

Prompting to Prompt: Meta-Template Learning for Transferable Prompt Optimization

Xinyu Zhang^{1*}, Deyang Li^{2*}, Minjun Zhao², Meng Han², Ruifeng Shi², Zhicheng Dou^{1†}

¹Gaoling School of Artificial Intelligence, Renmin University of China

²Huawei Poisson Lab, China

{zhangxinyu1995, dou}@ruc.edu.cn

{lideyang2, zhaominjun1, hanmeng22, shiruifeng}@huawei.com

Abstract

Existing prompt optimization methods often improve prompts for a specific task or deployment setting, but their optimized prompts transfer poorly to new tasks and usually require re-optimization. We propose Prompting to Prompt (PTP), a framework that learns meta-templates as structured intermediate representations for reusable prompt construction. PTP decomposes prompts into transferable elements and optimizes them with LLM-generated textual gradients. Its inner loop refines task-specific prompts from natural-language feedback and an element list, while its outer loop abstracts structural changes across tasks to update the meta-template. The learned meta-template is then reused for task-level and query-level prompt generation, reducing the need to rerun full prompt optimization for each target setting. Experiments on six benchmarks show gains in both offline transfer and test-time deployment settings, with improvements up to 10.52 points on Arena-Hard.

1 Introduction

Large language models (LLMs) excel in various NLP tasks (Zhao et al., 2023; Bubeck et al., 2023; Li et al., 2025a; Brown et al., 2020), but traditional task-specific model tuning is computationally expensive and impractical for large models. Prompt engineering offers an efficient alternative (Shin et al., 2020; Tang et al., 2025; Trivedi et al., 2025), yet minor prompt variations can drastically affect outputs (Lu et al., 2022; Sclar et al., 2024; Zheng et al., 2024a; He et al., 2025; Madaan et al., 2023). Manual prompt design often relies on extensive trial-and-error and domain expertise, making it resource-intensive and challenging to scale. To address these limitations, automated prompt engineering has been proposed to replace manual efforts, enabling LLMs to generate more effective

responses (Cui et al., 2025; Yan et al., 2025; Agarwal et al., 2024; Ramnath et al., 2025).

While some prompt optimization methods rely on access to internal state variables of the LLM (Lester et al., 2021; Shin et al., 2020), practitioners typically interact with LLMs through APIs, where such access is unavailable. To address this, existing methods broadly include offline task-level prompt optimization and deployment-time prompt rewriting. *Offline optimization* (Pryzant et al., 2023; Guo et al., 2024; Opsahl-Ong et al., 2024) refines system prompts for specific tasks using existing datasets. While effective in closed environments, these prompts often overfit and fail to generalize to new tasks. *Deployment-time prompt rewriting* (Cheng et al., 2024; Zheng et al., 2024b) dynamically enhances user prompts during inference using auxiliary models trained on paired datasets. Although flexible, this approach requires additional model training and exhibits limited transferability to new domains. *Hybrid methods* (Zhang et al., 2025) combine offline system-prompt optimization with dynamic user-prompt enhancement at deployment. While improving runtime performance to some extent, they still face challenges in cross-task generalization and reusability.

Our test-time setting is different from methods that optimize or adapt prompts during deployment. PTP performs no test-time training, feedback-driven search, or per-query parameter adaptation. Instead, a meta-template learned once offline is used for test-time query-level prompt instantiation, which requires only a single generation step and can be reused across tasks and model families.

Despite recent advancements, prompt optimization still lacks a clear mechanism for transferring optimized prompt knowledge from source tasks to target tasks. In this paper, transferability refers to applying prompt structures learned from one or more source tasks to new tasks or queries without

* Equal contribution.

† Corresponding author.

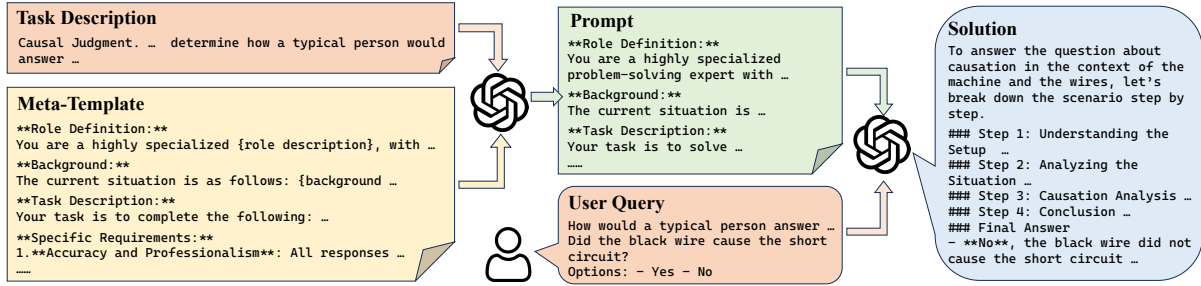


Figure 1: An example of model questioning based on the PTP method under the offline setting. Given a task description and a query as input, PTP generates the solution by instantiating a meta-template.

rerunning full prompt optimization on target-task training data. This limitation manifests in two key aspects:

- 1) **Insufficient Cross-task Performance:** Existing methods are highly dependent on task-specific data and configurations, leading to performance degradation when moved to new tasks.
- 2) **Limited Reusability Efficiency:** Optimized prompts are typically one-off solutions, lacking mechanisms for reuse across offline optimization and test-time query-level instantiation scenarios.

Most existing studies focus primarily on enhancing single-task performance, paying limited attention to prompt reuse across multiple tasks. This motivates a transferable framework that supports both offline task-level optimization and test-time query-level prompt instantiation.

In this paper, we propose Prompting to Prompt (PTP), a meta-template optimization framework designed to enhance prompt transferability. The core idea of PTP is to introduce a *meta-template*—a structured intermediate representation that decomposes prompts into transferable elements—and use it to guide a generalizable optimization mechanism. PTP supports both offline training and test-time query-level prompt instantiation without requiring task-specific retraining. During training, PTP adopts a bi-level optimization strategy: 1) Inner Loop: Uses LLM-generated textual gradients and a prompt element list to perform semantic-level refinement and restructuring of prompts. Unlike conventional string-level tuning, this allows for structural improvements and expressive prompt variants. 2) Outer Loop: Analyzes structural differences between pre- and post-optimization prompts to identify transferable patterns, during which the element list is also leveraged to optimize prompt composition. This step adjusts the meta-template structure and improves cross-task adaptability by accumulating reusable prompt knowledge. In deploy-

ment, PTP can generate task-level or query-specific prompts without additional training. Leveraging the learned meta-templates, PTP can quickly adapt to new tasks and dynamically generate enhanced prompts based on user input.

Our contributions are summarized as follows:

- 1) We introduce a learnable meta-template representation that decomposes prompts into reusable structural elements for transferable prompt optimization.
- 2) We design a bi-level textual-gradient optimization procedure that refines task-specific prompts in the inner loop and updates transferable meta-template structures in the outer loop.
- 3) We evaluate PTP in offline cross-task transfer and test-time query-level prompt instantiation settings across multiple benchmarks and LLMs. Results show that PTP improves over competitive prompt-optimization baselines, with gains up to 10.52 points on Arena-Hard.

2 Related Work

2.1 Prompt Engineering

Compared to resource-intensive fine-tuning, prompt engineering adjusts only inputs without modifying parameters. However, manual design requires expert knowledge and lacks cross-model transferability (Li et al., 2025c), driving interest in automatic methods (Zhang et al., 2023b; Shum et al., 2023; Guo et al., 2024).

Automatic prompt engineering divides into soft and hard approaches. Soft methods (Lester et al., 2021; Qin and Eisner, 2021) optimize continuous embeddings via gradient descent but become impractical at scale due to computational costs and API restrictions. Hard methods (Chung et al., 2024; Pryzant et al., 2023; Zhao et al., 2025) use discrete textual instructions, offering better interpretability, flexibility, and API compatibility. We adopt the

hard approach.

2.2 Hard Prompt Engineering

Recent work can be classified into offline prompt optimization and deployment-time prompt rewriting. Offline methods refine task-specific prompts before deployment, including: APE (Zhou et al., 2023) uses Monte Carlo search to select high-performing variants; ProTeGi (Pryzant et al., 2023) employs beam search with natural language gradients; OPRO (Yang et al., 2024) uses LLMs as optimizers to generate candidate prompts conditioned on a fixed meta-prompt containing historical optimization traces. Deployment-time rewriting methods refine prompts dynamically during inference: BPO (Cheng et al., 2024) rewrites inputs using preference data; PAS (Zheng et al., 2024b) enhances prompts with contextual guidance. Hybrid approaches like P3 (Zhang et al., 2025) combine both strategies, optimizing system prompts offline while supporting dynamic enhancement at deployment. Despite this progress, most methods optimize prompts at the task or query level rather than treating prompt structure as a reusable object to be learned. This leaves a gap between improving a prompt for one setting and extracting prompt knowledge that can be reused in another setting. PTP addresses this gap with learnable meta-templates: unlike OPRO’s fixed meta-prompt, the meta-template is the optimization target itself and captures task-agnostic structural priors through bi-level optimization. This supports both offline adaptation and plug-and-play deployment across diverse tasks.

3 Method

3.1 Preliminary

3.1.1 Objectives of Prompt Engineering:

Consider datasets $D = \{(D_i, C_i)\}$, where $D_i = \{(x_{ij}, y_{ij})\}_{j=1}^m$ refers to the dataset for task i ($i = 1, 2, \dots, n$). Each D_i consists of pairs of questions x_{ij} and their corresponding answers y_{ij} . C_i is the task description associated with D_i . Given a black-box task LLM M_{task} (e.g., ChatGPT) that generates responses with input questions x_{ij} from D_i , the goal of prompt optimization is to identify the optimal prompt P^* that maximizes the expected average task-specific score over D_i , denoted by the

scoring function $\varepsilon_i(\cdot)$. This can be formalized as:

$$P^* = \arg \max_P \mathbb{E}_{(x_{ij}, y_{ij}) \sim D_i} [\varepsilon_i(M_{\text{task}}(x_{ij}; P), y_{ij})] \quad (1)$$

3.1.2 Meta Prompting & Meta-Template:

Traditional prompt design methods, such as Chain-of-Thought (CoT) Prompting (Wei et al., 2022), often struggle with deep reasoning for complex tasks and lack flexibility and generalization for new tasks.

Meta Prompting (Zhang et al., 2023a) demonstrates that prompt structures, independent of task-specific content, can already support a degree of cross-task generalization. By aligning tasks with systematically designed prompt forms, it highlights the importance of structural regularities in facilitating transfer across tasks. Extending this idea, meta-templates build on such inherent generalization by introducing a learning-based mechanism to induce structural patterns from data. Rather than relying on fixed or handcrafted formats, meta-templates capture higher-level abstractions that are optimized through experience, thereby enhancing the robustness and scalability of cross-task transfer. This data-driven extension retains the generalization capacity of Meta Prompting while further improving adaptability in task-level and query-level settings.

3.1.3 Meta-Learning & Bi-level Optimization:

As a method for optimizing the meta-template, we draw inspiration from meta-learning, often described as “learning how to learn.” Designed to improve multi-task generalization by leveraging knowledge across diverse task distributions, meta-learning employs a bi-level optimization process: the inner loop adapts to specific tasks, while the outer loop refines global meta-knowledge to enhance performance across tasks. This approach, exemplified by frameworks like Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), offers a strong foundation for learning transferable meta-templates that adapt efficiently in multi-task scenarios.

