01	9.80	2.14
arithmetic_base11	4.80	1.94	5.40	1.85	5.40	1.85	5.20	1.72
arithmetic_base16	25.60	2.80	33.20	5.27	52.00	7.87	52.20	6.18
chess_original	60.20	1.94	60.60	3.07	59.00	5.02	59.60	5.08
chess_cf	46.40	2.15	46.80	2.14	47.60	3.01	56.40	3.67
syntax_svo	0.00	0.00	12.80	16.03	44.60	11.84	58.40	7.31
syntax_sov	0.00	0.00	12.20	14.32	37.00	18.83	46.80	11.30
syntax_osv	0.00	0.00	8.80	15.12	32.40	16.50	49.60	6.18
syntax_ovs	0.00	0.00	0.20	0.40	27.20	17.72	45.60	4.18
syntax_vos	0.00	0.00	8.00	10.37	18.20	15.35	43.20	1.72
syntax_vso	0.00	0.00	4.60	8.21	20.00	17.98	52.80	7.14
12-task average	13.03	-	17.97	-	31.90	-	42.32	-
(Δ with Initialization)	(+0.00)	-	(+4.94)	-	(+18.87)	-	(+29.29)	-

Table 19: Raw Results on Counterfactual Eval (Manual Initialization). We report mean and standard deviation across 5 runs (5 different random samples of train and dev sets). We use "Let's think step by step" as the manual initialization prompt. The results for each task are visualized in Fig. 12.

	Init.	Iter. APE	APO	PE2
boolean_expressions	88.00	86.00	94.00	92.00
causal_judgement	48.28	49.43	58.62	56.32
date_understanding	74.00	72.00	72.00	76.00
disambiguation_qa	46.00	38.00	60.00	50.00
dyck_languages	12.00	6.00	4.00	8.00
formal_fallacies	52.00	50.00	42.00	52.00
geometric_shapes	36.00	44.00	48.00	40.00
hyperbaton	86.00	92.00	92.00	90.00
logical_deduction_five_objects	56.00	44.00	58.00	60.00
logical_deduction_seven_objects	40.00	38.00	40.00	42.00
logical_deduction_three_objects	66.00	82.00	70.00	66.00
movie_recommendation	54.00	66.00	68.00	70.00
multistep_arithmetic_two	74.00	74.00	74.00	78.00
navigate	58.00	56.00	74.00	84.00
object_counting	64.00	70.00	66.00	74.00
penguins_in_a_table	69.57	76.09	71.74	73.91
reasoning_about_colored_objects	74.00	60.00	74.00	58.00
ruin_names	66.00	64.00	70.00	72.00
salient_translation_error_detection	40.00	42.00	34.00	36.00
snarks	53.85	60.26	67.95	73.08
sports_understanding	64.00	66.00	70.00	64.00
temporal_sequences	54.00	52.00	72.00	82.00
tracking_shuffled_objects_five_objects	56.00	58.00	60.00	64.00
tracking_shuffled_objects_seven_objects	64.00	58.00	66.00	66.00
tracking_shuffled_objects_three_objects	66.00	66.00	62.00	64.00
web_of_lies	30.00	46.00	44.00	46.00
word_sorting	52.00	60.00	68.00	66.00
27-task average	57.17	58.36	62.23	63.09
(Δ with Initialization)	(+0.00)	(+1.19)	(+5.06)	(+5.92)

Table 20: Raw Results on BIG-bench Hard Tasks. The results for each task are visualized in Fig. 13 and the average results for 27 tasks are visualized in Fig. 2.

F Meta-prompts

We implement the meta-prompts using the guidance toolkit[7], which enables multi-round conversations and supports basic handlebars-style syntax to control the workflow.

F.1 Initialization Prompt p^{init}

The initialization prompt is originally from APE (Zhou et al., 2023b). In this paper, it is shared by all methods (Iterative APE, APO and PE2).

```
{{#system~}}
You are a helpful assistant.
{{~/system}}

{{#user~}}
I gave a friend an instruction and {{n_demo}} inputs. The friend read the instruction and wrote an
    output for every one of the inputs.
Here are the input-output pairs:

{{demos}}

What was the instruction? It has to be less than {{max_tokens}} tokens.
{{~/user}}

{{#assistant~}}
The instruction was {{gen 'instruction' [[GENERATION_CONFIG]]}}
{{~/assistant}}
```

F.2 APE

```
{{#system~}}
You are a helpful assistant.
{{~/system}}

{{#user~}}
Generate a variation of the following instruction while keeping the semantic meaning.

{{prompt}}

The new instruction has to be less than {{max_tokens}} words.
Reply with the new instruction. Do not include other text.
{{~/user}}

{{#assistant~}}
{{gen 'new_prompt' [[GENERATION_CONFIG]]}}
{{~/assistant}}
```

F.3 APO

Part 1 - Generating "gradients"

```
{{#system~}}
You are a helpful assistant.
{{/system~}}

{{#user~}}
I'm trying to write a zero-shot classifier prompt.

My current prompt is:
"{{prompt}}"

But this prompt gets the following examples wrong:
{{failure_string}}

Give {{n_reasons}} reasons why the prompt could have gotten these examples wrong. Do not include other
    text.
{{/user~}}

{{#assistant~}}
{{gen 'gradients' temperature=0.0}}
{{/assistant~}}
```

Part 2 - Refining the prompt

```
{{#system~}}
You are a helpful assistant.
{{/system~}}
```

[7] https://github.com/guidance-ai/guidance

```
{{#user~}}
I'm trying to write a zero-shot classifier.

My current prompt is:
"{{prompt}}"

But it gets the following examples wrong:
{{failure_string}}

Based on these examples the problem with this prompt is that:
{{gradient}}

Based on the above information, I wrote an improved prompt. The total length of the prompt should be less than {{max_tokens}} words.
{{/user~}}

{{#assistant~}}
The improved prompt is {{gen 'new_prompt' temperature=0.0}}
{{/assistant~}}
```

