ProtoPrompt: Multi-perspective Prompt Optimization via Prototypical Feedback

Minhyuk An[1,3], **Tae-Hoon Kang**[2,3], **Haerin Byeon**[1,3], **Sung-Bae Cho**[2,3]
[1]Department of Artificial Intelligence
[2]Department of Computer Science
[3]Yonsei University
{als7928, thkyang324, byeonn, sbcho}@yonsei.ac.kr

Abstract

Large language models (LLMs) have demonstrated strong performance across a wide range of NLP tasks, yet their effectiveness remains highly sensitive to prompt design. To address this issue, automated prompt optimization (APO) has emerged as a promising direction. Existing APO approaches typically refine prompts using natural language feedback, but each step often relies on a single perspective. This narrow focus biases the base LLM towards one viewpoint, leading to overfitting and inefficiency in both the step-by-step reasoning process and the overall search procedure, thereby increasing computational cost. In this paper, we propose ProtoPrompt, a novel method that integrates Monte Carlo tree search (MCTS) with a prototypical feedback mechanism. Our method enables the optimizer LLM to generate diverse perspectives and identify the most representative feedback. This mechanism provides the base LLM with richer and more balanced guidance throughout the search trajectory. Experiments on six BIG-Bench tasks and three general natural language understanding (NLU) benchmarks show that ProtoPrompt achieves up to a 1.64% improvement over strong baselines, converges faster with fewer iterations, and reduces overall cost by 18.19%. Our code is publicly available at https://example.com.

1 Introduction

Large language models (LLMs) have rapidly advanced the state of the art in a wide spectrum of natural language processing (NLP) tasks, ranging from common sense reasoning and question answering to instruction following and step-by-step problem solving (Brown et al., 2020; OpenAI et al., 2023; Sahoo et al., 2024). Their ability to generalize from in-context examples and natural language instructions has driven widespread adoption in both academic and industrial applications (Lampinen et al., 2025; Bubeck et al., 2023; Wei et al., 2022; Ouyang et al., 2022). However, despite these successes, the performance of LLMs remains highly sensitive to prompt engineering (Sahoo et al., 2024; Cao et al., 2024; Wei et al., 2022; Chen et al., 2023; Zamfirescu-Pereira et al., 2023). Even small variations in prompt wording, structure, or exemplars can yield large differences in accuracy, reasoning quality, and robustness (Wei et al., 2022; Fernando et al., 2024; Zhao et al., 2021). Consequently, effective prompt engineering has emerged as a crucial yet challenging component of the deployment of LLMs at scale (Sahoo et al., 2024).

Traditional prompt engineering often relies on manual trial-and-error and human expert intuition, which are costly, inconsistent, and difficult to scale (Fernando et al., 2024). To reduce dependence on human-crafted prompts, recent approaches have therefore turned to automated prompt optimization (APO) (Pryzant et al., 2023; Shin et al., 2023), where prompts are iteratively improved without relying on human-crafted exemplars using LLMs. Based on natural language feedback, these methods have made important progress by enabling prompts to dynamically adapt across tasks and domains (Wang et al., 2023b; Madaan et al., 2023). However, they remain vulnerable to instability: Optimization often amplifies the noisy or misleading context, causing the resulting prompts to inherit narrow biases or overfit to a specific feedback set (Chu et al., 2024). Consequently, such search processes require a large number of trials, increasing the computational cost and exploration burden of identifying effective prompts. (Ramnath et al., 2025)

Addressing this challenge requires a new perspective on how feedback itself is represented and used. Instead of treating feedback as a single correction or exemplar, it can be viewed as a distribution of plausible perspectives (Saba, 2023; Wang et al., 2023a). Identifying the feedback that lies closest to the semantic center of this distribution (Elekes et al., 2017; Saba, 2023), we can exploit the feedback that is maximally representative while limiting the misleading or overly specific ones. This approach reduces the risk of propagating noisy information and, critically, reduces the number of unproductive exploration paths during optimization. In this way, editing feedback becomes a principled mechanism for stabilizing the search process and ensuring that prompts evolve based on feedback that are both informative and generalizable.

Motivated by this, we propose ProtoPrompt, an APO-based method that introduces prototypical feedback (PF) as a core mechanism. The key insight is that unreliable optimization stems from treating any single feedback—whether overly specific, noisy, or biased—as equally valid guidance, consistent with prior observations that noisy signals can degrade the quality of optimized prompts (Li et al., 2023). This information, when propagated through iterative search, can easily lead to spurious exploration paths and increased search cost. To address this, ProtoPrompt reframes the feedback as a distribution of plausible corrections derived from the errors of the base model. In practice, the optimizer LLM generates a diverse set of feedback candidates, each reflecting a distinct perspective on the observed error from the base LLM. These candidates are then embedded into a dense vector space, forming a semantic distribution of potential corrections. Within this space, ProtoPrompt computes a centrality to identify the feedback that lies closest to the semantic center. The selected prototype functions as the most representative correction: It retains essential task-relevant semantics while filtering out misleading or overly narrow details.

We validate ProtoPrompt on a diverse set of benchmarks spanning structured reasoning tasks (e.g., tabular inference, geometric recognition, temporal and causal reasoning) (Suzgun et al., 2023; Srivastava et al., 2023) and general NLU benchmarks (Pang and Lee, 2004; Voorhees and Tice, 2000; De Marneffe et al., 2019) (e.g., subjectivity classification, question classification, and natural language inference). The results show that ProtoPrompt achieves performance improvements in several tasks, while also reducing the cost of exploration.

Our contributions are as follows:

- We identify that feedback-based prompt optimization can be undermined by misleading or noisy feedback, which propagates errors during search and leads to inefficient exploration.
- We propose a novel method, ProtoPrompt, that introduces prototypical feedback (PF) as a mechanism for the most representative feedback from a distribution of feedback candidates.
- Through extensive experiments on various tasks, we show that ProtoPrompt delivers performance improvements while reducing cost, demonstrating both the effectiveness and efficiency of PF in guiding prompt optimization.

2 Related Work

Chain-of-thought. Chain-of-thought (CoT) prompting (Wei et al., 2022; Kojima et al., 2022) has emerged as a widely used technique to improve the reasoning capabilities of LLMs by decomposing complex problems into a sequence of intermediate steps. This structured reasoning process has proven effective in a variety of tasks, including arithmetic, logic, and commonsense inference. ALthough CoT prompting improves interpretability and multi-hop reasoning, recent studies highlight its vulnerability to overfitting (Saba, 2023; Chu et al., 2024), as LLMs tend to latch onto spurious patterns or overly specific details present in intermediate steps. Such an over-reliance can lead to degraded performance when the CoT trace includes noisy or biased content. Furthermore, human-crafted static CoT exemplars limit their flexibility and adaptability across tasks and model variants. Our work builds on these insights by aiming to generalize intermediate reasoning steps to prevent such overfitting and improve the robustness of CoT-style prompting methods.

Automated prompt optimization. Prompt design has long been recognized as a critical determinant of LLM performance. Since manual prompt engineering is costly and error-prone, recent research has focused on automated prompt optimization. Automatic prompt engineering (APE) (Zhou et al., 2022) demonstrated that LLMs could be used to iteratively paraphrase instructions, score them against a validation set, and retain higher performing variants. Building on this direction, OPRO (Yang et al., 2023) conceptualizes LLMs themselves as prompt optimizers. OPRO iteratively generates candidate prompts using an optimizer LLM and scores them using a scorer LLM in a black-box loop, similar to evolutionary optimization. UniPrompt extends the optimization problem to a multi-task setting. By decomposing the prompts into multiple sections and editing them in a structured way, UniPrompt (Juneja et al., 2025) demonstrates improved performance and generalization between tasks. In our study, we extend this iterative process by encouraging the optimizer to consider multiple diverse feedback candidates and then to score them to exploit the most representative.

Planning-based prompt optimization. PromptAgent (Wang et al., 2023b) reframes prompt optimization as a strategic planning problem. Instead of greedy iteration, PromptAgent employs Monte Carlo tree search (MCTS) to explore the vast space of candidate prompts. Similarly to other recent approaches, the design separates the roles of two models: a base LLM, whose performance is optimized, and an optimizer LLM, which identifies errors and generates corrective feedback. The optimization loop proceeds as follows. First, the base model is evaluated on a validation set. When the base LLM makes an error, the optimizer LLM identifies the mistake and generates natural language feedback that highlights missing constraints, misinterpretations, or inadequate reasoning strategies. Feedback is treated as an action in the MCTS framework, leading to new updated prompts (states). Each new prompt is reevaluated in the task, and the resulting reward is backpropagated through the search tree. However, a limitation of this approach is that the feedback generated for each action can be highly diverse and random, meaning that even misleading or incorrect feedback may be propagated as reward during search. In our work, we retain the strengths of MCTS-based strategic exploration, while mitigating this weakness by searching through a diverse distribution of feedback candidates and selecting the most representative feedback to guide the subsequent search.