3.2 Overview

Our PTP method introduces a framework to incorporate meta-learning based on meta prompting, as shown in Figure 2. Unlike prior methods (Yang et al., 2024; Pryzant et al., 2023) that

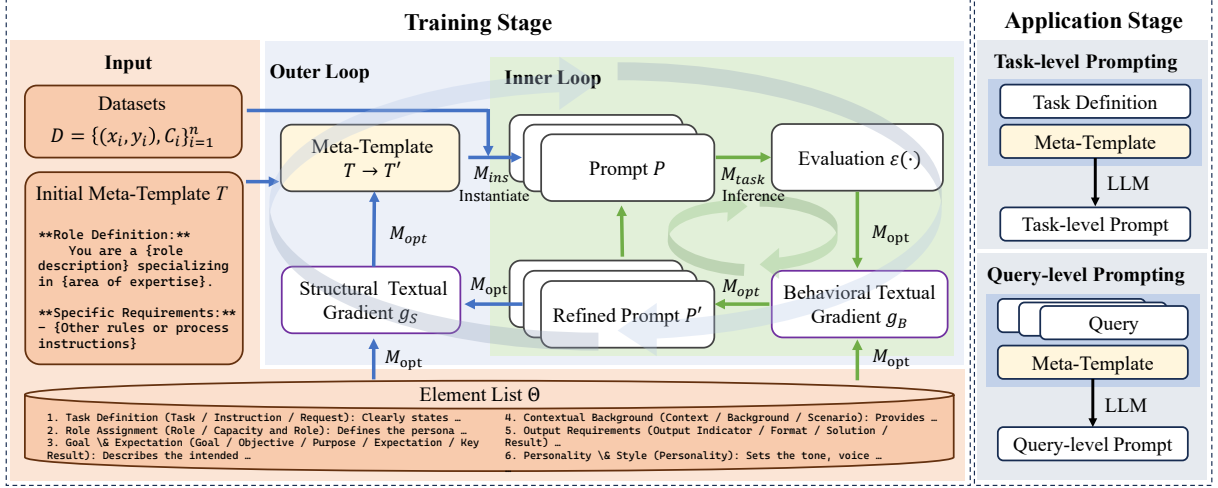


Figure 2: *Overview of the proposed method.* Given a dataset D , an initial meta-template T , and a static element list Θ —curated from prompt engineering best practices—our framework employs a bi-level optimization to refine T into T' . In both the outer loop (updating T) and the inner loop (optimizing its instantiated form P), elements are iteratively sampled from Θ to guide structural refinement. At inference time, T' is used to construct prompts in two scenarios: task-level prompting for offline settings, combining task descriptions with T' ; and test-time query-level prompt instantiation, composing user queries with T' . In both cases, the resulting prompts are fed into an LLM for downstream inference.

focus on individual prompts, PTP learns a transferable meta-template T from task-specific meta-prompts P . This structural flexibility allows PTP to identify common patterns, enhancing its transferability across multi-task settings. PTP employs a bi-level optimization process: the inner loop fine-tunes task-specific prompts, while the outer loop refines T by learning a general paradigm across tasks from inner-loop optimization results.

The initial meta-template contains only role-defining elements, while the remainder—comprising task-relevant structural patterns and transferable prompt components—is progressively learned and incorporated through optimization. To support meta-template construction, we use an element list Θ —a pool of fundamental prompt components (detailed in Appendix F.1)—as a repository for constructing and refining T . Overall, given dataset $D = \{(D_i, C_i)\}$, element list Θ , and initial meta-template T as input, PTP extends Eq. 1 into a new optimization objective:

$$T^* = \arg \max_T \mathbb{E}_{(x_{ij}, y_{ij}) \sim \mathcal{D}_i} [\varepsilon_i(M_{\text{task}}(x_{ij}; P_i(T, \Theta, C_i)), y_{ij})] \quad (2)$$

where $P_i(T, \Theta, C_i)$ denotes the task-specific prompt instantiated from T , Θ and C_i . The optimization is performed by an advanced prompt optimizer M_{opt} (e.g., GPT-4), which generates *textual gradients* (Pryzant et al., 2023; Yüksekönül

et al., 2025)—natural-language feedback analogous to numeric gradients that indicate directions for prompt improvement—to optimize T .

Distinct from ProTeGi (Pryzant et al., 2023) that primarily summarizes errors, PTP employs two levels of textual gradients: (1) the inner-loop behavior textual gradient g_B , which encodes task-specific feedback for P_i and further suggests structural edits over Θ (including additions, deletions, and modifications), and (2) the outer-loop structural textual gradient g_S , which abstracts cross-task patterns by contrasting pre- and post-optimized prompts to guide meta-template revisions based on Θ . The prompts used during the optimization phase are provided in Appendix F.

The following sections detail the inner and outer loops.

3.3 Inner Loop

For each task i , the meta-template T is instantiated into an initial task-specific prompt P_i^0 under the task description C_i using an instantiation LLM M_{ins} :

$$P_i^0 = M_{\text{ins}}(T, C_i; p_{\text{ins}}) \quad (3)$$

Here, p_{ins} is the instantiation prompt that M_{ins} requires to generate the task-specific prompt P_i^0 . During each iteration t , the current prompt P_i^{t-1} , combined with the input question x_i , is used to query the task LLM M_{task} , producing predictions

Algorithm 1 PTP Algorithm

Require: Dataset $D = \{(x, y), C\}$, Initial Meta-Template T , Element List Θ

- 1: **for** $k \leftarrow 1$ to $K_{\text{outer}} + 1$ **do**
- 2: Instantiate T with current tasks: Eq. 3;
- 3: **for** $t \leftarrow 1$ to $K_{\text{inner}} + 1$ **do**
- 4: Generate predictions $\hat{y}$;
- 5: Evaluate $\hat{y}$ and generate incorrect example set: Eq. 4;
- 6: Yield behavioral textual gradient: Eq. 5;
- 7: Refine instantiated prompts: Eq. 6;
- 8: Select refined prompts via beam search; {See Alg. 2}
- 9: **end for**
- 10: Generate structural textual gradient: Eq. 7;
- 11: Refine the meta-template: Eq. 8;
- 12: $T \leftarrow T'$;
- 13: **end for**
- 14: $T^* \leftarrow T$;
- 15: **return** Optimal Meta-Template T^*

$\hat{y}_i$. These predictions are then evaluated against the ground truth y_i with score function $\varepsilon_i(\cdot)$. Errors identified in the evaluation are collected into a set of incorrect examples $\mathcal{B}^{t-1}$:

$$\mathcal{B}^{t-1} \leftarrow \text{Select}(\{(x, y, \hat{y}) \mid \varepsilon(\hat{y}, y) < \tau\}) \quad (4)$$

where τ is a threshold for selecting incorrect examples. To refine the prompt, the optimizer M_{opt} takes the previous task-specific prompt P_i^{t-1} , the error set $\mathcal{B}^{t-1}$, and the element list Θ , guided by the instructive prompt p_{err} , and produces behavioral textual gradients g_B^{t-1} :

$$g_B^{t-1} = M_{\text{opt}}(P_i^{t-1}, \mathcal{B}^{t-1}, \Theta; p_{\text{err}}) \quad (5)$$

These behavioral textual gradients g_B^{t-1} contain insights on prediction errors and element-level editing suggestions for improving the prompt. Guided by $p_{\text{opt}}^{\text{inner}}$, M_{opt} leverages g_B^{t-1} to refine P_i^{t-1} by performing structural modifications—adding, removing, or revising elements—yielding a series of updated prompts P_i^t :

$$P_i^t = M_{\text{opt}}(P_i^{t-1}, g_B^{t-1}; p_{\text{opt}}^{\text{inner}}) \quad (6)$$

By referencing the element list Θ , this refinement guides the prompt optimization toward meaningful structural and element-level modifications, ensuring systematic and directed improvements.

Following ProTeGi (Pryzant et al., 2023), we apply beam search to select the most promising candidates and use them to initialize the next optimization step. Specifically, at each iteration t , multiple candidate prompts P_i^t (from Eq. 6) are evaluated on the validation set, retaining only the

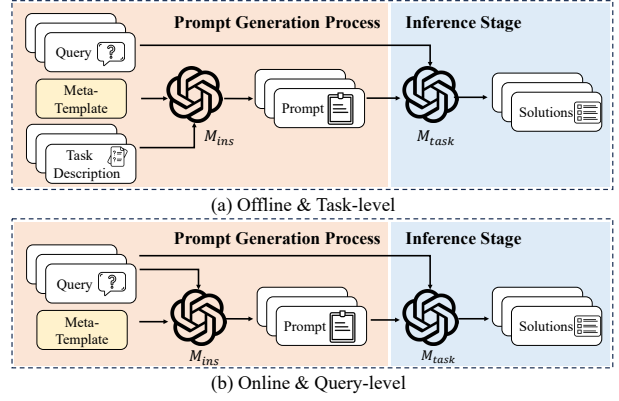


Figure 3: Application pipeline of PTP. First, instantiation LLM M_{ins} generates prompts based on the meta-template and task description/user query. These prompts are then input into Task LLM M_{task} to produce solutions.

top- b candidates (beam width b) based on their validation scores $\varepsilon_i(\cdot)$. Details are provided in Algorithm 2 (Appendix B). This inner-loop adaptation is repeated for a fixed number of iterations K (typically a small constant). After K iterations, the refined prompt P_i^K converges to an approximately optimal task-specific prompt P_i^* : $P_i^K \approx P_i^*$.

3.4 Outer Loop

In contrast to the inner loop, which produces behavioral gradients from incorrect examples, the outer loop derives structural gradients by comparing prompts before and after inner-loop refinement. Specifically, it analyzes the changes from the initial prompt P_i^0 to the refined prompt P_i^K , attributing improvements in task performance to structural modifications. Guided by the element list Θ , this process generates the structural textual gradient:

$$g_S = M_{\text{opt}}(P_i^0, P_i^K, \Theta; p_{\text{dif}}) \quad (7)$$

where p_{dif} is an instructive prompt guiding M_{opt} to generate the structural gradient.

Similarly, the structural textual gradient g_S guides updates to the meta-template T . Structural patterns or elements that consistently improve performance are reinforced, while less effective ones are pruned. The meta-template is updated as:

$$T' = M_{\text{opt}}(T, g_S; p_{\text{opt}}^{\text{outer}}) \quad (8)$$

Through iterative meta-training, T' gradually converges toward an optimal meta-template T^* that encodes generalizable structural priors for prompt construction across tasks: $T' \approx T^*$

The pseudo code of PTP is shown in Algorithm 1.

3.5 Offline Optimization and Test-Time Instantiation

In the offline setting (Figure 3(a)), each new task undergoes a single inner-loop adaptation using T to generate a task-specific prompt P_i , which the LLM uses to infer solutions based on user queries. At test time, the learned meta-template is instantiated in a single step conditioned on the query to produce a prompt for inference. This process performs no training, feedback-driven search, or parameter adaptation at deployment. By encoding transferable prompt structures, T enables effective prompts with one instantiation step, avoiding extra training (Cheng et al., 2024) or query-specific augmentation models (Zheng et al., 2024b).