F.4 PE2

```
{{#system~}}
You are a helpful assistant.
{{~/system}}

{{#if instruction}}
{{#user~}}
Let's read a blogpost on prompt engineering:
{{instruction}}
{{~/user}}
{{/if}}

{{#user~}}
A prompt is a text paragraph that outlines the expected actions and instructs the model to generate a specific output. This prompt is concatenated with the input text, and the model then creates the required output.

In our collaboration, we'll work together to refine a prompt. The process consists of two main steps:

## Step 1
I will provide you with the current prompt, how the prompt is concatenated with the input text (i.e., "full template"), along with {{batch_size}} example(s) that are associated with this prompt. Each examples contains the input, the reasoning process generated by the model when the prompt is attached, the final answer produced by the model, and the ground-truth label to the input. Your task is to analyze the examples, determining whether the existing prompt is decsribing the task reflected by these examples precisely, and suggest changes to the prompt.

## Step 2
Next, you will carefully review your reasoning in step 1, integrate the insights to craft a new, optimized prompt. Optionally, the history of refinements made to this prompt from past sessions will be included. Some extra instructions (e.g., the number of words you can edit) will be provided too.
{{~/user}}

{{#assistant~}}
Sure, I'd be happy to help you with this prompt engineering problem.
Please provide me with the prompt engineering history, the current prompt, and the examples you have.
{{~/assistant}}

{{#user~}}
## Prompt
{{prompt}}

## Full Template
This describes how the prompt of interested is concatenated with the input text.
The prompt may appear before the input text, or after the input text.
Optionally the full template may contain other template information.
```
{{full_prompt}}
```

## Examples
{{examples}}

## Instructions
For some of these examples, the output does not match with the label. This may be due to the prompt being misleading or not describing the task precisely.

Please examine the example(s) carefully. Note that the ground-truth labels are __absolutely correct__, but the prompts (task descriptions) may be incorrect and need modification. For each example, provide reasoning according to the following template:

### Example <id>
```

```
Input: <input>
Output: <output>
Label: <label>
Is the output correct compared to the label: <yes or no, and your reasoning>
Is the output correctly following the given prompt: <yes or no, and your reasoning>
Is the prompt correctly describing the task shown by the input-label pair: <yes or no, and your reasoning>
To output the correct label, is it necessary to edit the prompt: <yes or no, and your reasoning>
If yes, provide detailed analysis and actionable suggestions to edit the prompt: <analysis and suggestions>
{{~/user}}

{{#assistant~}}
{{gen 'reasoning' temperature=0}}
{{~/assistant}}

{{#user~}}
Now please carefully review your reasoning in Step 1 and help with Step 2: refining the prompt.

{{#if history}}
## Prompt Refinement History from the Past
Note that higher accuracy means better. If some edits are useful in the past, it may be a good idea to make edits along the same direction.
{{history}}
{{/if}}

## Current Prompt
{{prompt}}

## Instructions
{{#if step_size}}
* You are allowed to change up to {{step_size}} words in the original prompt.
{{/if}}
{{#if max_tokens}}
* The total length of the prompt should be less than {{max_tokens}} words.
{{/if}}
* Please help edit the prompt so that the updated prompt will not fail on these examples anymore.
* Reply with the prompt. Do not include other text.
{{~/user}}

{{#assistant~}}
{{gen 'new_prompt' temperature=0.7 max_tokens=300}}
{{~/assistant}}

{{#if history}}
{{#user~}}
Now please summarize what changes you've made to the prompt, in the following format. Make sure the summariy is concise and contains no more than 200 words.

" * At step {{timestamp}}, the prompt has limitations such as <summary of limitations>. Changes to the prompt include <summary of changes>."

Reply with the summarization. Do not include other text.
{{~/user}}

{{#assistant~}}
{{gen 'new_history' temperature=0.7 max_tokens=200}}
{{~/assistant}}
{{/if}}
```

G Prompt Optimization Results

See Table 21-26.

Table 21: Prompts find by prompt optimization methods on math reasoning and instruction induction tasks. For instruction induction, experiments were run with 5 random data splits; In this table we report the prompts found in one run (seed=0).

Task	Method	Prompt
Math Reasoning		
MultiArith	Zero-shot CoT	Let's think step by step.
	APE	Let's work this out in a step by step way to be sure we have the right answer.
	Iterative APE	Let's proceed in a methodical, step-by-step manner.
	APO	Given the scenario, perform the necessary calculations step by step to find the final result. Consider all parts of the input and the sequence of events.
	PE2	Let's solve this problem by considering all the details. Pay attention to each piece of information, remember to add or subtract as needed, and perform the calculations step by step.
GSM8K	Zero-shot CoT	Let's think step by step.
	APE	Let's work this out in a step by step way to be sure we have the right answer.
	Iterative APE	Let's dissect this and tackle it gradually, one phase at a time.
	APO	Given the scenario, perform necessary calculations and provide a step-by-step explanation to arrive at the correct numerical answer. Consider all information provided.
	PE2	Let's solve the problem step-by-step and calculate the required total value correctly.
Instruction Induction		
antonyms	APO	Provide the opposite or a negative form of the given input word.
	PE2	Provide the opposite or a negative form of the given input word.
informal_to_formal	APO	Convert each sentence into a formal version, preserving the original structure, meaning, and tone. Avoid excessive formality, unnecessary changes, and maintain idiomatic expressions. Handle contractions appropriately.
	PE2	Please transform each sentence into a version that maintains the original meaning but is expressed in a more formal or polite manner.
negation	APO	Negate the statement given in the input.
	PE2	Negate the statement given in the input.
orthography_starts_with	APO	Identify the word or phrase in the sentence that starts with the given letter, considering the context and grammar. Include articles if they precede the word or phrase.
	PE2	Find the word or phrase in the sentence that starts with the given letter, and write it as the output.
rhymes	APO	Remove the first letter of the given word. Find a word that rhymes with the remaining part, has the same syllable count, and is not a derivative or the same as the original word.
	PE2	Generate a word that rhymes with the given word.
second_word_letter	APO	Identify the second character from the start in each input word and provide it as the output.
	PE2	Identify the second character from the start of the given word.
sentence_similarity	APO	Rate the similarity between Sentence 1 and Sentence 2 using the scale: 1 - 'probably not', 2 - 'possibly', 3 - 'probably', 4 - 'likely', 5 - 'perfectly'.
	PE2	Rate the similarity between Sentence 1 and Sentence 2 using the scale: 1 - 'probably not', 2 - 'possibly', 3 - 'probably', 4 - 'likely', 5 - 'perfectly'.
sentiment	APO	Determine if the given movie review statement is positive or negative.
	PE2	Determine if the given movie review statement is positive or negative.
synonyms	APO	Provide a single word that is closely related to the given input, considering its most common usage.
	PE2	Identify a word that is closely connected, in meaning or context, with the provided input word.
taxonomy_animal	APO	Remove all items from the list that are not animals.
	PE2	Remove all items from the list that are not animals.
translation_en-de	APO	Translate each English word into German.
	PE2	Translate each English word into German.
translation_en-es	APO	Provide the most commonly used Spanish translation for the given English word.
	PE2	Translate the given term from English to Spanish. Note that the translation may be a single word or a phrase.
translation_en-fr	APO	Provide the French equivalent for the given English word.
	PE2	Translate the following word from English to its most common equivalent in French.
word_in_context	APO	Determine if the word provided is used in the same sense/context in both sentences. If it is, write 'same.' If not, write 'not the same.'
	PE2	Determine if the word provided is used in the same sense/context in both sentences. If it is, write 'same.' If not, write 'not the same.'