3 Motivating Example

To motivate the prototypical feedback (PF) mechanism, we first examine how different forms of feedback influence reasoning performance (Figure 1). In a CoT setting, the model relies on a single exemplar to guide its reasoning. When providing

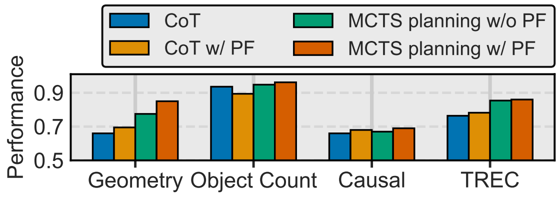

Figure 1: Motivating example comparing feedback strategies. PF represents prototypical feedback that provides generalized, representative guidance.

an exemplar which encodes highly concrete cues, such as domain-specific lexical patterns and exact symbolic templates, the model achieves strong performance. However, this success is largely attributable to exploiting surface correlations rather than cultivating genuine reasoning ability.

When we manually replace concrete exemplars with more generalized ones that capture only the essential structure of the task (CoT w/PF), we observe improved performance on Geometry, Causal Judgment (Causal), and TREC, but performance degradation on Object Counting. This highlights a fundamental trade-off: while generalizing information can aid task performance, certain tasks benefit more from specific cues. However, manually constructing prompts to strike this balance is extremely difficult. Therefore, we investigate how PF influences when incorporated with strategic planning methods such as MCTS.

When prompts are automatically searched using MCTS, performance improvements are generally observed; however, in certain cases such as Causal, the performance can even degrade. In contrast, applying PF consistently achieves the highest performance across all tasks, while in some cases like CoT w/ PF the gains may be less pronounced. This strong synergistic effect provides the motivation for employing PF in our framework.

4 Methodology

4.1 Overview

Our proposed method, ProtoPrompt, is illustrated in Figure 2. ProtoPrompt is designed for automated prompt optimization and operates within an MCTS–based planning framework that addresses the underlying search problem. The system consists of two interacting models with distinct roles: a base LLM, which performs reasoning given an input prompt, and an optimizer LLM, which identifies errors in the base LLM's outputs and provides

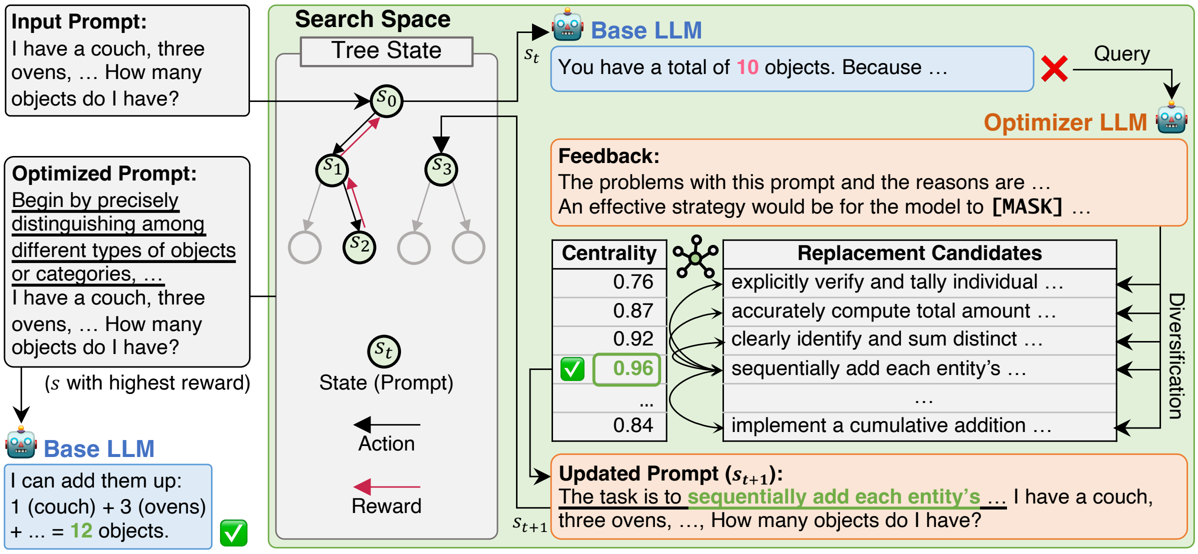

Figure 2: Overall framework of ProtoPrompt. The process begins with a search space of candidate prompts, which are iteratively expanded through MCTS. At each state, the base LLM (GPT-4o-mini) generates task outputs, and an optimizer LLM (GPT-4-turbo) provides multiple feedback candidates. These feedback candidates are embedded into dense vector space, and the prototypical feedback reflecting high centrality across candidates is selected to update the prompt. The updated prompt forms a new state in the search tree, where its quality is evaluated via reward signals. Through repeated selection, expansion, simulation, and backpropagation, MCTS identifies prompts that maximize performance with efficient exploration.

corrective feedback.

The optimizer LLM aims to deliver general and representative feedback, avoiding solutions that are overly specific to a single instance. To achieve this, before providing feedback, it generates a diverse set of unique feedback candidates, each capturing a different perspective on the identified error. These candidates collectively approximate the broad distribution of plausible corrections derivable from the base LLM's error.

Each candidate, expressed in natural language, is mapped into a dense vector space using a language embedding model. Within this space, we compute an inner product–based centrality score to locate the feedback that lies closest to the semantic center of the candidate distribution. The selected feedback thus represents the most prototypical correction: it is maximally representative of the distribution, while being robust against noisy, overly specific details or spurious biases. This feedback is then used to optimize the prompt, which yields a new state in the MCTS search process.

4.2 Problem Definition

We consider a base language model $\mathcal{O}$ tasked with solving the questions $q \in \mathcal{Q}$. A prompt s conditions the model output, that is, $y = \mathcal{O}(q|s)$. The objective of prompt optimization is to discover an improved prompt s^* that maximizes expected task performance:

$$s^* = \underset{s \in \mathcal{S}}{\operatorname{argmax}}\ \mathbb{E}_{q \sim \mathcal{Q}}[R(\mathcal{O}(q|s))], \tag{1}$$

where R denotes the reward function, typically accuracy, F1, or task-specific evaluation metrics.

4.3 MCTS-based Strategic Planning

We exploit MCTS as the backbone of our optimization framework. Each node in the search tree represents a prompt state, and the edges correspond to feedback-guided modifications (action). The objective is to identify high-reward paths that converge toward expert-level prompts. Formally, the search tree is defined as $T = (\mathcal{S}, \mathcal{A})$, where each state $s \in \mathcal{S}$ corresponds to a candidate prompt, and each action $a \in \mathcal{A}$ corresponds to a feedback-driven modification. At iteration t, the algorithm excutes four canonical steps.

Selection. Starting from the root node s_0, the algorithm recursively selects the child nodes according to the upper confidence bound applied to trees (UCT) algorithm (Kocsis and Szepesvári, 2006):

$$a^* = \underset{a}{\operatorname{argmax}} \left[Q(s,a) + c \cdot \sqrt{\frac{\ln N(s)}{1 + N(s,a)}} \right], \tag{2}$$

where $Q(s, a)$ denotes the average reward of action a from state s, $N(s)$ is the number of visits to state s, $N(s, a)$ is the number of times action a has been taken, and $c > 0$ is a constant of tunable exploration. This balances exploitation of high-reward actions with exploration of less-visited braches.

Expansion. When a leaf node s_L with unvisited action is reached, we expand the tree by generating a new state. In our work, this process involves the optimizer LLM performing the action of generating prototypical feedback and subsequently creating a new prompt based on it, as described in Sections 4.4 and 4.5.