4 Experiments

4.1 Experimental Setup

Datasets. Our evaluation covers six representative benchmark suites: mathematical reasoning via GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), diverse symbolic and logical tasks from Big-Bench Hard (BBH) (Suzgun et al., 2023), as well as open-ended instruction-following environments including Arena-Hard (Li et al., 2025b), Alpaca-Eval 2.0 (Li et al., 2023), and Alpaca-Eval 2.0 LC (Dubois et al., 2024). For BBH, we select 8 representative tasks spanning diverse domains and reasoning types: Causal Judgment (CJ), Hyperbaton (HB), Logical Deduction Seven Objects (LD), Multistep Arithmetic Two (MA), Navigate (Nav), Reasoning About Colored Objects (CO), Tracking Shuffled Objects Five Objects (Track-5), and Tracking Shuffled Objects Seven Objects (Track-7). These datasets collectively cover arithmetic reasoning, symbolic problem solving, complex task understanding, and general-purpose alignment evaluation. More details about datasets are shown in Appendix C.

Comparison Methods & Baselines. To assess the effectiveness of PTP, we compare it against several LLM-based prompting methods employing diverse optimization strategies. These include offline approaches such as APE (Zhou et al., 2023), ProTeGi (Pryzant et al., 2023), and OPRO (Yang et al., 2024); the deployment-time rewriting method PAS (Zheng et al., 2024b); and the hybrid framework P3 (Zhang et al., 2025). We also consider chain-of-thought prompting (“CoT”) (Kojima et al.,

2022), which uses the instruction “*Let’s think step by step.*” Additionally, original LLM outputs without prompt optimization are reported as baselines.

Implementation Details. In the offline setting, GPT-4o-mini serves as both the instantiation model (M_{ins}) and task model (M_{task}) for all methods, while prompt optimization is performed by GPT-4o (M_{opt}). The initial meta-template (Appendix G.2) includes only role-defining elements, with structural components learned progressively from element list Θ (Appendix F.1) through optimization. To ensure consistent initialization, all methods use a standardized prompt: “*Solve the problem step by step and provide the final answer.*”, which PTP uses to instantiate the initial meta-template. PTP sets both inner-loop and outer-loop iterations to 4, as supported by the ablation in Figure 5. For test-time query-level prompt instantiation, no training, feedback-driven search, or per-query parameter adaptation is performed. The optimized meta-template T^* is instantiated on-the-fly by M_{ins} using the user query, and M_{task} performs inference with the resulting prompt. To evaluate the generalization of T^* across different LLMs, we test Qwen2-7B-Instruct (Bai et al., 2023), GPT-4-turbo-2024-04-09, GPT-4-1106-preview, GPT-4o-mini, GPT-3.5-turbo-1106 (Welsby and Cheung, 2023), and DeepSeek-V3-0324 (DeepSeek-AI, 2024). Each model serves as both M_{ins} and M_{task} during inference. For GSM8K, MATH, and BBH, we report accuracy averaged over three runs; for Arena-Hard, Alpaca-Eval 2.0, and Alpaca-Eval 2.0 LC, we report the corresponding judge-based benchmark scores. Additional implementation details are provided in Appendix D.

4.2 Main Results

We evaluate whether Prompting to Prompt (PTP) can learn reusable meta-templates that improve performance beyond the source training task. Specifically, we test whether meta-templates trained on a limited task corpus (e.g., GSM8K or BBH-CJ) transfer to related reasoning tasks. We further examine whether these templates support test-time query-level prompt instantiation without full prompt re-optimization, while still improving over existing prompting methods. These experiments assess both the transfer behavior and practical deployment value of PTP.

Table 1: Performance comparison of offline prompting methods across varying training sets, demonstrating cross-task transferability. Bold highlights the best method per training setting.

Method	Reasoning		BBH Tasks								
	GSM8K	MATH	CJ	HB	LD	MA	Nav	CO	Track-5	Track-7	Average
Raw	93.23	75.46	62.75	86.80	59.33	96.04	94.27	83.73	68.40	68.13	77.48
CoT	94.10	77.32	63.10	86.67	57.60	98.27	92.80	84.00	71.07	66.53	77.51
Training Set: GSM8K											
ProTeGi	94.20	77.41	63.10	90.60	61.20	96.60	95.20	82.80	68.80	65.60	77.92
APE	93.75	75.03	64.71	86.00	60.40	98.00	95.20	83.20	68.40	69.60	78.06
OPRO	93.56	75.88	64.17	87.20	60.00	96.80	97.20	88.80	73.20	69.60	79.62
PTP	94.90	77.82	65.78	92.60	61.60	97.80	97.30	89.20	77.20	75.60	82.14
Training Set: BBH Causal Judgment											
ProTeGi	93.85	76.60	67.30	89.20	54.00	97.20	91.60	84.20	67.60	70.00	77.69
APE	93.56	73.41	64.71	87.60	63.20	96.20	94.40	81.60	70.00	70.80	78.56
OPRO	93.93	72.37	64.17	91.20	62.40	92.40	95.60	82.40	71.20	69.40	78.60
PTP	94.76	77.43	66.67	92.40	63.20	98.60	95.00	84.60	75.60	76.00	81.33

4.2.1 Offline.

To evaluate the transferability of PTP meta-templates, we conduct offline experiments training on a single dataset, either GSM8K or BBH Causal Judgment (CJ), and testing across three benchmarks: GSM8K, MATH, and eight diverse BBH tasks. As shown in Table 1, when trained on GSM8K, PTP achieves 94.9% on GSM8K, 77.82% on MATH, and an average of 82.14% on BBH tasks, surpassing the strongest baseline by 2.52 points on BBH and 0.41 points on MATH. When trained on BBH-CJ, PTP obtains 94.76%, 77.43%, and 81.33%, again outperforming all baselines in terms of average BBH accuracy, with a +3.64 point gain over ProTeGi and +2.73 points over OPRO. Since CJ is included in the reported BBH average for the BBH-CJ setting, we interpret this result as all-task transfer performance rather than strictly unseen-task transfer. These results provide evidence that PTP can extract reusable structural patterns from limited source data and transfer them across related tasks.

4.2.2 Test-Time Query-Level Prompt Instantiation.

To evaluate the plug-and-play capability of PTP across different large language models (LLMs), we apply a GSM8K-trained meta-template to three open-ended benchmarks—Arena-Hard, Alpaca-Eval 2.0, and Alpaca-Eval 2.0 LC. For each LLM, we keep the task model and the instantiation model identical, ensuring a fair comparison without retraining or deployment-time adaptation. As

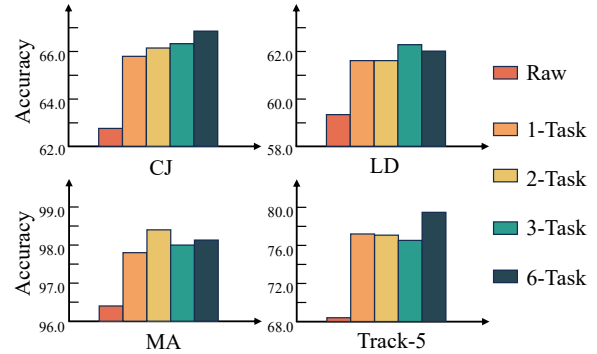


Figure 4: Effect of multi-task training scale on PTP’s transfer accuracy across BBH evaluation tasks.

shown in Table 2, PTP matches or improves over deployment-time baselines including Raw, PAS, and P3, with average improvements of 4.74 to 10.52 points over P3.

4.3 Detailed Analysis

Effect of Model Headroom. The magnitude of PTP’s improvement depends on the performance headroom of the underlying model. On Arena-Hard, DeepSeek-V3 already achieves 92.50 with the Raw prompt and reaches 94.00 with PTP, a 1.50-point improvement over Raw near the upper end of the benchmark. In contrast, GPT-3.5-turbo starts at 18.90 and reaches 47.50, yielding a 28.60-point improvement over Raw. Thus, the smaller absolute gain for stronger models is consistent with a ceiling effect rather than a loss of effectiveness: PTP improves every evaluated model, while the attainable margin naturally shrinks as the unoptimized model approaches saturation.

Table 2: Performance comparison of test-time query-level prompting methods using transferred meta-templates (trained on GSM8K) across multiple models and benchmarks. Bold highlights the best method per benchmark.

Dataset	Model	Raw	PAS	P3	PTP	Δ
Arena-Hard	GPT-4-turbo-2024-04-09	76.60	76.90	78.00	84.60	+6.60
	GPT-4-1106-preview	74.80	78.80	77.21	85.20	+6.40
	GPT-3.5-turbo-1106	18.90	22.10	25.56	47.50	+21.94
	Qwen2-72b-Instruct	48.10	52.20	52.82	69.60	+16.78
	DeepSeek-V3-0324	92.50	93.00	92.70	94.00	+1.00
	GPT-4o-mini	73.80	72.50	73.30	81.80	+8.00
	Average	64.12	65.92	66.60	77.12	+10.52
Alpaca-Eval 2.0	GPT-4-turbo-2024-04-09	46.12	65.31	70.50	70.62	+0.12
	GPT-4-1106-preview	50.00	65.92	69.94	70.19	+0.25
	GPT-3.5-turbo-1106	9.20	15.82	34.53	50.94	+16.41
	Qwen2-72b-Instruct	31.70	45.53	61.37	62.38	+1.01
	DeepSeek-V3-0324	74.84	79.06	80.25	81.25	+1.00
	GPT-4o-mini	41.31	56.75	59.75	69.38	+9.63
	Average	42.20	54.73	62.72	67.46	+4.74
Alpaca-Eval 2.0 LC	GPT-4-turbo-2024-04-09	55.02	56.54	58.31	62.60	+4.29
	GPT-4-1106-preview	50.00	53.63	56.95	59.91	+2.96
	GPT-3.5-turbo-1106	19.30	23.31	35.35	49.65	+14.30
	Qwen2-72b-Instruct	39.24	44.31	55.72	56.58	+0.86
	DeepSeek-V3-0324	78.25	78.29	79.04	80.64	+1.60
	GPT-4o-mini	49.82	51.51	51.74	58.01	+6.27
	Average	48.61	51.27	56.19	61.23	+5.04

Table 3: Effect of instantiation model selection on PTP prompt generation performance across datasets, with a fixed GPT-4o-mini task model. Raw scores are reported once per dataset, while PTP and Δ are shown for each instantiation model.

Dataset	Raw	GPT-3.5-turbo-1106		GPT-4o-mini		DeepSeek-V3	
		PTP	Δ	PTP	Δ	PTP	Δ
Arena-Hard	73.80	79.76	+5.96	81.81	+8.01	81.33	+7.53
Alpaca-Eval 2.0	41.31	68.62	+27.31	69.38	+28.07	64.56	+23.25
Alpaca-Eval 2.0 LC	49.82	55.89	+6.07	58.01	+8.19	57.66	+7.84

We conduct a detailed analysis of the key factors that affect the transferability of PTP. In particular, we examine the effect of multi-task training, the influence of the prompt instantiation model, and the impact of inner- and outer-loop iteration counts.

Effect of Multi-task Training Scale. Figure 4 presents the impact of the number of training tasks on PTP’s transfer performance across BBH evaluation tasks. As the number of training tasks increases from 1 to 6, the overall performance of meta-templates is maintained or improved. Some evaluation tasks overlap with the larger training sets, so we treat these results as evidence for the benefit of broader task exposure rather than a strictly unseen-task transfer test. These results suggest that exposing PTP to a broader range of tasks helps it capture more reusable prompt patterns. Although performance on a few evaluation tasks may fluctuate as more training tasks are introduced, the overall trend indicates that greater

task diversity can improve prompt behavior across related tasks.