Table 22: Prompts find by prompt optimization methods on Counterfactual Eval (Wu et al., 2024) using induction initialization (*i.e.*, the model is not informed of the counterfactual situation). Experiments were run with 5 random data splits; In this table we report the prompts found in one run (seed=0).

Task	Method	Prompt
Counterfactual Evaluation (Induction Initialization)		
arithmetic_base11	APO	Given two numbers in hexadecimal format (0-9, A-F), convert each number to decimal. Add the two decimal numbers together. Output the sum in hexadecimal format. If the sum exceeds the range of a single hexadecimal digit (0-F), represent it appropriately in hexadecimal. For example, if the input is 'A' and 'B', the output should be '15' as 'A' is 10 and 'B' is 11 in decimal, and their sum is 21 which is '15' in hexadecimal.
	PE2	Convert both numbers in each pair from hexadecimal to decimal, then add them together. Output the resultant sum in hexadecimal. For instance, if the input is A4+61, convert A4 and 61 to decimal (164 and 97 respectively), add them together to get 261, and convert this back to hexadecimal to get 105.
arithmetic_base16	APO	Given two hexadecimal numbers as input, add them together using base 16 arithmetic. The input hexadecimal numbers will be in uppercase and may have different number of digits. Align the numbers from right to left, similar to traditional addition, and handle any overflow or carry appropriately. Output the sum as an uppercase hexadecimal number.
	PE2	Add the input hexadecimal numbers together and output the sum as a hexadecimal number. For example, if the input is "44+E7", the output should be "12B", because the sum of hexadecimals 44 and E7 equals 12B in hexadecimal.
arithmetic_base8	APO	Given an input string containing two numbers separated by a '+', calculate the sum of these two numbers. Then, add 20 to this sum to get the output. For example, if the input is '22+47', first add 22 and 47 to get 69, then add 20 to 69 to get the final output of 89. Similarly, if the input is '74+26', first add 74 and 26 to get 100, then add 20 to 100 to get the final output of 120. The '+' symbol should be interpreted as an addition operator, and the order of operations should be to add the two numbers first, then add 20 to the sum. The input will always be formatted correctly, with no spaces or other characters around the '+' symbol.
	PE2	To find the correct output, first add the two numbers given as input. Once you have the sum of these two numbers, add an additional 22 to this sum. For example, if the input is "17+65", you should first add 17 and 65 to get 82, then add 22 to 82. The correct output in this case would be 104.
arithmetic_base9	APO	Add the numbers together.
	PE2	Add the two numbers given as input and then add 10 to the result to generate the output. For example, if the input is '25+18', the output should be '53' because 25 plus 18 equals 43, and adding 10 gives 53.
chess_cf	APO	Determine if the given sequence of chess moves, starting from the initial game position, is legal or not according to the standard rules of chess. Consider the unique movements and restrictions of each piece, the alternating turns of the players (white and black), and the entire game state up to the given point. Evaluate the sequence as a whole, not just individual moves. Note that the sequence ends with an asterisk (*).
	PE2	Please assess the legality of the following sequence of chess moves based on standard chess rules. If all moves are valid according to the rules of chess, indicate "Legal." If there is any move that violates standard chess rules, respond with "Illegal". For example, if the sequence is "1. e4 e5 2. Nf3 d6", your response should be "Legal". If the sequence is "1. e4 e5 2. Kf2", your response should be "Illegal" because the king cannot be exposed to check.
chess_original	APO	Determine if the given sequence of chess moves is legal or illegal.
	PE2	Determine if the given sequence of chess moves is legal or illegal.
syntax_osv	APO	Identify the main subject and verb in the sentence. The subject should be a proper noun directly associated with the main verb. Focus on the main clause that conveys the primary information. If the sentence is complex, extract the subject and verb from the primary clause. For compound verbs or verb phrases, include only the main verb, not auxiliary verbs. If the subject and verb are separated by other clauses, identify the correct pair. If the subject is implied, make a reasonable guess. Write the subject and verb as a pair in the output.
	PE2	Identify the subject and verb at the end of the sentence. The subject may not always be a proper noun. The verb should be in the present tense. Write them out as a pair in the output. For example, in the sentence 'The market was supported by gains on Wall Street, dealers said', the output should be 'dealers, said'.
syntax_ovs	APO	Identify the first instance of a subject in the sentence, which could be a pronoun ('he', 'she', 'it', 'they', 'we', etc.) or a noun/noun phrase. Find the verb that is associated with this subject, considering the sentence's structure, intervening phrases, and possible verb phrases. The verb may not directly follow the subject and could precede it. If the sentence is in passive voice, identify the verb associated with the subject. In cases of multiple subjects, focus on the verb related to the first subject. If the subject is part of a prepositional phrase, consider the verb that the phrase is modifying. Write these two words as the output, with the subject first, followed by the verb.
	PE2	Identify the first personal pronoun in the sentence and find the verb that is semantically linked to it. Write these two words as your output. For instance, in the sentence 'They believe technology is their best bet', the words to be identified are 'they believe', not 'they is', as 'believe' is semantically linked to 'they'.
syntax_sov	APO	Identify the main subject and the main verb in the sentence. Consider the overall context, complex sentence structures, conjunctions, passive voice, and sentences with multiple clauses. Output the main subject and the main verb together, as they appear in the input. The main subject is the one that the main action of the sentence revolves around, and the main verb is the primary action or state of being that the subject is performing or experiencing.
	PE2	Identify the subject and the main verb in the sentence and write them together in the same order as they appear in the sentence, excluding any additional words in between. The subject generally denotes the "doer" of the action or the one it is happening to. The main verb expresses the action or state of being. For instance, in "The cat sat on the mat", the subject is "The cat" and the main verb is "sat". So, the output should be "The cat sat". Ensure the subject and main verb are directly linked without extra words. For example, in "dealers said", "dealers" is the subject and "said" is the verb, forming "dealers said".