Simulation (Rollout). From the expanded state, we evaluate the updated prompt on a validation subset $\mathcal{D}_{\text{val}}$. The base LLM $\mathcal{O}$ produces outputs conditioned on the candidate prompt, and a reward is computed as:

$$R(s) = \frac{1}{\mathcal{D}_{\text{val}}} \sum_{(q,y)} \mathbf{1}\{\mathcal{O}(q|s) = y\}, \tag{3}$$

for classification tasks, or extended to task-specific metrics. This reward serves as a stochastic estimate of the value of the prompt.

Backpropagation. Finally, the reward obtained during the simulation is propagated backward along the path from the leaf node to the root. Each visited node updates its statistics:

$$Q(s, a) \leftarrow \frac{Q(s, a) \cdot N(s, a) + R}{N(s, a) + 1}, \tag{4}$$

$$N(s, a) \leftarrow N(s, a) + 1. \tag{5}$$

This process incrementally refines the value estimates for state-action pairs, allowing the tree to progressively focus search on high-reward prompts.

4.4 Multi-Perspective Feedback Augmentation

A key challenge in prompt optimization lies in the fact that the base LLM's error can often be explained from multiple plausible perspectives, each focusing on distinct deficiencies such as factual correctness, depth of reasoning, or linguistic clarity. Instead of constraining the optimization loop to a single feedback trajectory, we explicitly model feedback generation as a distributional process.

Algorithm 1 Prototypical Feedback Selection

Require: Prompt template: $\mathcal{T}$, Optimizer LLM: $\mathcal{O}'$, Text embedding model: g
Generate $(f, C) \leftarrow \mathcal{O}'(\mathcal{T})$
$F \leftarrow \varnothing$
for $i \leftarrow 1$ to m **do**
  for all $c_k \in C_i$ **do**
    $f \leftarrow$ replace $[MASK_i]$ with c_k
    $F \leftarrow F \cup f$
  end for
end for
Compute $\mathbf{E} \leftarrow g(F)$
Compute $I(f) = \sum_{f' \in F} \langle \mathbf{e}_f, \mathbf{e}_{f'} \rangle$
$f_p \leftarrow \arg\max_{f \in F} I(f)$
return f_p

Formally, given an erroneous output y from $\mathcal{O}$ under prompt s, we condition an optimizer LLM $\mathcal{O}'$ on (s, y) and obtain a feedback distribution.

$$\mathcal{F} \sim P(F \mid s, y; \mathcal{O}'), \tag{6}$$

where $F = f_1, f_2, ..., f_k$ denotes a set of feedback candidates. Each f_i represents a distinct hypothesis about how the prompt should be refined.

From a Bayesian perspective, $\mathcal{F}$ can be viewed as sampling from a posterior predictive distribution:

$$P(f \mid s, y) \propto P(y \mid f, s) P(f \mid s), \tag{7}$$

where $P(f \mid s)$ encodes the prior plausibility of a feedback statement given the prompt context, and $P(y \mid f, s)$ reflects the likelihood that the feedback explains the observed error y. By approximating this posterior with multiple feedback $f_1, ..., f_k$, we temporarily expand the search space to include diverse yet task-relevant information.

It is done using a specialized prompt template $\mathcal{T}$, which incorporates the prompt s (See Appendix F). Specifically, this template guides the optimizer LLM $\mathcal{O}'$ to generate initial feedback f and to identify exactly m segments within f. Each identified segment is then replaced with a unique placeholder token, denoted as $[\text{MASK}_i]$ for $i = 1, \ldots, m$, and for each mask token, the model generates k unique alternative candidates $c_1, c_2, \cdots, c_k\} \subseteq C_i$.

4.5 Prototypical Feedback Selection

After the diversification process, the centrality selection strategy is used to determine the most general and representative feedback. Let each feedback f replaced by c be assigned to a dense vector embedding $\mathbf{e}_f \in \mathbb{R}^d$ by a text embedding model $g : \mathcal{F} \to \mathbb{R}^d$. The similarity between two feedback candidates f and f' is quantified by the inner product $\langle \mathbf{e}_f, \mathbf{e}_{f'} \rangle$. We then define a score $I(f)$ for each candidate as the following equation:

$$I(f) = \sum_{f' \in F} \langle \mathbf{e}_f, \mathbf{e}_{f'} \rangle. \tag{8}$$

In practice, we compute the embedding matrix $\mathbf{E} \in \mathbb{R}^{k \times d}$, where each row corresponds to an embedding $\mathbf{e}_f$, and then obtain $I(f)$ by adding the rows of the matrix product $\mathbf{E}\mathbf{E}^\top$. Feedback candidates are ranked according to their scores, and we select the top-1 index as the most representative feedback f_p, marginalizing the others. This centrality score $I(f)$ reflects the degree to which a given feedback candidate represents the overall set in the embedding space. Since semantically similar candidates concentrate around high-density regions of $p(f \mid x)$, the score $I(f)$ naturally favors those near the centroid of the embedding distribution (Elekes et al., 2017; Saba, 2023; Fodor and Pylyshyn, 1988). For further details, we provide the Algorithm 1.

Finally, we generate the optimized prompt, which serves as the next node in the MCTS search trajectory for subsequent exploration using $\mathcal{O}$ and the selected f_p.

5 Experiments

5.1 Set-up

Datasets. Datasets are used to evaluate the effectiveness of prompt optimization in various reasoning and classification tasks from a subset of challenging BIG-Bench Hard (BBH) tasks (Suzgun et al., 2023; Srivastava et al., 2023) (Penguins in a table, Geometry, Epistemic Reasoning, Object Counting, Temporal Sequences, and Causal Judgment) and general natural language understanding (NLU) tasks, i.e., Subjective (Pang and Lee, 2004), TREC (Voorhees and Tice, 2000), and CB (De Marneffe et al., 2019). Each dataset is divided into training, validation, and test sets. A more detailed description is reported in the Appendix C.

Implementation details. In the experiment, we use the same evaluation strategy as PromptAgent. We adopt GPT-4o-mini as the base LLM $\mathcal{O}$, GPT-4-turbo as the optimizer LLM $\mathcal{O}'$, and the text embedding model g as `text-embedding-3-large` from OpenAI. In all experiments, we set m to 2 and k to 50 for the purpose of computational efficiency. We set the temperature to 0.0 for the base LLM and 1.0 for the optimizer LLM. We set the number of iterations for MCTS to 12, adding 3 children of the leaf node to the tree, the maximum depth to 8, the minimum depth to 2, and the exploration weight to 2.5 for UCT Algorithm. We select the test accuracy based on the node with the highest reward, which is calculated on $\mathcal{D}_{\text{val}}$.

Baselines. For baseline comparisons, we consider both human-crafted and APO techniques. The human prompt (ZS) and the few-shot version of the human prompt (FS) (Suzgun et al., 2023) baselines provide a reference for manually designed prompts, while CoT and the zero-shot version of CoT (ZS) (Wei et al., 2022; Kojima et al., 2022) serve as strong reasoning-based baselines. Meanwhile, we compare our method against strong baselines including OPRO (Yang et al., 2023), PromptAgent (Wang et al., 2023b), UniPrompt (Juneja et al., 2025).

5.2 Performance Analysis

Table 1 compares the proposed method with several baselines, including human baselines, CoT baselines, OPRO, PromptAgent, and UniPrompt, in both BBH and general NLU tasks. On the six BBH tasks, ProtoPrompt achieves the best results on three tasks and the second-best on two, with an overall average accuracy of 89.46%. This corresponds to a gain of +1.65 percentage points over the strongest baseline, UniPrompt (87.81%). On the three general NLU benchmarks, ProtoPrompt achieves the best result on one task and the second-best on the remaining two, resulting in the second-highest average. When evaluated across all tasks, including both human-crafted prompts and APO methods, ProtoPrompt achieves the highest overall average of 87.21%, surpassing UniPrompt (86.61%) by +0.6 percentage points. In addition, compared to PromptAgent, which also employs MCTS-based prompt optimization, ProtoPrompt achieves a further improvement of +1.65% on average. These consistent gains across heterogeneous benchmarks provide strong evidence that PF enhances both task-specific reasoning and language understanding capabilities in LLMs.