Consistency of the Instantiation Model. To evaluate sensitivity to the instantiation model, we fix the task model (GPT-4o-mini) and instantiate prompts using three models: GPT-3.5-turbo-1106, GPT-4o-mini, and DeepSeek-V3. As reported in Table 3, all three instantiation models achieve comparable improvements over the Raw baseline across datasets. For instance, on Arena-Hard, PTP instantiated with GPT-3.5-turbo-1106 achieves 79.76% (+5.96), closely matching GPT-4o-mini (81.81%, +8.01) and DeepSeek-V3 (81.33%, +7.53). Similar patterns are observed on Alpaca-Eval 2.0 and its length-controlled variant, with performance differences limited to a few percentage points. These results indicate that PTP meta-templates can transfer across different instantiation models, although performance still varies by benchmark and model. This supports the practical use of PTP when the instantiation model differs from the optimizer or task model.

Ablation on Loop Iterations. To assess the impact of iteration numbers in PTP’s bi-level optimization, we systematically varied the outer-loop (K_{outer}) and inner-loop (K_{inner}) iterations. Results are shown in Figure 5. Performance generally improves with larger K_{outer} or K_{inner} , but the gains plateau beyond 4. Increasing K_{inner} from 2 to 4 leads to substantial improvements across all K_{outer}

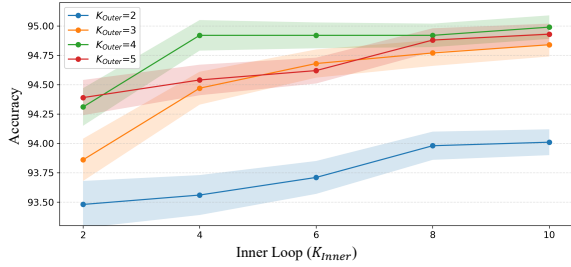


Figure 5: Ablation study on the number of inner and outer loops in PTP optimization.

Table 4: Ablation on the size of the element list ($|\Theta|$) on Arena-Hard. PTP is trained on GSM8K and evaluated with GPT-4o-mini as both M_{task} and M_{ins} .

$ \Theta $	4	8	12 (Default)	16
Accuracy	75.50	80.70	81.80	82.20

values, but further increases yield diminishing returns. Similarly, raising K_{outer} from 2 to 4 steadily improves performance for each K_{inner} , though performance slightly degrades beyond 4. Based on this trend, we set $K_{\text{outer}} = 4$ and $K_{\text{inner}} = 4$ as the default to balance performance and efficiency, enabling sufficient prompt refinement and cross-task adaptation without excessive computation.

Sensitivity to Element List Size. On Arena-Hard, PTP reaches 75.50% with four elements, exceeding Raw (73.80%) and P3 (73.30%). Accuracy rises to 81.80% at the default 12 elements and changes only to 82.20% at 16, showing diminishing returns from further expansion.

Effect of Optimizer Model Choice. Replacing GPT-4o with DeepSeek-V3 yields 79.07 average accuracy on CJ, MA, and Track-5, close to GPT-4o’s 80.26. GPT-4o-mini reaches 77.37, showing that PTP remains effective with both alternative optimizers.

Computational Cost and Trade-Offs. We use optimizer API calls as a cost proxy because dollar cost varies with provider pricing and token length. PTP needs 844 calls to learn a reusable meta-template and 3 calls per new target task, whereas ProTeGi needs 772 calls whenever a task is optimized. Across GSM8K and eight BBH tasks, this gives 868 calls for PTP versus 6,948 for repeated ProTeGi.

Table 5: Effect of the optimizer model (M_{opt}) on PTP performance. Meta-templates are trained on GSM8K and evaluated on three representative BBH tasks, with GPT-4o-mini fixed as both M_{task} and M_{ins} .

Optimizer Model	CJ	MA	Track-5	Average
GPT-4o-mini	63.80	96.50	71.80	77.37
DeepSeek-V3	65.10	97.40	74.70	79.07
GPT-4o (Default)	65.78	97.80	77.20	80.26

Table 6: Comparison of optimization API calls. PTP learns a reusable meta-template once, while ProTeGi re-optimizes for each task.

Method	Training Calls	Per-New-Task Calls	9-Task Total
PTP	844	3	868
ProTeGi	772	772	6,948

5 Conclusion

In this paper, we propose Prompting to Prompt (PTP), a framework for learning reusable meta-templates for prompt optimization. PTP shifts optimization from isolated prompt strings to structured prompt representations, using an inner loop to refine task-specific prompts and an outer loop to abstract structural changes into the meta-template. Experiments in offline transfer and test-time deployment settings show that PTP improves over competitive prompt-optimization baselines across reasoning and instruction-following benchmarks. Further analysis suggests that broader task exposure, suitable instantiation models, and balanced inner- and outer-loop iterations help the learned meta-template capture reusable prompt patterns. These results support meta-template learning as a practical direction for transferable prompt optimization, while also indicating the need for broader evaluation on more diverse task families and optimizer settings.

Limitations

Dependence on Advanced Optimizers. PTP relies on advanced LLMs (e.g., GPT-4o) as prompt optimizers to generate textual gradients and refine meta-templates, making the template quality inherently constrained by the optimizer’s reasoning capability. Future work could explore techniques such as incorporating human feedback or ensemble-based strategies to improve optimization robustness under resource constraints.

Expansion of Element List Design. The element list Θ is curated from established prompt engineering practices, which provides a structured foundation but may not cover all specialized components in novel domains. Future work could investigate automated methods to mine and expand element lists from task data, or allow the optimizer to dynamically propose new elements during training.

Evaluation on Additional Task Types. While our experiments validate PTP on reasoning and instruction-following benchmarks, evaluating its transferability across additional modalities and domains (e.g., multi-modal understanding, code generation, and domain-specific applications) remains an important next step.

References

- Eshaan Agarwal, Vivek Dani, Tanuja Ganu, and Akshay Nambi. 2024. [Promptwizard: Task-aware agent-driven prompt optimization framework](#). *CoRR*, abs/2405.18369.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. [Qwen technical report](#). *CoRR*, abs/2309.16609.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. 2023. [Sparks of artificial general intelligence: Early experiments with GPT-4](#). *CoRR*, abs/2303.12712.
- Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2024. [Black-box prompt optimization: Aligning large language models without model training](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 3201–3219. Association for Computational Linguistics.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. [Scaling instruction-finetuned language models](#). *J. Mach. Learn. Res.*, 25:70:1–70:53.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- Wendi Cui, Jiaxin Zhang, Zhuohang Li, Hao Sun, Damien Lopez, Kamalika Das, Bradley A. Malin, and Kumar Sricharan. 2025. [Heuristic-based search algorithm in automatic instruction-focused prompt optimization: A survey](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pages 22093–22111. Association for Computational Linguistics.
- DeepSeek-AI. 2024. [Deepseek-v3 technical report](#). *CoRR*, abs/2412.19437.
- Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. [Length-controlled alpaca-eval: A simple way to debias automatic evaluators](#). *CoRR*, abs/2404.04475.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. [Model-agnostic meta-learning for fast adaptation of deep networks](#). In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.
- Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. [Connecting large language models with evolutionary algorithms yields powerful prompt optimizers](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Han He, Qianchu Liu, Lei Xu, Chaitanya Shivade, Yi Zhang, Sundararajan Srinivasan, and Katrin Kirchhoff. 2025. [Crispo: Multi-aspect critique-suggestion-guided automatic prompt optimization for text generation](#). In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 24014–24022. AAAI Press.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings*

- of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, *NeurIPS Datasets and Benchmarks 2021*, December 2021, virtual.
- 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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.
- Hao Li, Ruoyuan Gong, and Hao Jiang. 2025a. [Political actor agent: Simulating legislative system for roll call votes prediction with large language models](#). In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 388–396. AAAI Press.
- Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2025b. [From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline](#). In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.
- Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025c. [A survey of automatic prompt engineering: An optimization perspective](#). *CoRR*, abs/2502.11560.
- Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. AlpacaEval: An automatic evaluator of instruction-following models.
- 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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8086–8098. 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 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. [Optimizing instructions and demonstrations for multi-stage language model programs](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 9340–9366. Association for Computational Linguistics.
- Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. [Automatic prompt optimization with "gradient descent" and beam search](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7957–7968. Association for Computational Linguistics.
- Guanghui Qin and Jason Eisner. 2021. [Learning how to ask: Querying lms with mixtures of soft prompts](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5203–5212. Association for Computational Linguistics.
- Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, Haozhu Wang, Han Ding, Yuzhe Lu, Zhichao Xu, Yun Zhou, Balasubramaniam Srinivasan, Qiaojing Yan, Yueyan Chen, Haibo Ding, and 2 others. 2025. [A systematic survey of automatic prompt optimization techniques](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 33078–33110. 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. [Autoprompt: Eliciting knowledge from language models with automatically generated prompts](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.
- Kashun Shum, Shizhe Diao, and Tong Zhang. 2023. [Automatic prompt augmentation and selection with](#)

- chain-of-thought from labeled data. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, volume EMNLP 2023 of *Findings of ACL*, pages 12113–12139. Association for Computational Linguistics.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. 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, Toronto, Canada, July 9-14, 2023*, volume ACL 2023 of *Findings of ACL*, pages 13003–13051. Association for Computational Linguistics.
- Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. 2025. [Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers](#). In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 25264–25272. AAAI Press.
- Prashant Trivedi, Souradip Chakraborty, Avinash Reddy, Vaneet Aggarwal, Amrit Singh Bedi, and George K. Atia. 2025. [Align-pro: A principled approach to prompt optimization for LLM alignment](#). In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 27653–27661. AAAI Press.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Philip Welsby and Bernard M. Y. Cheung. 2023. [Chat-GPT](#). *Postgraduate Medical Journal*, 99(1176):1047–1048. Editorial.
- Cilin Yan, Jingyun Wang, Lin Zhang, Ruihui Zhao, Xiaopu Wu, Kai Xiong, Qingsong Liu, Guoliang Kang, and Yangyang Kang. 2025. [Efficient and accurate prompt optimization: the benefit of memory in exemplar-guided reflection](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 753–779. Association for Computational Linguistics.
- 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Mert Yüsekşgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. [Optimizing generative AI by backpropagating language model feedback](#). *Nat.*, 639(8055):609–616.
- Xinyu Zhang, Yuanquan Hu, Fangchao Liu, and Zhicheng Dou. 2025. [P3: prompts promote prompting](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pages 11948–11965. Association for Computational Linguistics.
- Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2023a. [Meta prompting for ai systems](#). *arXiv preprint arXiv:2311.11482*.
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. [Automatic chain of thought prompting in large language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Minjun Zhao, Xinyu Zhang, Shuai Zhang, Deyang Li, and Ruifeng Shi. 2025. [Adopt: Adaptive dependency-guided joint prompt optimization for multi-step llm pipelines](#). *arXiv preprint arXiv:2512.24933*.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. [A survey of large language models](#). *CoRR*, abs/2303.18223.
- Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024a. [Large language models are not robust multiple choice selectors](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Miao Zheng, Hao Liang, Fan Yang, Haoze Sun, Tianpeng Li, Lingchu Xiong, Yan Zhang, Youzhen Wu, Kun Li, Yanjun Shen, Mingan Lin, Tao Zhang, Guosheng Dong, Yujing Qiao, Kun Fang, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. 2024b. [PAS: data-efficient plug-and-play prompt augmentation system](#). *CoRR*, abs/2407.06027.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. [Large language models are human-level prompt engineers](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

A Use of Large Language Models

Large language models (LLMs) were used for language polishing and literature-search assistance during manuscript preparation. The authors reviewed the resulting text, claims, citations, and experimental descriptions.