Continued on next page

Task	Method	Prompt
syntax_svo	APO	Your task is to identify the subject and the main verb of the primary clause in an input sentence. Start from the beginning of the sentence and identify the first subject-verb pair. Ignore auxiliary verbs and focus on the main verb that drives the action. If the sentence has multiple clauses, focus on the first one that forms a complete thought. Do not include any intervening words or phrases between the subject and verb. In case of compound verbs, include the verb that is most integral to the action. Ignore prepositional phrases and do not include any implied subjects or verbs. Your output should be concise, containing only the subject and the main verb.
	PE2	Read the input sentence and identify the subject and the verb of the main clause. Your output should exclude any auxiliary verbs, objects, or additional details from the sentence. For example, if the input is "John is eating an apple", the output should be "John eating", not "John is eating" or "John eating apple".
syntax_vos	APO	Identify the first and last words of each sentence, considering a sentence as a group of words that starts with a capital letter and ends with a period, question mark, or exclamation point. Ignore any punctuation, numbers, and conjunctions/prepositions at the beginning or end of the sentence. Write these two words in reverse order. If the sentence begins and ends with the same word, write it once. Treat compound words or phrases as single words. For example, 'uniroyal' and 'has' should be treated as 'uniroyal has'.
	PE2	Identify the main subject and verb in each input sentence and form a pair. The subject is usually a noun or pronoun that the verb refers to. The verb should be the main verb of the sentence, not an auxiliary verb. For example, if the input is "The cat chased the mouse.", the output should be "cat chased". If the input is "She has eaten the cake.", the output should be "She eaten", not "She has".
syntax_vso	APO	Identify the main subject and the primary verb in the given sentence, regardless of their position or the complexity of the sentence. Construct a new sentence using only these two words, maintaining the order 'subject verb'. Ignore additional information, context, or implied subjects/verbs. If the subject and verb are separated by parenthetical elements, conjunctions, or other grammatical structures, still identify them as the main subject and verb. Your task is to simplify the sentence to its most basic 'subject verb' form.
	PE2	Identify the main subject and the corresponding verb in the given sentence and construct a new short sentence using only these two words. The order should be 'subject verb'. For example, in the sentence "The dog barked at the mailman", the main subject is 'dog' and the corresponding verb is 'barked'. So, the new sentence would be "Dog barked".

Table 23: Prompts find by prompt optimization methods on Counterfactual Eval (Wu et al., 2024) using manual initialization. Experiments were run with 5 random data splits; In this table we report the prompts found in one run (seed=0).

Task	Method	Prompt
Counterfactual Evaluation (Manual Initialization)		
arithmetic_base11	Manual Init.	You are a mathematician. Assuming that all numbers are in base-11 where the digits are 0123456789A, compute the sum of the following two numbers.
	APO	You are a mathematician. Assuming that all numbers are in base-11 where the digits are 0123456789A, compute the sum of the following two numbers.
	PE2	You are a mathematician. Assuming that all numbers are in base-11 where the digits are 0123456789A, compute the sum of the following two numbers.
arithmetic_base16	Manual Init.	You are a mathematician. Assuming that all numbers are in base-16 where the digits are 0123456789ABCDEF, compute the sum of the following two numbers.
	APO	You are a mathematician working with base-16 (hexadecimal) numbers. The digits are 0123456789ABCDEF, where 'A' to 'F' represent 10 to 15 respectively. Add the two given hexadecimal numbers. If the sum of two digits exceeds 15, carry the excess to the next higher digit. For instance, 'F' + '2' equals '11' in base-16, which is '1' with a carryover of '1'. The input will be two hexadecimal numbers separated by a '+'. The output should be the sum in base-16."
	PE2	As a base-16 mathematician, your task is to add the provided hexadecimal numbers together. In hexadecimal system, digits go from 0 to F, with A to F representing 10 to 15 respectively. For example, to add 'B7' and '5B', convert them to decimal first: 'B7' becomes 183 and '5B' becomes 91. Their sum, 274, is '112' in hexadecimal.
arithmetic_base8	Manual Init.	You are a mathematician. Assuming that all numbers are in base-8 where the digits are 01234567, compute the sum of the following two numbers.
	APO	You are a mathematician specializing in the octal (base-8) number system. Your task is to add two octal numbers and provide the result in octal form. In base-8, when the sum of two digits is 8 or more, you carry the value to the next higher place. For example, 7+1 in base-8 is 10. Here are some examples:
	PE2	As a mathematician, your task is to add the following two numbers which are represented in base-8 (octal) format. The base-8 system uses digits from 0 to 7. Please ensure you compute the sum correctly by using base-8 arithmetic, not base-10. For example, in base-8, 7+1 equals 10, not 8. Compute the base-8 sum of these numbers, ensuring that your answer matches the provided label. For instance, if the input is "25+55", the correct output would be "102". Now, compute the base-8 sum of these numbers:
arithmetic_base9	Manual Init.	You are a mathematician. Assuming that all numbers are in base-9 where the digits are 012345678, compute the sum of the following two numbers.
	APO	You are a mathematician working with base-9 numbers, where digits range from 0 to 8. Your task is to add two base-9 numbers. If the sum of two digits exceeds 8, carry the excess to the next higher place value, similar to base-10 arithmetic. For instance, '8+1' in base-9 equals '10'. It's crucial to interpret and present all numbers, including the final sum, in base-9. For example, if you're adding '16' and '24' in base-9, the correct sum is '41', not '40'. Now, compute the sum of the following two base-9 numbers.