5.3 Ablation Study

Effect on accuracy. Table 2 shows that the incorporation of Prototypical Feedback (PF) consis-

Method	BIG-Bench Hard tasks							General NLU tasks				Avg.
	Penguins	Geometry	Epistemic	Object Count	Temporal	Causal Judge	Avg.	Subj	TREC	CB	Avg.	
Human (ZS)	98.73	48.00	84.40	78.80	91.00	64.00	77.49	66.80	65.60	80.36	70.92	75.30
Human (FS)	97.47	41.50	80.60	49.40	93.40	58.00	70.06	85.30	75.20	85.71	82.07	74.06
CoT (ZS)	97.47	55.50	82.00	93.20	90.80	65.00	80.66	67.80	65.00	**87.50**	73.43	78.25
CoT	97.47	69.50	87.40	93.80	96.80	66.00	85.16	79.60	77.40	85.71	80.90	83.74
OPRO	**100.00**	83.36	82.15	88.23	97.93	66.94	86.44	73.42	80.62	74.40	76.15	83.01
PromptAgent	98.73	77.50	**89.20**	94.80	**99.60**	67.00	87.81	73.90	85.40	82.14	80.48	85.36
UniPrompt	**100.00**	78.90	81.75	93.50	99.44	**69.89**	87.11	**88.19**	82.34	86.28	**85.60**	86.61
ProtoPrompt (Ours)	98.73	**85.00**	**89.20**	**96.20**	98.60	69.00	**89.46**	80.00	**86.00**	82.14	82.71	**87.21**

Table 1: Comparison across BBH tasks and General NLU tasks. Test accuracy (%) is reported.

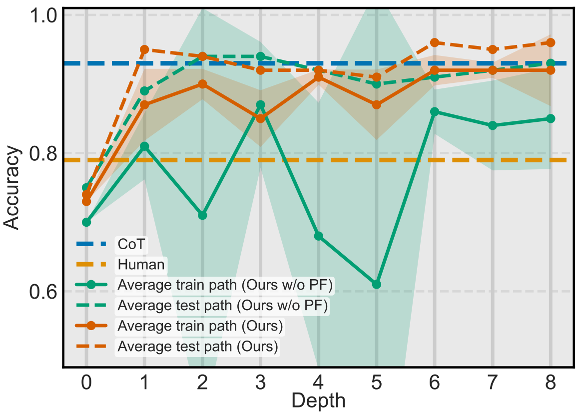

Figure 3: Convergence analysis of different methods based on tree depth in the task of Object Counting. Shaded regions indicate the variance.

Dataset	Accuracy ↑ (%)		Total cost ↓ ($)	
	w/o PF	w/ PF	w/o PF	w/ PF
BBH	87.81	**89.46** (+1.65)	50.22	**40.00** (-20.34%)
NLU	40.24	**41.36** (+1.12)	19.45	**17.00** (-12.59%)
Total	85.36	**87.21** (+1.85)	69.67	**57.00** (-18.19%)

Table 2: Comparison of performance and inference cost between our method without prototypical feedback (w/o PF) and with prototypical feedback (w/ PF).

tently improves task performance in both BBH and general NLU tasks. In BBH tasks, average accuracy increases from 87.81% to 89.46% (+1.65 points). Similarly, the results on general NLU show an improvement from 40.24% to 41.36% (+1.12 points). In particular, the aggregated accuracy between datasets increases by +1.85 points, underscoring the robustness of PF as a feedback mechanism to improve the quality of optimization.

Effect on inference cost. In addition to accuracy improvements, PF achieves notable reductions in inference cost. Without PF, the total cost of inference in all tasks amounts to $69.67, while the PF-enhanced approach lowers this to $57.00, corresponding to an 18.19% reduction. This cost efficiency arises from the ability of the PF to compress feedback information into more concise and representative prompts, which reduces redundant exploration and search iterations. As a result, PF not only strengthens performance, but also saves cost.

5.4 Convergence Analysis

Figure 3 shows the performance trends in varying tree depths for Human, CoT, ours without PF (w/o PF), and ours with PF (w/ PF) in the task of Object Counting. The trajectories illustrate the evolution of average performance during the training (reward) and testing steps. As tree depth increases, our model consistently outperforms the base model, demonstrating superior generalization by effectively narrowing the gap between training and testing performance. Notably, ours not only achieves higher performance after full exploration, but also shows a better result over Human and CoT even at earlier depths.

5.5 Prompt Generalization Analysis

To assess the transferability of the prompts between the base LLMs, we evaluate whether the optimized prompts in GPT-4o-mini retain their effectiveness when applied to another base LLM, GPT-o3-mini in Table 3. In particular, we observe that the optimized prompts maintain high performance in most tasks even when transferred to GPT-o3-mini. While other methods tend to degrade on the Causal Judge benchmark, our method exhibits improved performance. This indicates that generalized feedback not only enhances performance, but also yields prompts that are robust to model shifts, highlighting the practical utility of our method.

Task	GPT-4o-mini (source)			GPT-o3 mini (target)		
	PromptAgent	**UniPrompt**	**Ours**	**PromptAgent**	**UniPrompt**	**Ours**
Penguins	98.73	**100.00**	98.73	**100.00**	**100.00**	98.73
Geometry	77.50	78.90	**85.00**	97.50	93.50	**98.50**
Epistemic	**89.20**	81.75	**89.20**	88.40	87.20	**92.60**
Object Count	94.80	93.50	**96.20**	99.40	99.40	**99.60**
Temporal	**99.60**	99.44	98.60	**99.60**	99.20	99.20
Causal Judge	67.00	**69.89**	69.00	65.00	67.00	**70.00**
Subj	73.90	**89.19**	80.00	74.60	**89.70**	79.50
TREC	85.40	82.34	**86.00**	85.00	77.00	**87.80**
CB	82.14	**86.28**	82.14	85.71	**89.28**	87.50
Avg.	85.36	86.61	**87.21**	88.36	89.14	**90.38**

Table 3: Prompt generalization performance. Prompts optimized by GPT-4o-mini demonstrate effective transfer to other stronger LLM GPT-o3-mini.

Task	#Tokens		
	Human	**w/o PF**	**w/ PF**
Penguins	12	177 (+165)	12 (0)
Geometry	8	478 (+470)	62 (+54)
Epistemic	8	89 (+81)	50 (+42)
Object Count	17	476 (+459)	228 (+211)
Temporal	11	126 (+115)	169 (+158)
Causal Judge	6	481 (+475)	137 (+131)
Subj	14	265 (+251)	149 (+135)
TREC	50	490 (+440)	447 (+397)
CB	31	31 (0)	31 (0)

Table 4: Comparison of the final prompt length (#Tokens) optimized by different methods across tasks. We report human prompts (Human), prompts optimized without prototypical feedback (w/o PF), and with prototypical feedback (w/ PF). The numbers in parentheses indicate the difference in token count relative to Human.

5.6 Final Prompt Analysis

Table 4 reports the number of tokens in the final optimized prompts for different tasks. Human-written prompts are generally short and concise, serving as a baseline reference. When optimization is conducted without prototypical feedback (w/o PF), the resulting prompts become significantly longer than the human baseline, often exceeding by more than +400 tokens. This indicates that conventional optimization tends to rely on excessively verbose prompts to capture task-relevant information. In contrast, our method (w/ PF) consistently produces more compact prompts while maintaining effectiveness. These results demonstrate that PF enables the optimization process to distill essential task semantics into shorter prompts, thereby mitigating redundancy and improving efficiency.

6 Concluding Remarks

In this work, we investigate the challenge of feedback-driven prompt optimization, highlighting that misleading feedback can propagate errors, inflate the cost of the exploration, and undermine the stability of search. To address this, we propose ProtoPrompt, a framework that introduces prototypical feedback (PF) as a principal mechanism for selecting representative corrections from diverse candidates. By anchoring optimization to feedback that captures the semantic core of task requirements, ProtoPrompt mitigates spurious search paths and provides a more stable basis for iterative improvement. Our experimental evaluation demonstrated that ProtoPrompt yields performance improvements while requiring fewer trials, validating both the effectiveness and efficiency of PF. These results suggest that robust feedback selection is a key ingredient in the advancement of automated prompt optimization and the assurance that LLM reasoning remains accurate and generalizable.

Looking ahead, we envision PF as more than a mechanism for stabilizing prompt optimization. Since PF encapsulates the most representative signal from a wide distribution of feedback, it serves as a compact yet rich information. This property opens avenues for storing and reusing PF in tasks, enabling efficient memory usage and knowledge transfer. We anticipate that PF could form the basis for a foundation in diverse applications.

Limitations

Although effective in the general tasks tested, its performance in specialized domains could be still uncertain and may rather decrease. Moreover, our method can be computationally intensive, especially with many candidates. Furthermore, the use of another pre-trained model and additional algorithms raises concerns about increased computational costs. Our future work will focus on balancing generalization with the domain-specific details required in complex reasoning scenarios.