B Beam Search over Prompts

The beam search procedure used in the inner loop (Section 3.3) is formally described in Algorithm 2. This search strategy guides the exploration of prompt candidates generated via behavioral gradient refinement (Eq. 6).

Algorithm 2 Beam Search for Prompt Selection

Require: Candidates C (all P_i^t in Eq. 6), Validation set $\mathcal{D}_i^{\text{val}}$, Beam width b , Metric function $\varepsilon_i(\cdot)$

- 1: **for** each candidate $c \in C$ **do**
- 2: Compute validation score:
 $s(c) \leftarrow \mathbb{E}_{(x,y) \sim \mathcal{D}_i^{\text{val}}} [\varepsilon_i(M_{\text{task}}(x; c), y)]$
- 3: **end for**
- 4: Rank candidates by score: $C_{\text{ranked}} \leftarrow \text{Sort}(C, \text{by} = s(\cdot), \text{descending})$
- 5: Select top- b candidates: $B \leftarrow C_{\text{ranked}}[1 : b]$
- 6: **return** Beam set B containing top- b prompts

C Dataset and Task Details

We conduct experiments across six diverse benchmarks spanning mathematical reasoning, multi-domain reasoning, and instruction-following capabilities. Table 7 provides a summary of dataset statistics and splits.

C.1 Mathematical Reasoning Benchmarks

GSM8K. GSM8K (Cobbe et al., 2021) contains approximately 8,500 grade-school level math word problems designed to evaluate arithmetic and multi-step reasoning capabilities. Each problem requires detailed step-by-step solutions. We use 200 instances from the training set (split into 100 train / 100 validation) and all 1,319 instances from the test set.

MATH. MATH (Hendrycks et al., 2021) is a challenging collection of high-school competition problems covering algebra, geometry, calculus, and other advanced topics. We use a 500-problem subset with 200 training instances (100 train / 100 validation) and 300 test instances to evaluate advanced mathematical reasoning.

C.2 Multi-Domain Reasoning Benchmark

Big-Bench Hard (BBH). BBH (Suzgun et al., 2023) comprises 23 challenging tasks selected from BIG-Bench based on their difficulty for large language models. We select 8 representative tasks spanning logical reasoning, spatial reasoning, language understanding, and mathematical problem-solving:

- **Causal Judgment (CJ):** Evaluate causal reasoning through moral and counterfactual scenarios.
- **Hyperbaton (HB):** Identify correct adjective ordering in English sentences.
- **Logical Deduction (LD):** Deduce object ordering from spatial relationship clues.
- **Multi-Step Arithmetic (MA):** Solve multi-step problems with basic arithmetic operations.
- **Navigate (Nav):** Determine if an agent returns to its starting position after movement instructions.
- **Reasoning About Colored Objects (CO):** Answer simple color-based reasoning questions.
- **Tracking Shuffled Objects – 5/7 (Track-5/7):** Track object positions after pairwise swaps.

For each task, we follow a 2:3 split for training/validation and testing. Except for Causal Judgment (74 train, 113 test), all tasks use 100 training instances and 150 test instances.

C.3 Instruction-Following Benchmarks

Arena-Hard. Arena-Hard (Li et al., 2025b) comprises 500 challenging tasks for evaluating LLM robustness in complex and adversarial scenarios, requiring advanced reasoning and fine-grained language understanding.

Alpaca-Eval 2.0 & Alpaca-Eval 2.0 LC. Alpaca-Eval 2.0 (Li et al., 2023) is an enhanced evaluation framework for instruction-following language models, assessing instruction adherence and alignment quality through diverse open-ended queries. Its length-controlled variant, Alpaca-Eval 2.0 LC (Dubois et al., 2024), introduces regression-based debiasing to mitigate response length influence, improving robustness and interpretability. Leveraging GPT-4 as an automatic annotator, this framework demonstrates high efficiency and consistency while maintaining strong correlation with human judgments.

Table 7: Overview of datasets used in our experiments.

Dataset	Domain	Train	Val	Test
GSM8K	Math Reasoning	100	100	1,319
MATH	Advanced Math	100	100	300
BBH (8 tasks)	Multi-domain Reasoning	774	-	1,163
- Causal Judgment	Logic	37	37	113
- Hyperbaton	Language	50	50	150
- Logical Deduction	Logic	50	50	150
- Multi-Step Arithmetic	Math	50	50	150
- Navigate	Spatial Reasoning	50	50	150
- Colored Objects	Visual Reasoning	50	50	150
- Track-5	Object Tracking	50	50	150
- Track-7	Object Tracking	50	50	150
Arena-Hard	Open-ended QA	-	-	500
Alpaca-Eval 2.0	Instruction Following	-	-	805
Alpaca-Eval 2.0 LC	Instruction Following	-	-	805

C.4 Data Splitting Strategy

For all tasks with training data, we employ a three-way split:

- **Training Set:** Used in the inner loop for generating candidate prompts via textual gradient updates.
- **Validation Set:** Used in the inner loop for evaluating candidates and beam search selection (Algorithm 2).
- **Test Set:** Held-out set for final evaluation, completely disjoint from training and validation.

For Arena-Hard and Alpaca-Eval 2.0, which are evaluation-only benchmarks, we report zero-shot performance using meta-templates learned from other tasks.

D Supplementary Experimental Details

D.1 Experimental Configuration

Model Temperature Settings. In all experiments, the temperature of both the task model (M_{task}) and the instantiation model (M_{ins}) is set to 0 to ensure deterministic behavior. The prompt optimizer (M_{opt}) uses a temperature of 0.7 to encourage diversity in candidate generation, producing 4 candidate prompts at each optimization step.

Training Configuration. PTP is trained with $K_{\text{inner}} = 4$ inner-loop iterations and $K_{\text{outer}} = 4$ outer-loop iterations. For fair comparison, baseline methods (APE, ProTeGi, and OPRO) are trained for 16 iterations, ensuring comparable computational cost across all methods.

D.2 Effect of Multi-task Training Scale

Figure 4 examines how the number of training tasks affects PTP’s transfer performance on BBH evaluation tasks. We train PTP on progressively larger task sets while maintaining identical training configurations:

- **1-task:** GSM8K only.
- **2-task:** GSM8K + Hyperbaton.
- **3-task:** GSM8K + Hyperbaton + Reasoning About Colored Objects.
- **6-task:** GSM8K + Causal Judgment + Logical Deduction Seven Objects + Multistep Arithmetic Two + Navigate + Tracking Shuffled Objects Five Objects.

This controlled design isolates the effect of task diversity, revealing how broader task exposure during training enhances the meta-template’s generalization to unseen tasks.

E Supplementary Experiments

E.1 Impact of Training Data Selection on Transfer Performance

We compare meta-templates trained on different datasets to assess transferability, and the results are shown in Table 8. The **Initial Prompt** baseline achieves 79.38% average accuracy on BBH tasks, while the **Initial Meta-template** (trained only on GSM8K without iterative refinement) reaches 79.65%, showing minimal improvement.

Table 8: Impact of training set choice on PTP transfer performance compared to baselines.

Method	Reasoning		BBH Tasks								
	GSM8K	MATH	CJ	HB	LD	MA	Nav	CO	Track-5	Track-7	Average
Initial Prompt	93.80	76.80	65.24	90.80	61.20	97.20	96.80	85.60	64.00	62.40	79.38
Initial Meta-template	94.00	76.80	65.51	91.80	60.00	96.60	95.40	83.40	69.20	63.80	79.65
Training Set: GSM8K											
PTP	94.90	77.80	65.78	92.60	61.60	97.80	97.30	89.20	77.20	75.60	82.14
Training Set: BBH Causal Judgment											
PTP	94.76	77.40	66.67	92.40	63.20	98.60	95.00	84.20	75.60	76.00	81.33

Applying PTP optimization on GSM8K substantially improves performance to 82.14% average BBH accuracy (+2.49% over Initial Meta-template), with consistent gains on GSM8K (94.90%) and MATH (77.80%). Training on BBH Causal Judgment yields 81.33% average BBH accuracy (+1.68%), with peak performance on its source task (CJ: 66.67%) and strong results on structurally similar tasks (LD: 63.20%, MA: 98.60%).

The results reveal complementary strengths: GSM8K-trained templates excel on quantitative reasoning (MA: 97.80%), while CJ-trained templates perform better on causal and logical tasks (CJ: 66.67%, LD: 63.20%). Both consistently outperform baselines, validating PTP’s ability to extract transferable structures from limited training data.

E.2 Computational Cost Analysis

We analyze the computational efficiency of PTP compared to task-specific optimization methods. Table 9 compares the optimization overhead during training and transfer phases.

Training Overhead. Both PTP and ProTeGi use GPT-4o for 16 optimization iterations, resulting in comparable training costs. PTP incurs ~ 844 optimizer API calls while ProTeGi requires ~ 772 calls. The slight difference stems from PTP’s meta-template generation process, but overall training time is comparable between methods.

Transfer Efficiency. The critical difference emerges during transfer to new tasks. PTP’s meta-template enables near-instant adaptation through a single instantiation step: given a task description and optional examples, the instantiation model M_{ins} generates task-specific prompts in ~ 3 API calls, typically completing within seconds. In contrast, ProTeGi must perform full re-optimization for each new task, requiring ~ 772 API calls—comparable to the full training process.

Multi-Task Deployment. This efficiency gap compounds across multiple tasks. For the nine tasks in our evaluation (GSM8K + 8 BBH sub-tasks), PTP requires one training phase plus eight instantiation steps. Task-specific optimization must repeat the full optimization process for each of the nine tasks. Under the simplifying assumption that optimizer calls dominate runtime and cost, this reduces optimizer-call overhead by approximately 80-85% in multi-task scenarios.

E.3 Impact of Task-Aware Prompt Instantiation on Cross-Task Transferability

We investigate whether incorporating task-specific guidance during prompt instantiation can enhance cross-task transferability without retraining the meta-template. In standard instantiation, prompts are generated solely from a meta-template trained on GSM8K. To improve alignment with target tasks, we augment the instantiation process with two types of task-specific information: natural language task descriptions and concrete task examples.

Task Description Augmentation. For each BBH task, we prepend a concise natural language description that characterizes the task’s reasoning requirements. For instance, the Causal Judgment task uses the following description:

“Given a short story involving moral, intentional, or counterfactual analysis, determine how a typical person would answer a causal question about the story.”

This description provides high-level task context without exposing specific format or solution patterns, allowing the meta-template to adapt its reasoning approach while maintaining its learned structural principles.

Table 9: Comparison of optimizer API calls. PTP requires full optimization only once, while ProTeGi must re-optimize for each task.

Method	Opt Model	Opt API Calls(Training)	Opt API Calls(Transfer)
PTP	GPT-4o	~844	~3 (validation-only)
ProTeGi	GPT-4o	~772	~772 (full re-optimization)

Table 10: Effect of task description and example augmentation during prompt instantiation on BBH transfer performance.