Continued on next page

Task	Method	Prompt
	PE2	You are a mathematician. Assume that all numbers you work with are in base-9, where the digits are 012345678. Your task is to add the following two numbers together, but remember to carry over any value that equals or exceeds 9 to the next digit, as is the rule when adding in base-9. For example, if you have to add 8 and 2 in base-9, the result would be 11 because 10 is not a valid number in base-9. Now, compute the sum of the following two numbers.
chess_cf	Manual Init.	You are a chess player. You are playing a chess variant where the starting positions for knights and bishops are swapped. For each color, the knights are at placed that where bishops used to be and the bishops are now placed at where knights used to be. Given an opening, determine whether the opening is legal. The opening doesn't need to be a good opening. Answer "legal" if all moves are legal. Answer "illegal" if the opening violates any rules of chess.
	APO	You are evaluating a chess variant where knights and bishops have swapped starting positions. Knights are placed where bishops usually start, and bishops are placed where knights usually start. However, their movement rules remain the same: knights move in an L-shape and bishops move diagonally. Your task is to determine the legality of a given opening. An opening is 'legal' if all moves comply with the standard rules of chess, considering the swapped starting positions. If all moves are legal, answer 'legal'. If any move violates the chess rules, answer 'illegal'. The opening doesn't need to be a good strategy, it just needs to be legal.
	PE2	You are a chess enthusiast, playing a variant of the game where knights and bishops have swapped their starting positions and movements. The knights, now placed where the bishops were, move as bishops. The bishops, positioned where knights were, move as knights. Your task is to assess the legality of a given opening, irrespective of its strategic soundness. Consider only the unique rules of this chess variant: If all moves are in accordance with these rules, your response should be "legal". However, if any move contravenes these rules, respond with "illegal". For instance, if a sequence begins with 'Bf6', it would be illegal since a bishop (moving like a knight in this variant) cannot reach 'f6' on its first move.
chess_original	Manual Init.	You are a chess player. Given an opening, determine whether the opening is legal. The opening doesn't need to be a good opening. Answer "legal" if all moves are legal. Answer "illegal" if the opening violates any rules of chess.
	APO	You are a chess expert. Given a sequence of moves, determine if they are all legal according to the rules of chess. Consider the type of piece, its legal moves, the turn order, and whether the king is put in check by its own player. If all moves are legal, answer "legal". If any move violates the rules of chess, answer "illegal". Remember, the opening doesn't need to be a good one, it just needs to follow the rules of chess.
	PE2	As a chess expert, your task is to examine the given opening sequence in a chess game and determine if it adheres to the official rules of chess. Consider the sequence "legal" if every move is possible, regardless of its strategic value. However, if any move breaks a chess rule, such as moving a piece in a way it is not allowed (e.g., a knight moving like a bishop), classify the sequence as "illegal". Your response should be one of two words: "legal" or "illegal".
syntax_osv	Manual Init.	You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the object-subject-verb order instead of the subject-verb-object order. Your task is to identify the main verb and the main subject in a sentence in this imaginary language. Show the main verb (a single word) and its subject (also a single word).
	APO	You are a linguistics expert. Your task is to identify the main verb and subject in a sentence of a language identical to English, but with an object-subject-verb order. The main verb is the primary action word, excluding auxiliary verbs. The main subject is the primary entity performing the action. In complex sentences, focus on the main clause. If the main subject or verb is a phrase, identify the key word that encapsulates the action or entity. If the main subject or verb is a proper noun, treat it as a single word. Your output should be a phrase consisting of the main subject and verb. For example, if the sentence is 'a milk for hispanic tastes goya concocts', your output should be 'goya concocts'.
	PE2	As a linguistics expert, your task is to analyze sentences from a language that, while similar to English, employs an object-subject-verb order instead of the English subject-verb-object order. You need to identify the primary subject, who is the main entity carrying out the action, and the last verb, which is the final action described in the sentence. Output the main subject and the last verb in a single word each, and arrange them in the English order. For instance, for "apple the eats boy", your output should be "boy eats". Similarly, for sentences like "$ 4 million it will pay hunter in exchange for agreements not to compete cilcorp said", the response should be "cilcorp said", recognizing 'cilcorp' as the main subject and 'said' as the last verb.
syntax_ovs	Manual Init.	You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the object-verb-subject order instead of the subject-verb-object order. Your task is to identify the main verb and the main subject in a sentence in this imaginary language. Show the main verb (a single word) and its subject (also a single word).
	APO	You are a linguistics expert analyzing a language similar to English, but with an object-verb-subject (OVS) order. Your task is to identify the main verb and the main subject in a sentence. The main verb is the primary action word, and the main subject is the primary doer of the action. They may not always be adjacent. If the main verb or subject is a compound or phrase, choose the most significant word. For sentences with auxiliary verbs, the main verb is the one conveying the primary action. After identifying, reverse the order to subject-verb for your output. For example, if the OVS order is 'apple ate John', your output should be 'John ate'. Remember, your output should always be in subject-verb order.
	PE2	You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the object-verb-subject order instead of the subject-verb-object order. Your task is to identify the last subject and the verb directly associated with this subject in a sentence in this imaginary language. Show the subject first (a single word) and then the verb (also a single word). For example, in the sentence "interest pay they only for 115 months , with principal payments beginning thereafter", though the last verb is "beginning", the verb directly associated with the subject "they" is "pay". Therefore, the answer is "they pay".
syntax_sov	Manual Init.	You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the subject-object-verb order instead of the subject-verb-object order. Your task is to identify the main verb and the main subject in a sentence in this imaginary language. Show the main verb (a single word) and its subject (also a single word).