Ethical Considerations

We follow fundamental ethical principles to ensure responsible use of datasets while minimizing potential social harms. All datasets are sourced from publicly available materials and used in accordance with applicable privacy and copyright regulations.

References

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. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. arxiv. *arXiv preprint arXiv:2303.12712*.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. *Advances in Neural Information Processing Systems*, 37:69022–69042.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. Navigate through enigmatic labyrinth a survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1173–1203.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *Proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Abel Elekes, Martin Schaeler, and Klemens Boehm. 2017. On the various semantics of similarity in word embedding models. In *2017 ACM/IEEE Joint Conference on Digital Libraries*, pages 1–10.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Gurusha Juneja, Gautam Jajoo, Nagarajan Natarajan, Hua Li, Jian Jiao, and Amit Sharma. 2025. Task facet learning: A structured approach to prompt optimization. In *Findings of the Association for Computational Linguistics*, page 23473–23496.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer.

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*, volume 35, pages 22199–22213.

Andrew K Lampinen, Arslan Chaudhry, Stephanie CY Chan, Cody Wild, Diane Wan, Alex Ku, Jörg Bornschein, Razvan Pascanu, Murray Shanahan, and James L McClelland. 2025. On the generalization of language models from in-context learning and finetuning: a controlled study. *arXiv preprint arXiv:2505.00661*.

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

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*, volume 36, pages 46534–46594.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 271–es.

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

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. *arXiv preprint arXiv:2502.16923*.

Walid S Saba. 2023. Stochastic llms do not understand language: towards symbolic, explainable and ontologically based llms. In *International Conference on Conceptual Modeling*, pages 3–19. Springer.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2023. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

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

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 200–207.

Jiashuo Wang, Haozhao Wang, Shichao Sun, and Wenjie Li. 2023a. Aligning language models with human preferences via a bayesian approach. *Advances in Neural Information Processing Systems*, 36:49113–49132.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

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

A Reproducibility Statement

In all experiments, we used OpenAI's API and acknowledge that the experimental results may vary depending on the knowledge cutoff of the models provided by OpenAI[1]. The knowledge cutoff[2] for GPT-4o-mini and GPT-o3 mini is October

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

[2] `https://platform.openai.com/docs/models`.

2023, for GPT-4-turbo is December 2023, and the `text-embedding-3-small` model was released in January 2024. However, to ensure reproducibility, we provide our source code and the datasets used for public access, the data split details in Table 5, exact input prompts in Table 6, 8, and 9, and the templates in Table 7. In addition, we manually set the seeds to 42.

B Computing Environment and License

All experiments were conducted on Ubuntu 22.10, with an Intel(R) Xeon(R) Gold 6326 CPU (2.90GHz) and NVIDIA A100 GPUs (80GB). The source code for this paper is adapted from PromptAgent (Wang et al., 2023b) (licensed under Apache-2.0).

C Dataset Description

We use six tasks from Big-Bench[3] (Srivastava et al., 2023), and three common NLU tasks. The description of each task is reported in Table 5.

Task	Train	Val.	Test
Penguins in A Table	70	70	79
Geometry	150	150	200
Epistemic Reasoning	300	200	500
Object Counting	150	150	500
Temporal Sequences	150	150	500
Causal Judgement	90	90	100
Subjective	200	200	1000
TREC	200	200	500
CB	125	125	56

Table 5: The number of samples in the training, validation, and test data split.

Big-Bench tasks. The `Penguins in A Table task` presents a structured reasoning challenge in which the models must answer questions based on a tabular representation of Penguin attributes. Each row corresponds to an individual penguin, and attributes such as age, height, and weight are provided. It requires the model to extract relevant information from the table, perform numerical reasoning.

The `Geometry` task tests the model's ability to recognize and classify geometric shapes based on their scalable vector graphics (SVG) paths. It contains a variety of simple and complex geometric figures, and the task demands understanding of how sequences of drawing commands correspond to different shapes.

In the `Epistemic Reasoning` task, models must determine whether a given premise entails a hypothesis. It evaluates the model's ability to recognize epistemic distinctions, such as differentiating between beliefs and factual statements. Some statements involve indirect inferences or degrees of uncertainty, making it necessary for the model to reason beyond simple surface-level text matching.

The `Object Counting` task focuses on numerical reasoning by presenting scenarios in which objects are grouped or described in varying ways. The challenge requires the model to correctly count items while accounting for linguistic ambiguities, such as plural forms, implicit quantities, and collectively referenced objects. It is particularly useful for assessing how well the models handle numeracy and aggregation.

The `Temporal Sequences` task assesses the model's ability to understand and reason about chronological events. Given a sequence of activities, the model must infer when a particular event could have taken place. It involves constraints such as available time slots, activity durations, and event dependencies, which requires the model to accurately track and manipulate time-based information.

The `Causal Judgment` task evaluates the model's understanding of causal relationships. Given a scenario, the model must determine whether an event was a direct result of a preceding action or simply correlated. It is particularly challenging, as it requires distinguishing between correlation and causation, as well as recognizing implicit intentions behind actions.

General NLU. The `Subjective`[4] (Pang and Lee, 2004) consists of text classification examples in which the model must determine whether a given sentence is subjective or objective. Subjectivity is typically marked by personal opinions, emotions, or qualitative judgments, whereas objectivity is associated with factual statements. The challenge involves correctly identifying linguistic cues that signal subjective interpretation without being misled by neutral descriptive language.

The `TREC`[5] task (Voorhees and Tice, 2000) is a well-established question classification benchmark.

[3] https://github.com/google/BIG-bench.

[4] https://huggingface.co/datasets/SetFit/subj.

[5] https://huggingface.co/datasets/CogComp/trec.

Given a natural language question, the model must categorize it into one of several predefined types, such as entity, abbreviation, location, or numeric value. It tests the model's ability to interpret the intent behind a question and align it with a structured taxonomy of possible responses.

The CB[6] task (De Marneffe et al., 2019) is designed for natural language inference, requiring models to determine whether a hypothesis is entailed, contradicted, or neutral concerning a given premise. It contains sentences with varying levels of implicit meaning, requiring careful semantic interpretation to distinguish between direct entailment, contradiction, and unrelated statements.

D Computational Complexity

We analyze the computational complexity of ProtoPrompt, focusing on the training phase. Let N denote the mini-batch size and M the complexity of a single model inference. Let k denote the number of feedback candidates per masked segment m. A prompt is optimized via search and evaluation over training data. Its per-iteration complexity can be approximated as $\mathcal{O}'(NM + M)$. The term NM corresponds to prompt evaluation over a batch of training examples, and the additional M accounts for the optimizer LLM $\mathcal{O}'$'s generation.

ProtoPrompt first generates k candidates, resulting in kM inference cost from $\mathcal{O}'$. Then, it embeds all feedback candidates using an embedding model and computes a centrality score for each candidate. This process involves $\mathcal{O}'(k^2)$ inner product computations and an additional $\mathcal{O}'(k)$ for the final selection via argmax, leading to the total complexity $\mathcal{O}'(NM + M + kM + k^2 + k)$. While our complexity scales with k, this dependency is minor in practice because $k \ll M$; the number of candidates is far smaller than the cost of full model inference over a batch. Under this assumption, the total complexity simplifies to $\mathcal{O}'((N + k)M)$.

E Human Prompts

We report the human prompts used in our experiments in Table 6. These prompts serve as the base input for prompt optimization, with additional task-specific prompts covering task descriptions, questions, and answer formats incorporated for each sample.

[6] https://huggingface.co/datasets/aps/super_glue.

F Feedback Template

The exact template used to generate feedback is reported in Table 7. We create the template considering the planning process with mini-batching. This template facilitates a structured approach to analyzing incorrect examples, identifying specific segments, and generating semantic replacements.

G Final Prompts

We report the final optimized prompts, token usage, and accuracy for each dataset used in our experiments in Tables 10 to 18. Each table contrasts our ProtoPrompt method (w/ PF) with its counterpart without PF (w/o PF). Across datasets, our method consistently achieves higher accuracy while requiring fewer tokens.

H Details of Motivating Example

To ground the design of our prototypical feedback (PF) mechanism, we examine how the form of feedback embedded in exemplars influences model reasoning. While CoT prompting has been shown to elicit step-by-step reasoning, the nature of the exemplars whether highly concrete or abstract can substantially alter both performance and generalization.