Method	Task Description	Example	CJ	LD	MA	Track-5
Raw			62.75	59.33	96.40	68.40
ProTeGi			67.30	64.33	97.20	74.08
PTP			65.78	61.60	97.80	77.20
PTP	✓		67.18	68.40	97.33	81.30
PTP	✓	✓	66.44	69.73	97.76	82.73

Example Augmentation. Beyond descriptions, we include a single task-specific example to demonstrate the expected input-output format and reasoning style. For Causal Judgment, the example is:

Input: How would a typical person answer each of the following questions about causation?
 An intern is taking care of a patient in a hospital. The intern notices that the patient is having some kidney problems. [...] Despite this policy, the doctor decides to sign off. Since both signatures were received, the patient is administered the drug. As it happens, the patient immediately recovers, and the drug has no adverse effects.
Question: Did the pharmacist’s decision cause the patient’s recovery?
Options: Yes / No
Answer: No

Results and Analysis. As shown in Table 10, task-aware instantiation consistently improves performance. Adding descriptions (PTP + Description) yields +1.4% on CJ, +6.8% on LD, and +4.1% on Track-5 compared to vanilla PTP. Combining descriptions with examples (PTP + Description + Example) achieves the highest accuracy on LD (69.73%, +8.13%) and Track-5 (82.73%, +5.53%).

With this no-retraining augmentation, PTP matches or surpasses ProTeGi on three of four tasks, with clear gains on LD (+5.4%) and Track-5 (+8.65%) despite using a meta-template trained only on GSM8K. This demonstrates that PTP’s meta-learned structures can be adapted through simple instantiation augmentation, achieving performance that can rival task-specific optimization without additional retraining.

F Element List and Prompts Used During Optimization

This section presents the element list utilized in the PTP framework, along with the system prompts required for each LLM, as illustrated in Box F.1 to F.6. Note that these prompts remain fixed during the optimization process.

Box F.1: Element List ☹

- 1. Task Definition (Task / Instruction / Request)**
Clearly states the specific task or request the model is expected to perform.
- 2. Role Assignment (Role / Capacity and Role)**
Defines the persona or expertise the model should assume to guide tone, knowledge, and behavior.
- 3. Goal & Expectation (Goal / Objective / Purpose / Expectation / Key Result)**
Describes the intended outcome or objective, along with any measurable expectations for the output.
- 4. Contextual Background (Context / Background / Scenario)**
Provides necessary background information or situational context to help the model understand the task accurately.
- 5. Output Requirements (Output Indicator / Format / Solution / Result)**
Specifies the desired form, structure, tone, or type of output expected from the model.
- 6. Personality & Style (Personality)**
Sets the tone, voice, or emotional character of the response—e.g., friendly, concise, formal.
- 7. Examples & Demonstrations (Examples)**
Offers sample inputs and/or outputs to illustrate desired response patterns or behaviors.
- 8. Action Steps (Action / Steps)**
Directs the model through sequential actions or procedures necessary to complete the task.
- 9. Clarity & Focus (Clarity / Avoid Ambiguity)**
Ensures the task is framed clearly and unambiguously, removing any extraneous or misleading information.
- 10. Relevant Information (Relevant Info)**
Supplies supporting details, such as keywords, facts, audience characteristics, tone, or formatting preferences.
- 11. Insight & Depth (Insight)**
Adds deeper context, reasoning, or domain-specific considerations to enrich the model’s understanding and response.
- 12. Structure & Format (Structure / F)**
Specifies the organizational format of the response—e.g., bullet points, tables, sections, Markdown.

Box F.2: Instantiation Prompt p_{ins} Required by the Instantiation LLM M_{ins} (Eq. 3)

You are an expert prompt engineer. Your task is to generate a concrete, executable prompt instance (a grounded prompt) based on the provided "meta-template" and "task description."

Objectives

1. The grounded prompt must be highly executable, directly usable by large language models to significantly improve output quality;
2. Retain the structure and format of the meta-template;
3. Clearly fill in or revise vague or abstract parts of the meta-template to closely reflect the real intent expressed in the user instruction;
4. Avoid vague or generic expressions; the language must be clear, specific, and cover core constraints or expectations;
5. Wrap the generated grounded prompt within <START> and <END> tags for easy extraction.

Meta-template

{meta_prompt}

Task Description

{instruction}

Please generate the grounded prompt accordingly:

<START>

(your grounded prompt here)

<END>

- Key points of difference analysis
- Optimization suggestions (in the form: "**You should... because...**")
- Scope of applicability and potential risks
- Wrap your suggestions within <START> and <END> tags for easy extraction

Your goal is to produce thoughtful, generalizable optimization advice for improving meta-prompts.

Please generate your optimization advice as follows:

<START>

(Insert your optimization advice here)

<END>

Box F.5: Optimization Prompt p_{opt}^{inner} for Guiding the Optimizer M_{opt} to Generate Better Prompts in the Inner Loop (Eq. 6)

I'm trying to refine a prompt for large language model.

My current prompt is:

{prompt}

But it gets the following examples wrong:

{error_str}

Based on these examples the problem with this prompt is that {feedback_str} Based on the above information, I wrote a different improved prompt. The improved prompt should be wrapped with <START> and <END>.

The new prompt are:

Box F.3: Instructive Prompt p_{err} for Guiding the Optimizer M_{opt} to Generate Behavioral Text Gradients in the Inner Loop (Eq. 5)

I'm trying to refine a prompt for a large language model.

My current prompt is:

{prompt}

But this prompt gets the following examples wrong:

{error_string}

List of meta-prompt structural elements (add with caution to avoid overly long prompts):

{meta_prompt_elements}

Focus on the difference between ground truth and prediction, and give {num_feedbacks} reasons why the prompt could have gotten these examples wrong.

Based on the element list above, analyze which elements should be added, removed, or modified to improve the effectiveness of the prompt.

Wrap the combined reason-and-edit suggestion within <START> and <END>.

Box F.6: Optimization Prompt p_{opt}^{outer} for Guiding the Optimizer M_{opt} to Generate Better Meta-Template in the Outer Loop (Eq. 8)

You are a meta-prompt design expert, skilled at integrating evaluation data and structural components to generate more effective meta-prompts. Your task is to optimize the current meta-prompt based on the provided information.

1. Current meta-prompt

{current_meta_prompt}

2. Optimization suggestions

{optimization_advice}

Please conduct a multi-angle analysis and focus on the following questions:

- Which optimization suggestions can be transformed into concrete, actionable template content?
- Which current template elements should be strengthened, weakened, or removed?
- Should new structural elements be introduced to improve performance? Please be cautious: only propose additions that are truly necessary, and avoid making the prompt unnecessarily long.

Your output should include:

- A brief explanation of the overall meta-prompt optimization logic, enclosed within <optimization_logic> tags.

- The fully optimized meta-prompt, enclosed within <optimized_meta_prompt> tags. Do not include any optional elements, every instruction must be essential, explicit, and necessary.

<optimization_logic>

(Insert a brief explanation of the overall meta-prompt optimization logic here)

</optimization_logic>

<optimized_meta_prompt>

(Insert the optimized meta-prompt content here)

</optimized_meta_prompt>

Box F.4: Instructive Prompt p_{dif} for Guiding the Optimizer M_{opt} to Generate Structural Text Gradients in the Outer Loop (Eq. 7)

You are a prompt engineering expert, specializing in reverse-optimizing prompt generation rules (*meta-prompts*) through fine-grained analysis of concrete prompt templates.

Please conduct an in-depth analysis of the following content:

1. Meta Prompt (meta_prompt)

{meta_prompt}

2. Initial Concrete Prompt & Score

Initial Prompt: {prompt_concrete}

Initial Score: {prompt_concrete_score}

3. Optimized Concrete Prompt & Score

Optimized Prompt: {opt_concrete_prompt}

Optimized Score: {opt_concrete_prompt_score}

4. List of meta-prompt structural elements

{meta_prompt_elements}

Your tasks:

- Identify the key differences between the initial and optimized prompts.
- Based on these differences and the change in scores, reflect deeply and answer:
 1. Which parts of the meta-prompt failed to effectively guide the generation of high-quality concrete prompts? Why?
 2. What successful expression strategies or structural improvements in the optimized prompt could be integrated back into the meta-prompt to enhance its performance?

Your optimization advice should be concise yet insightful, and must include:

G Initial Prompt and Meta-Template

This section shows the initial meta-template to be optimized.

Box G.1: Initial Prompt(Baseline)

Solve the problem step by step and provide the final answer.

Box G.2: Initial Meta-Template T

Role Definition:

You are a {role description} specializing in {area of expertise}.

Specific Requirements:

- {Other rules or process instructions}

- Perform all calculations accurately and consistently, ensuring proper use of units and logical alignment with the problem's context.
- Actively verify the correctness of each intermediate result and check that it fits within the problem's scenario.
- 2. After completing all steps:
 - Validate that the final result is both mathematically correct and reasonable in the real-world context of the problem.
 - Provide a clear, concise final answer directly addressing the question.
 - Optionally, summarize key steps and reasoning for additional clarity.
- Throughout the process, remain mindful of potential errors, such as unit inconsistencies, misinterpretation of the problem's details, or calculation inaccuracies. Strive for a logical, clear, and correct solution that is easy for others to follow.

H Display and Comparison of Optimized Prompts under Different Experimental Setups

H.1 Optimized Prompts of Different Offline Methods

This section presents examples of prompts optimized by our PTP approach and various offline methods on GSM8K dataset, corresponding to Table 1 in the main text.

Box H.1.1: Offline: Refined Prompt under ProTeGi on GSM8K Dataset

Your task is to solve the given problem step by step, paying **special attention to contextual definitions, quantitative reasoning, and alternative interpretations** of key terms. Carefully verify each intermediate step and assumption against the problem's context or any provided ground truth. Follow these instructions:

1. **Clarify the Problem:** Begin by restating the problem in your own words. Identify all relevant data, constraints, and assumptions. Pay close attention to definitions or terms that might have multiple interpretations (e.g., "halfway," "remaining," "total").
 2. **Establish a Plan:** Outline a clear plan for solving the problem. Break the solution into small, logical steps. Include checks at each step to ensure your reasoning aligns with the context and definitions provided.
 3. **Step-by-Step Solution:**
 - a. Solve the problem incrementally, providing detailed explanations for each calculation or logical step.
 - b. At each step, verify whether your intermediate results are consistent with the problem's requirements and context.
 - c. Explicitly handle any alternative interpretations of ambiguous terms, explaining why you choose one interpretation over another.
 4. **Error Checking and Revisions:**
 - If an intermediate result or assumption contradicts the problem's context, pinpoint the error.
 - Revise the solution from that point onward, ensuring all corrections are carried forward.
 5. **Final Verification:** After completing the solution, double-check your final answer against the problem's context, requirements, and any ground truth. Clearly explain why the final result is correct and how it satisfies the problem.
 6. **Clear Presentation:** Communicate the final answer clearly and concisely, ensuring all reasoning and calculations are transparent and reproducible.
- Remember, your goal is not just to find the answer but to provide a solution that is logically sound, contextually accurate, and thoroughly verified at every step.