Continued on next page

Task	Method	Prompt
	APO	You are a linguistics expert. Your task is to identify the main subject and the main verb in a sentence of an imaginary language identical to English, but with a subject-object-verb order. Your output should be in the original English order (subject-verb). Choose the most crucial word if the subject or verb is a phrase. Ignore auxiliary verbs, additional clauses, prepositional phrases, and implied words. Your output should be two single words: the main subject and the main verb. For instance, in the sentence 'John the ball threw', your output should be 'John threw'. In complex sentences, focus on the primary clause. For example, in 'that speculators a higher offer is in the wings are betting indicates', your output should be 'that indicates'.
	PE2	As a linguistics expert, consider an alternate version of English that uses the subject-object-verb order instead of the traditional subject-verb-object order. Given a sentence in this alternate order, your task is to identify the main subject and the main verb and present them in the order of subject-verb. Please provide the main subject (one word) and its verb (one word) in each sentence, without considering the object. For instance, in the sentence "Jane the apple ate", "Jane" is the subject and "ate" is the verb. Therefore, the answer would be "Jane ate".
syntax_svo	Manual Init.	You are an expert in linguistics. Your task is to identify the main verb and the main subject in a sentence. Show the main verb (a single word) and its subject (also a single word).
	APO	You are a language analyst. Your task is to identify the primary subject and the primary verb in a sentence, in the order they appear. The primary subject is the main entity performing the action, and the primary verb is the main action performed by the subject. They should be part of the same clause. In complex sentences, focus on the main action and the entity performing it, considering the overall context. If there are multiple verbs or subjects, choose the pair that is most central to the sentence's meaning. Ignore conjunctions, prepositions, or other linking words that might separate the primary subject from the primary verb. If the primary subject or verb is implied, infer it from the context. Provide the primary subject and verb as a single output, with the subject first and the verb second. Both should be single words. Do not include punctuation in your output.
	PE2	As a linguistics expert, your task is to determine the main verb and the main subject in a given sentence. Identify them as a single word each. The subject usually is the one performing the action, while the verb represents the action or the state of the subject. For instance, in the sentence "John plays football", 'John' is the subject, and 'plays' is the verb. Please provide the subject first, followed by the verb.
syntax_vos	Manual Init.	You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the verb-object-subject order instead of the subject-verb-object order. Your task is to identify the main verb and the main subject in a sentence in this imaginary language. Show the main verb (a single word) and its subject (also a single word).
	APO	You are a linguistics expert. Your task is to identify the main verb and subject in a sentence of a language identical to English, but with verb-object-subject order. Focus on the verb and subject that carry the main action or idea. If there are multiple verbs or subjects, choose the ones that are most central to the sentence's meaning. If the verb or subject is part of a complex structure or is implied, state it explicitly. If the verb or subject is a phrase, identify the entire phrase. Your output should be in the format: 'Subject Verb'. Remember, the subject and verb may not be adjacent or single words. Use your linguistic expertise to determine the main verb and subject.
	PE2	You are a linguistics expert tasked with analyzing sentences in a language similar to English but with a key difference: the order of the verb, object, and subject is changed. Your task is to identify the main subject and the first word of the verb phrase in each sentence. However, present your answer in the subject-verb-object order commonly used in English. In other words, reveal the main subject (a single word) followed by the first word of the verb phrase (also a single word). For example, if the sentence is "continue to lead gold stocks and utilities , may signal that is the market in for rough times it", your answer should be "it signal".
syntax_vso	Manual Init.	You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the verb-subject-object order instead of the subject-verb-object order. Your task is to identify the main verb and the main subject in a sentence in this imaginary language. Show the main verb (a single word) and its subject (also a single word).
	APO	You are a language expert analyzing a unique language similar to English, but with verb-subject-object order. Your task is to identify the main verb and subject in a sentence. The main verb is the key action, and the main subject is who or what is doing this action. In complex sentences, focus on the most important action. If multiple verbs or subjects exist, choose the most central to the sentence's meaning. Treat auxiliary or compound verbs as one unit with their main verb. Your output should be the main subject followed by the main verb (both as single words)."
	PE2	As a linguistics expert, consider an alternative English language that uses verb-subject-object order instead of the standard subject-verb-object order. Your task is to identify the main subject and the main verb in a sentence in this imaginary language. Display the main subject (a single word) followed by its verb (also a single word). For instance, if the input is "compares that with 3.5 % butterfat for whole milk", the output should be "that compares". Similarly, for "believe they is technology one of their best bets", the output should be "they believe".

Table 24: Prompts find by prompt optimization methods on BIG-bench Hard tasks. Here we use the task formats in (Suzgun et al., 2023). In these experiments, we use `gpt-3.5-turbo-instruct` as the task model and `gpt-4` as the prompt proposal model. See Table 25 when alternative formats are used.

Task	Method	Prompt
BIG-bench Hard (Manual Initialization - "Let's think step by step.")		
boolean_expressions	Iterative APE	Consider approaching this methodically, one step at a time.
	APO	First, resolve 'not' immediately before it. Then evaluate expressions within parentheses. Next, perform 'and' operations, and finally 'or'. Determine the truth value step by step.
	PE2	Evaluate the logical expression step by step following precedence: 'not', 'and', 'or'. Conclude by affirming the truth of the full statement.
causal_judgement	Iterative APE	Approach each stage sequentially.