Table 8 demonstrates a few-shot CoT prompt where the exemplar includes explicit error identification (highlighted in red) and rigidly structured reasoning cues (blue). This formulation enables the model to achieve strong performance on object counting tasks by leveraging surface-level heuristics, such as filtering out words that are not fruits. However, this success is fragile: the model tends to memorize lexical cues like clarinet or violin rather than developing robust symbolic reasoning. Prior studies have noted that LLMs often rely on such shallow correlations when presented with concrete cues.

Table 9 introduces PF, where concrete lexical references are replaced with generalized descriptions (highlighted in yellow). Instead of enumerating fixed categories, the exemplar instructs the model to distinguish between relevant and irrelevant objects. This abstraction promotes better generalization across tasks by emphasizing structural properties of the reasoning process. The approach resonates with evidence from prototypical representation learning, where abstraction enhances transferability by capturing task-invariant features. Yet, the object counting example reveals a limitation:

Task	Human prompt
Penguins in A Table	Answer questions about a table of penguins and their attributes.
Geometry	Name geometric shapes from their SVG paths.
Epistemic	Determine whether one sentence entails the next.
Object Counting	Questions that involve enumerating objects of different types and asking the model to count them.
Temporal Sequences	Answer questions about which times certain events could have occurred.
Causal Judgement	Answer questions about causal attribution.
Subjective	Given the text, choose between 'subjective' and 'objective'.
TREC	Tag the text according to the primary topic of the question. Choose from (A) Abbreviation, (B) Entity, (C) Description and abstract concept, (D) Human being, (E) Location, (F) Numeric value.
CB	Read carefully the following premise and hypothesis, and determine the relationship between them. Choose from 'contradiction', 'neutral' and 'entailment'.

Table 6: Human prompt used in the experiments.

removing specific anchors diminishes accuracy in tasks that inherently rely on discrete symbolic cues, exposing a trade-off between adaptability and task-specific precision.

Figure 1 quantifies this trade-off across Geometry, Object Counting, Causal Judgment, and TREC classification. On Geometry, Causal, and TREC, PF-enhanced CoT outperforms baseline CoT. In contrast, PF underperforms on Object Counting, confirming that excessive abstraction can weaken performance when precise symbolic filtering is essential. The introduction of Monte Carlo tree search (MCTS) planning further highlights the interaction between feedback form and strategic search. MCTS alone improves outcomes in Geometry and TREC but destabilizes reasoning in Causal Judgment, sometimes performing worse than baseline CoT. Crucially, combining PF with MCTS yields the best results across all tasks. This synergy arises because PF anchors exploration within semantically central regions of the feedback space, mitigating the brittleness of MCTS while enhancing its exploratory benefits.

Taken together, the motivating example underscores a fundamental tension: concrete feedback provides strong task-specific guidance but risks overfitting, whereas generalized feedback improves adaptability but may reduce performance on tasks requiring specificity. The integration of PF with MCTS reconciles this tension, providing both robust generalization and stable task execution. This dual role motivates our framework design, in which PF acts as a stabilizer of strategic planning, ensuring that search trajectories balance general semantic structure with context-sensitive detail.

Section	Description
Purpose	I'm writing prompts for a language model designed to handle various scenarios in a general and robust way. This means the model must identify or construct a solid plan that leads to correct, plan-oriented answers.
Current prompt	Below is my current prompt: {cur_prompt}.
Issues	Despite aiming for a plan-based solution, this prompt fails to address the following examples correctly: {example_string}.
Analysis	Please examine each incorrect example step by step. - Concentrate on how the existing plan (or lack thereof) leads to the wrong answer. - Pay special attention to any deficiencies in how the prompt organizes or outlines steps, rather than focusing on a single domain, version, or specific detail.
Requirements	Then, produce an integrated feedback that addresses these common plan-related issues collectively. Your feedback should highlight any overarching problems in the prompt's plan, propose corrections or improvements to that plan, and ensure that your advice remains sufficiently abstract and broadly applicable—avoid overly specific or domain-constrained details unless absolutely necessary for clarity.
Masking process	Next, identify exactly {mask_num} segments in your feedback that are too narrow, overly technical, or domain-specific. - Replace each identified sentence with a unique [MASK_n] placeholder (e.g. [MASK_1], [MASK_2], ...). - For each [MASK_n], propose {candidate_num} alternative candidates that broaden or generalize the concept, so the final plan remains applicable to a variety of scenarios.
Key reminders	- The goal is to understand the deeper, shared reasons for the planning failures and how to create a more robust plan overall. - The final feedback should be broadly applicable, rather than tailored to a single domain or overly specific detail. - The [MASK_n] replacements and candidate phrases should reflect more generalized or inclusive expressions.
Output formatting	Be sure that in the final feedback: 1. You only have {mask_num} total [MASK_n] tokens. 2. Each token has a different index (e.g., [MASK_1], [MASK_2], ... up to [MASK_{mask_num}]). 3. You provide exactly {candidate_num} candidates for each [MASK_n].
Example	Feedback without [MASK]: [Your feedback in a single consolidated paragraph or set of paragraphs, before inserting any [MASK]]. Feedback with [MASK]: [Your feedback text, but with the identified overly specific sentences replaced by [MASK_n]]. Then, for each mask token, provide candidates in the form: <START>{Candidates_[MASK_1]:[candidate_1, ..., candidate_num]}<END>.

Table 7: Template for generating structured error feedback.

Input
Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have? A: The most common wrong answer is 9. The mistake occurs when people count all listed objects instead of **filtering out the non-fruits**. Items such as **"clarinet," "flute," and "violin" are not fruits and should not be included in the total count.** Let's think step by step. We first identify the fruits on the list and include their quantity in parentheses: - blackberry (1) - nectarine (1) - plum (1) - strawberry (1) - banana (1) - orange (1) Now, let's add the numbers in parentheses: 1 + 1 + 1 + 1 + 1 + 1 = 6. The answer is 6. Q: I have three oranges, a piano, an apple and two clarinet. How many musical instruments do I have?
Output
We first **identify the fruits** on the list and include their quantity in parentheses: - orange (3) - apple (1) Now, let's add the numbers in parentheses: 3 + 1 = 4. The answer is 4.

Table 8: Few-shot CoT prompting with an exemplar. This setting incorporates explicit error identification, as highlighted in red, allowing the model to recognize common mistakes. The prompt in blue emphasizes logical reasoning but may lead to overfitting by enforcing rigid exmplar structures.

Input
Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have? A: The most common wrong answer is 9 , the total number of items in the list. Errors occur when people fail to differentiate between relevant and irrelevant objects. It is important to filter out **objects that do not belong to the target category**. Let's think step by step. We first identify the fruits on the list and include their quantity in parentheses: - blackberry (1) - nectarine (1) - plum (1) - strawberry (1) - banana (1) - orange (1) Now, let's add the numbers in parentheses: 1 + 1 + 1 + 1 + 1 + 1 = 6. The answer is 6. Q: I have three oranges, a piano, an apple and two clarinet. How many musical instruments do I have?
Output
We first identify the **objects that belong to the target** on the list and include their quantity in parentheses: - orange (3, not a musical instrument) - piano (1, musical instrument) - apple (1, not a musical instrument) - clarinet (2, musical instruments) Now, let's add the number of musical instruments: 1 (piano) + 2 (clarinets) = 3. The answer is 3.

Table 9: Few-shot CoT prompting with PF (CoT+PF). This setting refines exemplar by introducing a more generalized approach, highlighted in yellow, which guide the model in differentiating between relevant and irrelevant objects. The structured feedback promotes better generalization and mitigates overfitting risks.