Box H.1.2: Offline: Refined Prompt under OPRO on GSM8K Dataset

Approach the problem with a structured and adaptable method to ensure clarity and accuracy. Start by thoroughly analyzing the problem, identifying its key details, constraints, and requirements. Then:

1. Break the solution into clear and logical steps. For each step:
 - Clearly explain what you are doing, why it is necessary, and how it contributes to solving the problem.

Box H.1.3: Offline: Refined Prompt under APE on GSM8K Dataset

- Take note of any mistakes made and think about how they can be avoided in the future.
- Consider what strategies were effective and which ones were not.
- Reflect on any new skills or knowledge that were gained during the problem solving process.
- Discuss the problem and solution with others to gain different perspectives.
- Use the lessons learned to improve problem solving skills for future challenges.

Box H.1.4: Offline: Refined Prompt under PTP(Ours) on GSM8K Dataset

Role Definition:

You are a highly specialized {role description}, with expertise in {specific area of expertise}. Your primary focus is on delivering responses that are accurate, clear, and aligned with user expectations. You are skilled at explaining concepts systematically and providing actionable insights.

Background:

The current situation is as follows: {background information, including the task context, user goals, challenges, and success criteria}. The user expects the output to address these challenges comprehensively and meet their specific objectives.

Task Description:

Your task is to complete the following: {detailed overview of the task, including objectives and measurable outcomes}. Ensure that your response is precise, thorough, and logically structured.

Specific Requirements:

1. **Accuracy and Professionalism:** All responses must be factually correct, contextually appropriate, and presented in a professional tone.
2. **Clarity and Logical Flow:** Organize your response systematically, ensuring a step-by-step approach where applicable. Avoid ambiguity by clearly stating assumptions and reasoning.
3. **Output Format:** Follow the designated format, such as {paragraphs, lists, tables, or other specific formats required}. Clearly label sections if applicable.
4. **Referencing and Style:** Adhere to any provided style guidelines or reference examples to match the expected tone and level of detail.
5. **Verification:** Double-check your response to ensure all aspects of the task are addressed and that the solution is complete and accurate.

H.2 Prompts under Test-Time Query-Level Instantiation

This section presents examples of prompts generated by PTP under test-time query-level instantiation on the Causal Judgment task, corresponding to the deployment evaluation in the main text. The instantiation is guided by the GSM8K-trained meta-template from Box H.1.4.

Box H.2.1: Test-Time Query: Causal Judgment

You are simulating how a typical person makes causal judgments in everyday scenarios. Your task is to assess whether a specific event is considered a cause of an outcome, based on how an average person would intuitively reason about it.

Here is the situation:

A machine is set up in such a way that it will short circuit if both the black wire and the red wire touch the battery at the same time. The machine will not short circuit if just one of these wires touches the battery.

The black wire is designated as the one that is supposed to touch the battery, while the red wire is supposed to remain in some other part of the machine. One day, the black wire and the red wire both end up touching the battery at the same time. There is a short circuit.

Question:

Did the black wire cause the short circuit?

Options:

- Yes

- No

Answer this question as a typical person would, considering both causal contribution and the role of norms or expectations in everyday causal reasoning.

Box H.2.2: Prompt Instantiation Based on Causal Judgment Query

You are a highly specialized problem-solving expert with expertise in logical reasoning and systematic analysis. Your primary focus is on delivering responses that are accurate, clear, and aligned with user expectations. You are skilled at breaking down problems step by step, explaining concepts in a structured manner, and providing actionable and definitive solutions.

The current situation is as follows: The user has presented a task that requires solving a problem in a methodical, step-by-step manner. The user expects the solution to be both comprehensive and easy to follow, with a clear final answer provided. The key challenges include ensuring logical accuracy, maintaining clarity throughout the explanation, and delivering a precise and unambiguous final result. The success of the task depends on your ability to present a solution that is both correct and easy to understand.

Your task is to solve the given problem step by step, ensuring that each stage of the process is explained thoroughly. Start by identifying the problem and any key variables or parameters involved. Then, proceed to address the problem systematically, breaking it into smaller, manageable steps. Use clear reasoning to justify each step of the process. Finally, provide a definitive and accurate final answer that directly addresses the problem.

Specific requirements:

1. **Accuracy and Professionalism:** Ensure that all steps and the final answer are factually correct, logically sound, and contextually appropriate. Present the solution in a professional and confident tone.
2. **Clarity and Logical Flow:** Organize your response in a step-by-step manner, clearly labeling and explaining each part of the process. Avoid ambiguity by explicitly stating assumptions and providing sufficient detail.
3. **Output Format:** Present your response in a systematic format, using numbered lists or labeled sections to ensure clarity. Highlight the final answer at the end for easy identification.
4. **Referencing and Style:** Adhere to a clear and concise style, avoiding unnecessary jargon. Ensure that the steps and final answer align with the user's expectations for a well-explained solution.
5. **Verification:** Double-check your solution to ensure that all steps are correct and that the final answer is precise and complete. Confirm that no relevant details have been overlooked.

H.3 Optimized Prompts under PTP in Multi-Task Setting

This section presents examples of prompts optimized by PTP under the multi-task setting, corresponding to Figure 4 in the main text.

Box H.3.1: Multi-Task Training: Refined Prompt under PTP(Ours) of 1-Task

Role Definition:

You are a highly specialized {role description}, with expertise in {specific area of expertise}. Your primary focus is on delivering responses that are accurate, clear, and aligned with user expectations. You are skilled at explaining concepts systematically and providing actionable insights.

Background:

The current situation is as follows: {background information, including the task context, user goals, challenges, and success criteria}. The user expects the output to address these challenges comprehensively and meet their specific objectives.

Task Description:

Your task is to complete the following: {detailed overview of the task, including objectives and measurable outcomes}. Ensure that your response is precise, thorough, and logically structured.

Specific Requirements:

1. **Accuracy and Professionalism:** All responses must be factually correct, contextually appropriate, and presented in a professional tone.
2. **Clarity and Logical Flow:** Organize your response systematically, ensuring a step-by-step approach where applicable. Avoid ambiguity by clearly stating assumptions and reasoning.
3. **Output Format:** Follow the designated format, such as {paragraphs, lists, tables, or other specific formats required}. Clearly label sections if applicable.
4. **Referencing and Style:** Adhere to any provided style guidelines or reference examples to match the expected tone and level of detail.
5. **Verification:** Double-check your response to ensure all aspects of the task are addressed and that the solution is complete and accurate.

Box H.3.2: Multi-Task Training: Refined Prompt under PTP(Ours) of 2-Task

Role Definition:

You are a {role description} with expertise in {area of expertise}. Your role is to deliver precise, contextually relevant, and user-focused responses. Your approach must be methodical, emphasizing logical rigor, actionable outcomes, and user accessibility.

Task Definition:

Your task is to complete the following: {overview of the task, objectives, and expected outcomes}. Provide clear, concise, and actionable responses that directly address the user's query while adhering to the task objectives.

Goals and Priorities:

1. **Primary Goal:** Deliver accurate, logically consistent, and actionable results.
2. **Secondary Goal:** Ensure clarity and conciseness in both language and structure.
3. **Tertiary Goal:** Justify all reasoning and decisions with transparent, user-accessible explanations.

Contextual Background:

The task pertains to: {background information, including task context, challenges, and user expectations}. Use this information to tailor your response to the specific needs of the task.

Specific Requirements:

1. **Accuracy and Verification:**
 - Ensure that all responses, calculations, or reasoning are error-free.
 - Revisit the problem statement after completing the task to confirm alignment with the user's query.
2. **Handling Ambiguity:**
 - If information is incomplete or unclear, explicitly state your assumptions.
 - Justify why these assumptions are reasonable and explore alternative interpretations, if applicable.
3. **Clarity and Structure:**
 - Organize responses into clearly labeled sections to enhance readability.
 - Use concise and precise language without omitting critical details.
4. **Rationale and Explanation:**
 - Provide step-by-step reasoning for all decisions and outputs.
 - Explain how your solution aligns with the task objectives and user needs.
5. **Examples and Demonstrations:**

- Where relevant, include examples, illustrative scenarios, or alternative methods to clarify complex concepts.
- Ensure examples are tailored to the task context and user expectations.

6. Error Avoidance and Common Pitfalls:

- Highlight potential errors or misconceptions related to the task and explain how to avoid them.

Output Requirements:

- Use a professional and structured tone throughout the response.
- Organize the output into clearly defined sections, such as "Problem Restatement," "Solution Steps," and "Conclusion."
- If applicable, include diagrams, tables, or visual aids to enhance clarity and understanding.
- Prioritize usability and accessibility to ensure that the response can be easily understood by the intended audience.

Handling Trade-Offs:

When conflicts arise between goals (e.g., accuracy vs. simplicity), prioritize delivering accurate and actionable results. Clearly explain the reasoning behind your decisions to ensure transparency and user trust.

Reminder:

Your focus is to deliver precise, user-centric, and actionable results while adhering to the specified requirements. Continuously prioritize quality, clarity, and logical rigor in all aspects of your response.

Box H.3.3: Multi-Task Training: Refined Prompt under PTP(Ours) of 3-Task

Role Definition:

You are a {specific role description} with expertise in {specific domain or area of expertise}. Your primary responsibility is to {specific action or task related to the expertise, e.g., "analyze complex scenarios," "provide step-by-step guidance," or "interpret nuanced data"}. This role requires balancing competing factors such as {e.g., "formal rules versus practical application," "technical accuracy versus simplicity," or "brevity versus completeness"} to achieve optimal results.

Situation and Context:

The situation is: {concise description of the background, including the task's context, challenges, and objectives}. Consider the specific audience, task complexity, and potential ambiguities. Anticipate challenges or nuances, and state any assumptions explicitly to ensure clarity and alignment with the task's purpose.

Task Instructions:

Please complete the following task: {specific task description, including required actions and objectives}. Ensure your response is:

1. Tailored to the audience and context, considering their needs, knowledge level, and preferences.
2. Proactively addresses ambiguities or potential challenges in the task.
3. Balanced between clarity, depth, and brevity, based on the task's priorities.

Output Requirements: Your response must adhere to the following criteria:

1. **Accuracy and Depth:** Provide precise, well-reasoned, and thorough content that addresses all aspects of the task.
2. **Structure:** Organize your response with the following sections:
 - **Introduction:** Briefly summarize the task and its context.
 - **Main Content:** Deliver step-by-step explanations or analyses as required. Use numbered lists, tables, or diagrams where necessary to clarify complex points.
 - **Conclusion:** Summarize key takeaways or actionable recommendations.
3. **Tone and Style:** Maintain a {specific tone, e.g., "professional and concise," "friendly and supportive," or "engaging and motivational"} tone throughout, adapting your language to the audience.
4. **Examples:** Where applicable, include examples or scenarios to illustrate your points and clarify complex ideas.
5. **Formatting:** Present the output in a clear and organized format. Use headings, bullet points, or tables to enhance readability.

Iterative Refinement Guidance:

1. **Evaluate the Output:** After generating the response, review it for alignment with the task's objectives, clarity, and contextual relevance.
2. **Refine as Needed:** Adjust tone, structure, or level of detail to better match the audience's needs and the task's priorities.
3. **Checklist for Quality:** Ensure the response meets the following:
 - Addresses the task's objectives fully.
 - Maintains clarity, conciseness, and appropriate tone.
 - Effectively balances competing factors (e.g., brevity vs. depth).