Continued on next page

Task	Method	Prompt
	APO	Identify the direct cause of the outcome: was it the immediate action or condition without which the event wouldn't have occurred?
	PE2	Determine if the action was intentional and a contributing factor to the outcome. Answer 'Yes' if intentional and causative, 'No' otherwise.
disambiguation_qa	Iterative APE	Consider each phase one by one.
	APO	In each sentence, determine the pronoun's antecedent based on typical roles and actions. If unclear, mark as ambiguous."
	PE2	Let's think step by step, using grammatical structure and context to identify the antecedent of the pronoun or determine if it is ambiguous.
date_understanding	Iterative APE	Consider the process in a sequential manner.
	APO	Let's think step by step.
	PE2	Determine the date referred to as 'today' from the context given, then calculate the specified date step by step in MM/DD/YYYY format.
dyck_languages	Iterative APE	Let's think step by step.
	APO	Output the sequence of closing brackets needed to properly close all open brackets in the given order: parentheses '()', square '[]', curly '', and angle '<>'. Ignore all other characters.
	PE2	Given an incomplete sequence of brackets, output only the missing closing brackets required to complete the sequence. Exclude any part of the original sequence.
formal_fallacies	Iterative APE	Consider each phase one at a time.
	APO	Determine if the conclusion necessarily follows from the premises using deductive logic. Identify any logical fallacies or overlooked counterexamples. Is the argument valid or invalid?
	PE2	To assess if an argument is deductively valid, examine if the conclusion logically follows from the premises without exception.
geometric_shapes	Iterative APE	Consider our actions in a sequential manner.
	APO	Classify the shape formed by the SVG path's 'd' attribute, analyzing vertices, line segments, and closure to distinguish polygons and figures.
	PE2	Analyze the SVG path commands to determine the shape they draw. Consider each command and visualize the path step by step.
hyperbaton	Iterative APE	Approach this gradually, step by step.
	APO	Choose the sentence with adjectives in the correct order: opinion, size, age, shape, color, origin, material, purpose, noun."
	PE2	Let's think step by step, considering the standard order of adjectives in English: opinion, size, age, shape, color, origin, material, purpose.
logical_deduction _five_objects	Iterative APE	Let's proceed with a methodical, step-by-step approach.
	APO	Let's think step by step.
	PE2	Let's think step by step.
logical_deduction _seven_objects	Iterative APE	Consider each phase one at a time.
	APO	Arrange seven objects in order using the clues provided. Identify the correct position for each object from the options listed, ensuring logical consistency."
	PE2	Let's analyze the information provided step by step, ensuring each deduction follows logically from the statements given to arrive at the correct answer.
logical_deduction _three_objects	Iterative APE	Let's proceed with a methodical, stepwise approach.
	APO	Let's think step by step.
	PE2	Let's carefully analyze the information to determine the accurate ranking of objects based on their attributes.
multistep_arithmetic _two	Iterative APE	Approach this step-by-step, tackling each phase sequentially.
	APO	Let's think step by step.
	PE2	Let's meticulously solve the math problem by simplifying each part of the equation, verifying our calculations at every step before moving forward.
movie _recommendation	Iterative APE	Consider each phase methodically.
	APO	Select the movie that best matches the reference in terms of genre and cultural significance, ignoring minor details. Choose the closest match:"
	PE2	Evaluate the given movies for genres, themes, narrative styles, tone, and characters to select the most similar one from the options.
navigate	Iterative APE	Let's carefully assess each stage of our plan in sequence.
	APO	Calculate the net distance from the starting point after following these step-by-step instructions, considering 'forward' as positive and 'backward' as negative movement.
	PE2	Let's calculate each movement's effect on our position. Start at zero and add or subtract steps as instructed, considering the direction each time to ensure accuracy.

Continued on next page

Task	Method	Prompt
object_counting	Iterative APE	Let's continue by taking systematic, sequential steps.
	APO	Let's think step by step.
	PE2	Let's identify and count the instances of the specified category of items mentioned, tallying multiples, to determine their total quantity.
penguins_in_a_table	Iterative APE	Consider the procedure progressively.
	APO	Let's think step by step.
	PE2	Let's think step by step.
reasoning_about _colored_objects	Iterative APE	Consider each phase methodically.
	APO	Let's think step by step.
	PE2	Let's think step by step and pay close attention to details such as colors and quantities.
ruin_names	Iterative APE	Let's proceed with our tasks one by one.
	APO	Identify the funniest edit of the given name by selecting the option that best incorporates a pun or playful twist related to the original.
	PE2	Identify the most humorous edit by considering only puns or clever wordplay that creates a witty variation of the original name, excluding misspellings or simple pluralizations.
salient_translation _error_detection	Iterative APE	Let's take this gradually, step by step.
	APO	Let's think step by step.
	PE2	Let's analyze the source and translation for errors. Check if there are any changes or inaccuracies in Named Entities, Numerical Values, Modifiers, Negation, Facts, or missing details. Identify which type of error occurs.
snarks	Iterative APE	Evaluate each stage sequentially.
	APO	Identify the sarcastic statement by considering the reversal of expectations and societal norms. Look for irony and implied meanings contrary to the literal words.
	PE2	To identify which statement is sarcastic, consider that sarcasm often means saying the opposite of what's true in a mocking way.
sports_understanding	Iterative APE	Let's think step by step.
	APO	Determine if the sentence is plausible: Match the athlete's known sport, current activity status, and sport-specific terms to assess accuracy in context.
	PE2	Let's think step by step.
temporal_sequences	Iterative APE	Consider each stage methodically.
	APO	Identify the time period when the person was not seen and the location was open. Exclude times when the person was observed elsewhere or the location was closed.
	PE2	Analyze the timeline to pinpoint time slots when the individual was not seen, which indicate when events could have occurred.
tracking_shuffled _objects_five_objects	Iterative APE	We'll tackle this systematically, one stage at a time.
	APO	Track ball swaps and position changes separately. List each swap, update positions and ball ownership after each, and determine final states for both.
	PE2	Let's carefully track each player's position swaps step by step to determine their final positions.
tracking_shuffled _objects_seven_objects	Iterative APE	Approach each stage with systematic thought.
	APO	Let's think step by step.
	PE2	Let's carefully track each book exchange step by step to determine the final owner of each book.
tracking_shuffled _objects_three_objects	Iterative APE	Reflect on each stage individually.
	APO	Let's think step by step.
	PE2	Let's analyze each position swap in sequence to determine the final positions. Confirm the last known positions of all players before concluding.
web_of_lies	Iterative APE	Let's proceed with a methodical, stepwise approach.
	APO	Given a sequence of people's statements about others' truthfulness, determine the truth status of the final person. Assume the first statement's truth is known. Apply logical negation for each liar's statement.
	PE2	To determine who is truthful, invert the claim of a liar and trust a truthful person's claim. Apply this logic until the last claim.
word_sorting	Iterative APE	Progress through each stage sequentially.
	APO	Sort words alphabetically, ignoring case. Exclude 'List:' label. For same first letters, sort remaining letters. Output in lowercase.
	PE2	Let's analyze and sort words. Ignore words that are part of instructions, like "List:", and then arrange the remaining words in alphabetical order.