Method	Prompt	# Tokens	Acc
w/o PF	Carefully interpret and manage tasks involving a dynamic table of penguins and their attributes. Focus on precisely applying variable conditions such as age and weight to generate accurate responses. Pay close attention to updated numeric values and comparison terms like “more than,” “less than,” and “equals.” Following any modifications to the table (additions or deletions of entries), reassess the data to ensure that all criteria are based on the current state of the table. Confirm each criterion by checking that the penguin’s data complies with all specifics of the query post-modification. After determining a preliminary answer, cross-verify with the available answer options to ensure alignment between your conclusion and the correct multiple-choice option (A, B, C, D, E). Employ a methodical process to guarantee that the answers reflect the most recent and accurate information, avoiding discrepancies and maintaining a high level of precision.	177	0.9873
w/ PF	Answer questions about a table of penguins and their attributes.	12	0.9873

Table 10: Comparison of prompts for the Penguins task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Analyze SVG path descriptions to ensure precise geometric shape recognition. Focus on examining the vertices formed by line segments (L) and curve segments (C), critically evaluating their connectivity, closure, and geometric precision. Disregard 'move' commands (M) since they do not contribute to the shape's boundary. The analysis should incorporate segment continuity, geometric accuracy, and how each vertex contributes to forming a complete and identifiable shape: 1. **Vertex and Segment Analysis:** - List all vertices, tracing the connection between each based on the path data. Verify closure by identifying endpoints that overlap or explicit 'Z' commands indicating that the path closes. - Analyze each segment (line or curve) connecting these vertices: measure lengths, calculate angles at each vertex, and assess whether the segments follow straight, curved, or composite trajectories. 2. **Geometric Configuration Assessment:** - Utilize mathematical methods to evaluate the geometric properties systematically. This includes calculating distances, interior angles, and identifying symmetrical relationships. Analyze side equality and angle consistency across vertices to ascertain the potential for regular or irregular polygons, ellipses, and circles. 3. **Polygon and Curve Specification:** - Classify the types of curves involved and determine their role within the shape—whether they form complete shapes such as entire circles or contribute segments to polygons. - For polygonal paths, specify the type by exact segment measurements and angle calculations, such as identifying pentagons, hexagons, or non-standard polygons, based on side length and angle congruency. 4. **Shape Identification with Geometric Justification:** - With the evidence from the rigorous analysis of vertices, angles, and segment lengths, classify each shape into concrete geometric categories. Where a shape exhibits properties matching multiple categories, list potential classifications and use weighted geometric evidence to identify the most probable category. 5. **Explicit Handling of Geometric Ambiguities:** - Address ambiguities by detailing specific geometric inconsistencies or unconventional connections. Offer clear resolutions based on geometric rules and observed anomalies, prioritizing data-driven analysis over visual estimates. Your responsibility is to perform a detailed geometric analysis of SVG shapes, ensuring adherence to stringent criteria for shape identification. Record and justify each classification using thorough understanding and quantifiable data related to the path's geometric properties. Engage in this task with precision, aiming to cover all feasible geometric forms and corresponding justifications based on the analysis.	177	0.7750
w/ PF	Determine the type of geometric shape from the SVG path data by evaluating the number of vertices, their connecting edges, angles, and any distinguishing characteristics of the shape's structure. Compare these findings directly with each option provided to ascertain the shape's identity, ensuring an accurate match between the defined path and the geometric classification.	62	0.8500

Table 11: Comparison of prompts for the Geometry task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Assess the logical relationship between the given premise and hypothesis by examining the layers of beliefs, memories, or suspicions involved. Determine whether the hypothesis can be considered a logical consequence of the premise through necessary inferences that align with typical human reasoning, even if not directly stated. Focus on whether the implied or indirect truths in the premise support the conclusion presented in the hypothesis. Choose from the options 'entailment' or 'non-entailment'.	89	0.8920
w/ PF	Clarify the logical relationship between the sentences by considering direct statements and the implications of beliefs or suspicions therein. Choose from 'entailment' if the hypothesis necessarily follows from the premise, or 'non-entailment' if it does not.	50	0.8920

Table 12: Comparison of prompts for the Epistemic task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Prompt for Categorically Enumerating and Counting Items from Mixed Groups with Flexibility in Identification: This prompt aims to guide the identification, enumeration, and counting of mixed items based on specified categories. It focuses on enhanced flexibility and comprehensive treatment of all items, recognizing the broad range of potential relevant entities. Adherence to this prompt will ensure the inclusion of all pertinent items and enhance the enumeration accuracy: 1. **Initial Listing and Identification of Items**: - Enumerate all items mentioned in the provided input without initially filtering by category. This step avoids prematurely excluding relevant entities. - Example: If the input mentions "three drums," "two flutes," "four snails," and "six birds," list them all. 2. **Flexible Categorization of Items**: - Recognize and categorize broadly. For example, if the question specifies counting "animals," include mammals, birds, reptiles, amphibians, insects, and other relevant organisms. - Clearly delineate each category's included items, grouped by their broader classifications. 3. **Calculation of Totals with Multiplicative Handling**: - Clearly apply any numeric multipliers to the items identified. For example, "four snails" should be counted as four instances toward the total. - Document both individual and cumulative counts for clarity: - Individual: Snails: 4 - Cumulative for animals (if applicable): Total animals so far including snails, birds, etc. 4. **Confirmation and Verification of Item Counts**: - Re-inspect the list to ensure all items are appropriately categorized and all mentioned quantities are accurately represented and summed. - Verify that each item and quantity matches what was specified in the input. Correct any discrepancies found during this stage. 5. **Final Answer Formulation**: - Summate the counts of requested categories (e.g., total animals or total musical instruments) and present a clear and concise final count. - Example Final Answer: "You have a total of 15 animals," ensuring that the response specifies the category and the tally. This streamlined approach ensures that nothing is overlooked, and everything is counted with appropriate categorization and precise arithmetic. By focusing on flexible categorization and comprehensive enumeration, the prompt inherently reduces errors and enhances the clarity of responses. Use this prompt to accurately manage and respond to inquiries involving the enumeration of mixed items based on specified categories.	476	0.9480
w/ PF	Questions involving the identification, categorization, and enumeration of diverse objects require a nuanced, context-aware approach that can effectively handle varying specifics and potential ambiguities of each task. Begin by precisely distinguishing among different types of objects or categories, paying special attention to instances where terms might have dual meanings or objects could fit into multiple categories. Utilize a flexible counting methodology that starts with a broad classification, honing in on a detailed count of each item based on its context-appropriate category. Implement a robust validation process that actively questions both the counts and the initial categorizations, particularly focusing on areas where there could be overlapping characteristics or shifts in context that may impact classification. Ensure that the process is adaptable, continually refining counting strategies to better handle ambiguities and optimize categorizations. This iterative and responsive validation approach should facilitate real-time corrections of any miscounts or misclassifications, thereby increasing the reliability of the output. This adaptive framework is devised to consistently deliver precise, context-sensitive results by fluidly adjusting to the nuanced demands and challenges of each scenario, ensuring that every item is accurately accounted for in a way that reflects its real-world usage and significance.	228	0.9620

Table 13: Comparison of prompts for the Object Counting task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Carefully review the timelines and events provided, focusing on identifying time slots where no overlapping activities occur. Consider all related operational constraints, such as location hours. Among the response options provided (A, B, C, D), choose the one that represents a time slot free from other scheduled events, adhering strictly to the given conditions. Clearly link your selection process to the timeline data and constraints you have analyzed. Ensure your response is concise, directly mapping your choice to the corresponding labeled option and fully substantiated by your analysis. This prompt encourages precise, data-driven decision-making with an emphasis on directly correlating findings to specific answer options.	126	0.9960
w/ PF	Streamline the event scheduling framework by executing in-depth assessments of detailed timelines, authenticated eyewitness records, and precise operational scenarios. Begin the analysis by establishing a comprehensive visual representation of all scheduled events and their associated constraints. Progress by systematically resolving any time overlaps or conflicts using rigorous scrutiny reinforced by authenticated data integration. Emphasize adaptability, encouraging periodic adjustments to the schedule in response to evolving data or alterations in circumstances. Employ a combination of logical and statistical inference methods to effectively navigate and resolve ambiguities or deviations. The primary objective is to accurately define optimal time slots for the events, with a methodology that fully incorporates all available data while also proactively filling any data voids and clarifying uncertainties. This refined approach aims to elevate the accuracy and reliability of the scheduling process, ensuring it remains adaptable to timely updates and enhancements based on real-world feedback and new information.	169	0.9860