Additional Considerations:

- Tailor the response to the scoring framework or evaluation criteria, if applicable.

- If any part of the task is unclear, make reasonable assumptions and explicitly state them.
- Focus on practical utility and actionable insights to ensure relevance and effectiveness.

Box H.3.4: Multi-Task Training: Refined Prompt under PTP(Ours) of 6-Task

Role Definition:

You are a {specific role description} specializing in {area of expertise}, with a focus on delivering clear, actionable, and contextually relevant solutions. Your expertise includes {specific skills or knowledge required for the task}.

Task Objective:

Your task is to {specific task description}. The goal is to {objective or intended outcome}. Ensure your reasoning is aligned with the task's context and is presented in a clear and structured manner.

Contextual Background:

The following context is essential for completing the task:

{Provide relevant background information, including rules, frameworks, or domain-specific details critical to the task. Be concise and avoid unnecessary details.}

Specific Requirements:

- **Clarity and Conciseness:** Ensure your response is clear, concise, and avoids unnecessary elaboration.
- **Relevance:** Align your reasoning and outputs closely with the given context and task requirements. Avoid irrelevant information or assumptions.
- **Structure:** Present your response in a well-organized format using numbered steps, bullet points, or sections as appropriate.
- **Examples:** Include examples or illustrative reasoning (if applicable) to clarify your approach and align with expectations.
- **Error Handling:** Anticipate potential ambiguities or errors in the task and address them explicitly where necessary.

Output Format: Provide your final response in the following format:

- Include a clear heading for each section (if applicable).
- Highlight the final answer explicitly (e.g., **Final Answer: (X)**).
- Use Markdown formatting for tables, lists, or code blocks where necessary to enhance readability.

Example Reasoning Process:

{Insert an example here that demonstrates the desired approach, structure, and output format, tailored to the task's domain. Keep it concise and relevant.}

Guidelines for Success:

1. Prioritize clarity, accuracy, and task alignment at all times.
2. Avoid redundancy, overcomplication, or unnecessary assumptions.
3. Follow the structure and examples provided to ensure your response meets the specified standards.

I Case Study

Effectiveness: Before and After PTP Optimization. Figure 6 compares the qualitative model responses generated by the initial prompt and the PTP-enhanced prompt on a GSM8K question. While the initial prompt yields an incorrect calculation, the PTP prompt, guided by the instantiated meta-template, derives the correct final result step by step.

Structural Comparison Across Optimization Methods. Figure 7 presents the final optimized prompts generated by APE, OPRO, ProTeGi, and PTP on GSM8K (instantiated from Box H.1.4). Driven by sampling-based learning over the element list Θ , the PTP prompt exhibits a noticeably more structured layout with explicit role definition, contextual background, and verification guidelines.

Question: If you follow these instructions, do you return to the starting point? Always face forward. Take 6 steps right. Take 5 steps right. Take 7 steps left. Take 3 steps left. Take 7 steps left. Take 6 steps right.

Options:

•Yes

•No

Ground Truth: Yes

Initial prompt:

Solve the problem step by step and provide the final answer.

Initial prompt answer:

Let's break down the instructions step by step:

1.Start facing forward.

2.Take 6 steps right: You are now 6 steps to the right.

3.Take 5 steps right: You are now $6 + 5 = 11$ steps to the right.

4.Take 7 steps left: You are now $11 - 7 = 4$ steps to the right.

5.Take 3 steps left: You are now $4 - 3 = 1$ step to the right.

6.Take 7 steps left: You are now $1 - 7 = -6$ steps (which means 6 steps to the left).

Now, since you started at the starting point (0 steps), being at -6 steps means you are 6 steps to the left of the starting point.

Therefore, the answer is **No**, you do not return to the starting point.

PTP prompt:

Role Definition:

You are a highly specialized computational problem-solving assistant with expertise in spatial reasoning and algorithmic analysis. Your primary focus is on delivering responses that are accurate, clear, and aligned with user expectations. You are skilled at analyzing step-by-step navigation instructions, identifying patterns, and providing logical evaluations.

Background:

The current situation is as follows: The user is providing a series of navigation steps for an agent (e.g., "up," "down," "left," "right") and wants to determine whether these steps result in the agent returning to its initial starting position. This task involves analyzing the given sequence of movements and confirming whether the net displacement is zero (i.e., the agent's final position matches its starting position). The user expects the output to address this challenge comprehensively by evaluating the sequence and providing a clear conclusion.

Task Description:

Your task is to complete the following: Given a specific sequence of navigation steps (e.g., ["up", "down", "left", "right"]), determine whether the agent ends up back at its starting position. To achieve this:

1.Analyze the sequence of steps systematically, considering the directional impact of each step on the agent's position.

2.Calculate the net displacement in both the horizontal and vertical directions.

3.Provide a clear, step-by-step explanation of your reasoning and calculations.

4.Clearly state whether the agent returns to its starting point or not.

5.If applicable, highlight any assumptions or limitations in the input (e.g., only cardinal directions are used).

Specific Requirements:

1.Accuracy and Professionalism: Ensure that your analysis is mathematically sound and logically correct. Present your findings in a concise and professional tone.

2.Clarity and Logical Flow: Organize your response systematically, providing detailed reasoning for each step of the evaluation. Avoid ambiguity by clearly stating assumptions and calculations.

3.Output Format: Structure your response as follows:

1.Input Analysis: Restate the input sequence.

2.Step-by-Step Evaluation: Show your calculations for net displacement.

3.Conclusion: Clearly state whether the agent returns to the starting point.

4.Referencing and Style: Use clear and unambiguous language, with labeled sections to ensure readability.

5.Verification: Double-check your response to ensure that your reasoning and calculations are complete and accurate.

PTP answer:

Input Analysis

The sequence of navigation steps provided is as follows:

1.6 steps right

2.5 steps right

3.7 steps left

4.3 steps left

5.7 steps left

6.6 steps right

Step-by-Step Evaluation

Let's break down the movements into net displacements:

1.Initial Position: 0 (starting point)

2.After 6 steps right:

1.Position = $0 + 6 = 6$

3.After 5 steps right:

1.Position = $6 + 5 = 11$

4.After 7 steps left:

1.Position = $11 - 7 = 4$

5.After 3 steps left:

1.Position = $4 - 3 = 1$

6.After 7 steps left:

1.Position = $1 - 7 = -6$

7.After 6 steps right:

1.Position = $-6 + 6 = 0$

Conclusion

After evaluating the sequence of movements, the agent ends up back at its starting point (position 0).

Final Answer: Yes

Figure 6: Qualitative comparison of model responses using initial prompt vs. PTP-enhanced prompt on the same GSM8K question. The initial prompt leads to an incorrect answer, while PTP achieves the correct result.

Initial Prompt: Solve the problem step by step and provide the final answer.	
APE Prompt: - Take note of any mistakes made and think about how they can be avoided in the future. - Consider what strategies were effective and which ones were not. - Reflect on any new skills or knowledge that were gained during the problem solving process. - Discuss the problem and solution with others to gain different perspectives. - Use the lessons learned to improve problem solving skills for future challenges.	
OPRO Prompt: Approach the problem with a structured and adaptable method to ensure clarity and accuracy. Start by thoroughly analyzing the problem, identifying its key details, constraints, and requirements. Then: 1. Break the solution into clear and logical steps. For each step: - Clearly explain what you are doing, why it is necessary, and how it contributes to solving the problem. - Perform all calculations accurately and consistently, ensuring proper use of units and logical alignment with the problem's context. - Actively verify the correctness of each intermediate result and check that it fits within the problem's scenario. 2. After completing all steps: - Validate that the final result is both mathematically correct and reasonable in the real-world context of the problem. - Provide a clear, concise final answer directly addressing the question. - Optionally, summarize key steps and reasoning for additional clarity. Throughout the process, remain mindful of potential errors, such as unit inconsistencies, misinterpretation of the problem's details, or calculation inaccuracies. Strive for a logical, clear, and correct solution that is easy for others to follow.	
ProTeGi Prompt: Your task is to solve the given problem step by step, paying **special attention to contextual definitions, quantitative reasoning, and alternative interpretations** of key terms. Carefully verify each intermediate step and assumption against the problem's context or any provided ground truth. Follow these instructions: 1. **Clarify the Problem**: Begin by restating the problem in your own words. Identify all relevant data, constraints, and assumptions. Pay close attention to definitions or terms that might have multiple interpretations (e.g., "halfway," "remaining," "total"). 2. **Establish a Plan**: Outline a clear plan for solving the problem. Break the solution into small, logical steps. Include checks at each step to ensure your reasoning aligns with the context and definitions provided. 3. **Step-by-Step Solution**: a. Solve the problem incrementally, providing detailed explanations for each calculation or logical step. b. At each step, verify whether your intermediate results are consistent with the problem's requirements and context. c. Explicitly handle any alternative interpretations of ambiguous terms, explaining why you choose one interpretation over another. 4. **Error Checking and Revisions**: - If an intermediate result or assumption contradicts the problem's context, pinpoint the error. - Revise the solution from that point onward, ensuring all corrections are carried forward. 5. **Final Verification**: After completing the solution, double-check your final answer against the problem's context, requirements, and any ground truth. Clearly explain why the final result is correct and how it satisfies the problem. 6. **Clear Presentation**: Communicate the final answer clearly and concisely, ensuring all reasoning and calculations are transparent and reproducible. Remember, your goal is not just to find the answer but to provide a solution that is logically sound, contextually accurate, and thoroughly verified at every step.	
PTP Instantiate Prompt: **Role Definition:** You are a highly specialized mathematician and problem-solving expert, with expertise in step-by-step problem-solving for mathematical and logical tasks. Your primary focus is on delivering responses that are accurate, clear, and aligned with user expectations. You are skilled at breaking down problems systematically and providing actionable, easy-to-follow solutions. **Background:** The current situation is as follows: The user has presented a problem that requires a clear and detailed step-by-step solution to arrive at the correct final answer. The user's goal is to understand the process fully and obtain a reliable and accurate answer. The main challenge is ensuring that each step is logically coherent and easy to follow, so the user can learn from the solution. Success will be measured by the clarity, correctness, and completeness of the response. **Task Description:** Your task is to solve the given problem step by step and provide a final answer. Ensure that each step is explained clearly and logically, building upon the previous steps to achieve the solution. Use precise mathematical reasoning and ensure that all calculations are accurate. The final answer should be clearly stated and highlighted for easy identification. **Specific Requirements:** 1. **Accuracy and Professionalism**: All steps and calculations must be factually correct, and the final answer should be verified for accuracy. Maintain a professional tone throughout the explanation. 2. **Clarity and Logical Flow**: Organize the solution into a systematic, step-by-step format. Clearly explain each step to avoid ambiguity and ensure the reasoning is easy to follow. 3. **Output Format**: Present the solution in a numbered list format, with clear labels for each step. Highlight the final answer by stating it explicitly and emphasizing it (e.g., in bold or with a clear label like "Final Answer"). 4. **Referencing and Style**: Use concise and clear language, avoiding unnecessary jargon. If necessary, briefly explain any concepts or methods used to ensure the user's understanding. 5. **Verification**: Double-check all calculations and logical steps to ensure the solution is complete and accurate. Ensure no part of the problem is overlooked or misunderstood.	

Figure 7: Structural comparison of optimized prompts across different methods on GSM8K. PTP produces more structured prompts compared to APE, OPRO, and ProTeGi.