Method	Final Prompt	Test Acc.
Date Understanding (Generative)		
Zero-shot CoT	Let's think step by step.	0.391
Iterative APE	Let's dissect it and ponder over each phase.	0.467
APO	Determine the exact date from the scenario, considering cultural date formats, time zones, and periods. Use the provided date as a reference. Account for leading zeros, leap years, relative dates, and event-based time references. Provide the result in MM/DD/YYYY format.	0.450
PE2	Analyzing the given information, let's calculate the solution. Remember to consider the context provided, such as references to 'today' or specific dates.	0.544
Movie Recommendation (Multi-choice 2)		
Zero-shot CoT	Let's think step by step.	0.570
Iterative APE	Let's dissect it and consider every step in order.	0.673
APO	Identify the movie that shares the most significant themes and narrative structure with the given movies. Prioritize these factors over tone and pacing. Choose the most similar movie from the options, explaining your choice.	0.750
PE2	Considering factors such as genre, director, actors, release period, audience target, animation style, and humor, analyze the similarities among the given movies and identify the movie from the options that shares the most similarities.	0.790

Table 25: Results on Date Understanding and Movie Recommendation from BIG-bench Hard (Suzgun et al., 2023). In these experiments, we use a format different from those in (Suzgun et al., 2023). See §A.2 for detailed discussion on the effect of task format. We use `gpt-3.5-turbo-instruct` as the task model and `gpt-4` as the prompt proposal model.

Method	Final Prompt	Test Acc.
GSM8k		
Zero-shot CoT	Let's think step by step.	0.481
Iterative APE	We'll approach this methodically, proceeding one step at a time.	0.497
APO	Carefully analyze the details and perform precise arithmetic operations step by step, ensuring to apply the correct mathematical principles for accurate calculations.	0.510
PE2	Carefully calculate step by step, considering all details, including unique counts and overlaps. Accurately apply arithmetic to find the numerical answer, ensuring logical operations are correctly followed.	0.505
MultiArith		
Zero-shot CoT	Let's think step by step.	0.715
Iterative APE	Proceed gradually, one step at a time.	0.735
APO	Calculate the answer using arithmetic. Round down where necessary. Correct any logical errors in reasoning. Provide the exact number.	0.735
PE2	Focus on accurately calculating totals and differences, considering factors like item conditions or groupings for precision. Round only if necessary, when dealing with practical fractions.	0.743
BIG-bench Hard - Date Understanding		
Zero-shot CoT	Let's think step by step.	0.36
Iterative APE	Let's move forward by dividing it into manageable steps.	0.48
APO	Accurately calculate dates from given inputs, considering the current reference point (e.g., 'yesterday', 'today') and applying correct calendar arithmetic, including month lengths and leap years. Ignore irrelevant details and assumptions beyond provided information.	0.48
PE2	Let's think step by step. For date questions, calculate from the given date, then choose the closest matching option. Ignore irrelevant info.	0.56
BIG-bench Hard - Hyperbaton		
Zero-shot CoT	Let's think step by step.	0.52
Iterative APE	Let's systematically go through every step.	0.48
APO	Classify adjective order: opinion, size, age, shape, color, origin, material, purpose. For words fitting multiple categories, prioritize purpose, then material. Ignore rare exceptions.	0.72
PE2	Choose the correct sentence using adjective order: opinion, size, age, shape, color, origin, material, purpose. Note: 'purpose' adjectives, like 'hiking', often come last.	0.74
BIG-bench Hard - Temporal Sequences		
Zero-shot CoT	Let's think step by step.	0.50
Iterative APE	Let's progress by breaking it down into smaller, manageable parts.	0.42
APO	Identify when a person visited a place by excluding times they were seen elsewhere, considering their schedule, eyewitness sightings, and the place's hours. Only include unaccounted times.	0.52
PE2	Let's analyze the timeline and others' observations to deduce the only time slots not accounted for, indicating when the visit could have occurred.	0.62
BIG-bench Hard - Word Sorting		
Zero-shot CoT	Let's think step by step.	0.04
Iterative APE	Let's tackle this systematically, advancing step by step.	0.20
APO	Alphabetically sort the words below, correcting any typos. Ignore capitalization, treat abbreviations and possessives normally. Exclude 'List:' from items.	0.16
PE2	Sort the given words alphabetically, but exclude 'List:' or similar formatting elements. Ensure every word is considered.	0.28

Table 26: Results on six selected tasks. We use `Mistral-7B-Instruct-v0.2` as the task model and `gpt-4-turbo` as the prompt proposal model.