Table 14: Comparison of prompts for the Temporal task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Systematically dissect causal relationships with a nuanced emphasis on the interplay between direct actions, systemic contexts, and the intentions—both explicit and implicit—of various agents within complex scenarios. Employ the following criteria to deepen and refine your evaluation of causation, aiming for a balanced interpretation of both individual and collective responsibilities: 1. **Refined Causation Analysis**: - **Direct, Contributory, and Conditional Effects**: Identify actions that are directly responsible for outcomes, those that contribute alongside other factors, and conditions that are either redundant or necessary. Assess these actions and conditions to ascertain their roles as either independently sufficient or as part of a collective causation. - **Influence of Systemic Contexts**: Examine whether specific outcomes could have occurred in the absence of certain pre-existing systemic conditions, thereby distinguishing the impacts of direct actions from those influenced by existing frameworks. 2. **Intent and Awareness Evaluation**: - **Explicit and Implicit Intent Recognition**: Examine actions for both the overt intentions associated with them and the implications of tacit consent to known outcomes. Consider instances where non-actions (such as maintaining a status quo) result in outcomes, analyzing whether these reflect deliberate intent or indifference. - **Systemic and Policy-Driven Intent**: Analyze how organizational policies or cultural norms may shape, suppress, or reveal the true intentions of individuals or groups, particularly noting how these policies influence actions and decisions. 3. **Dynamic Interactions and Dependencies**: - **Direct and Indirect Causal Links**: Explore how various actions intertwine and contribute collectively to outcomes, especially in scenarios involving multiple agents. Assess the extent to which individual actions depend on or reinforce one another, and how these interactions influence the overall causal chain. - **Recognition of Complex Agent Interactions**: In multiparticipant scenarios, detail the contributions and intentions of each participant, considering their cooperative or conflicting roles in shaping outcomes and their awareness of these roles. 4. **Ethical and Governance Considerations**: - **Critical Ethical Review**: Delve into the ethical ramifications of decisions and actions within their broader systemic and operational contexts, evaluating how these choices align with or deviate from ethical norms and corporate governance. - **Evaluation of Legal and Normative Frameworks**: Assess how legal regulations and organizational norms either facilitate or restrict behaviors and decisions, exploring their role in shaping the resultant causal dynamics.	481	0.6700
w/ PF	Examine causal relationships in scenarios with multiple actors and actions, emphasizing a holistic understanding of how individual and collective decisions contribute to outcomes. Instead of merely distinguishing primary from secondary causes, delve into analyzing how actions interplay and depend on each other to shape the final result. Explore the network of direct and indirect influences, acknowledging that outcomes are often the product of several intertwined conditions and decisions. Flexibly attribute causality, considering both individual actions and the synergistic effects arising from multiple factors. Conclude your analysis by discussing which combinations of actions were critical in leading to the outcome, and identify any key individual contributions that significantly altered the course of events, giving a balanced view of all interfacing elements.	137	0.6900

Table 15: Comparison of prompts for the Causal Judgement task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Given a paragraph of text, determine if it falls under ’subjective’ or ’objective’ classification. It’s important to understand that ’Subjective’ text typically expresses personal opinions, feelings, or beliefs, characterized by emotional or judgmental language. ’Objective’ text, on the other hand, presents factual information or describes events and scenarios neutrally without personal bias or emotional coloring. Note: 1. Subjective text can include personal interpretations or emotional responses, using words that evaluate or reflect personal perspective (e.g., "wonderful", "horrible", "I feel", "I believe"). 2. Objective text can include factual descriptions or general observations stated without personal involvement, even if the text uses descriptive or vivid language (e.g., "the car is red", "the meeting starts at 9 AM", "he wore a blue shirt"). Assess the usage of language in the text, considering the context and the narrative mode (first-person may hint at subjectivity, while third-person does not automatically imply objectivity). Sometimes, descriptive elements in third-person narratives are purely informative and should be classified as objective. Now, analyze the following text and classify it as either: - Objective or - Subjective Text: Is the above text subjective or objective? Select the correct option from: - Objective - Subjective	265	0.7390
w/ PF	Analyze the text provided to ascertain its primary mode of communication. Focus on whether the text is primarily descriptive of events, situations, or behaviors without incorporating personal feelings, assumptions, or speculative elements, thereby classifying it as objective. Alternatively, assess if the text predominates in expressing feelings, personal interpretations, or subjective opinions, thereby categorizing it as subjective. Utilize distinct markers such as the use of emotive language, the presence of speculative phrases, and the explicit mention of personal opinions. Importantly, differentiate between straightforward descriptions which may include character actions or stated facts in a narrative form and interpretative language that reflects personal insights or an emotional undertone. State your conclusion clearly as either ’objective’ or ’subjective’ based on your analysis.	149	0.8000

Table 16: Comparison of prompts for the Subj task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	For each question presented, you are tasked with identifying the core theme by precisely analyzing the keywords and directives provided in the language of the question. Consider both nouns and action verbs that elucidate either explicit requests like listing, defining, pinpointing locations, identifying entities, exploring human-related information, or seeking numerical data. Distinctly select the category that most closely aligns with the main inquiry of the question based on these cues. The refined emphasis on understanding the specific language cues aims to guide you more accurately in distinguishing between similar categories and avoiding misinterpretations. The categories for classification are as follows: - (A) Abbreviation: Opt for this category when the question explicitly asks for an acronym, initialism, or the abbreviated form of a term. - (B) Entity: Choose this when the primary focus is on identifying specific objects, substances, or categories of items explicitly named (including languages but excluding organizations). - (C) Description and Abstract Concept: This should be selected when the request revolves around extensive explanations, narratives, or theoretical discussions about a topic. - (D) Organization or Human: Relevant if the question is centered on companies, notable individuals, or organizations, directly discussing their actions, contributions, or creation-related queries. - (E) Location: Use this for questions that are explicitly inquiring about specific places, geographical points, or well-defined areas. - (F) Numeric Value: This applies if the question expressly asks for exact numbers, measurable data, specific dates, or other quantifiable information. When classifying, prioritize direct language cues that instruct specific actions such as "Define", "Name", "Describe", or "Quantify". Moreover, ensure clarity in differentiating between 'Entity' and 'Location' by focusing mainly on what the question is specifically asking about—whether it is a thing (B) or a place (E). Example Questions for Guidance: - "Define what PTSD stands for." - (A) Abbreviation - "List all elements found in the air." - (B) Entity - "Explain the theory behind quantum computing." - (C) Description and Abstract Concept - "Who founded Microsoft?" - (D) Organization or Human - "What city is the Louvre Museum located in?" - (E) Location - "What is the population of Tokyo?" - (F) Numeric Value Let these examples serve as your reference to meticulously categorize each question using the lead provided by keywords and immediate contexts.	490	0.8540
w/ PF	Please critically assess the main intent of each query by selecting the most fitting category from the list provided. Each category is comprehensively designed to capture a wide range of inquiries, emphasizing the need to focus on the query's principal intent or underlying reason over mere textual elements. Be aware that while some inquiries might encompass elements from multiple categories, proper discernment should guide you to the category that best encapsulates the query's overarching focus. Categories: (A) Abbreviation - Use this for queries where the main goal is to understand or decode abbreviations, initials, or acronyms. (B) Entity - Appropriate for questions that concentrate on specific tangible or intangible entities such as objects, organizations, ideas, or large concepts. (C) Description and Abstract Concept - Opt for this category when the inquiry demands an in-depth explanation or extended discourse on abstract ideas, theories, or complex concepts. (D) Human Individual - Select this for queries specifically about persons or roles defined by professional or social status, with a particular emphasis on individual attributes or notable actions. (E) Location - Best suited for queries primarily about the specifics, characteristics, or descriptions of geographical locations or important structures. (F) Numeric Value - Choose this when the query explicitly seeks numerical information, such as dates, measurements, and statistical data. Through nuanced understanding and careful analysis of the query's context and key aims, ensure that your categorization reflects the most substantial aspect of the inquiry. This precision will provide clarity and enhance the effectiveness of query processing. Example Query: "How are historical eras determined and categorized?" While this question could involve numeric data about historical timelines (F), the deeper intent is to understand the underlying principles or theories related to classifying periods in history, placing it firmly in category (C) Description and Abstract Concept. Now, assess the following query with the revised guidelines in mind: Text: "Who is considered the pioneer of modern computing and what were their key contributions?" Options: (A) Abbreviation (B) Entity (C) Description and Abstract Concept (D) Human Individual (E) Location (F) Numeric Value The guiding principle for selecting categories should always prioritize the core subject or intention of the query over surface-level keywords or secondary themes.	447	0.8600

Table 17: Comparison of prompts for the TREC task with and without Prototypical Feedback (PF).

Method	Optimized Prompt	# Tokens	Acc
w/o PF	Read carefully the following premise and hypothesis, and determine the relationship between them. Choose from ’contradiction’, ’neutral’ and ’entailment’.	31	0.8214
w/ PF	Read carefully the following premise and hypothesis, and determine the relationship between them. Choose from ’contradiction’, ’neutral’ and ’entailment’.	31	0.8214

Table 18: Comparison of prompts for the CB task with and without Prototypical Feedback (PF).