

# LLM Prompt Duel Optimizer: Efficient Label-Free Prompt Optimization

Yuanchen Wu<sup>1\*†</sup> Saurabh Verma<sup>2</sup> Justin Lee<sup>2</sup> Fangzhou Xiong<sup>2</sup>  
 Poppy Zhang<sup>2</sup> Amel Awadelkarim<sup>2</sup> Xu Chen<sup>2</sup> Yubai Yuan<sup>1</sup> Shawndra Hill<sup>2</sup>  
<sup>1</sup>Department of Statistics, The Pennsylvania State University <sup>2</sup>Meta

## Abstract

Large language models (LLMs) are highly sensitive to their input prompts, making prompt design a central challenge. While automatic prompt optimization (APO) reduces manual engineering, most approaches assume access to ground-truth references such as labeled validation data. In practice, however, collecting high-quality labels is costly and slow. We propose the **Prompt Duel Optimizer (PDO)**, a sample-efficient framework for label-free prompt optimization. PDO formulates the problem as a dueling-bandit setting, where supervision signal comes from pairwise preference feedback provided by an LLM judge. The framework combines *Double Thompson Sampling (D-TS)*, which prioritizes informative prompt comparisons, with *Top-Performer Guided Mutation*, which expands the candidate pool by mutating high-performing prompts. PDO naturally operates in label-free settings and can also incorporate partial labels to mitigate judge noise. Experiments on BIG-bench Hard (BBH) and MS MARCO show that PDO consistently outperforms baseline methods. Ablation studies further demonstrate the effectiveness of both D-TS and prompt mutation.

## 1 Introduction

LLM performance hinges on well-crafted prompts that unlock their capabilities (Sun et al., 2023). Creating effective prompts typically requires extensive trial-and-error or task-specific techniques (e.g., chain-of-thought prompting for reasoning tasks; Wei et al., 2022), which often do not transfer across tasks or domains. This limitation motivates Automatic

Prompt Optimization (APO) (Ramnath et al., 2025), which iteratively generates and evaluates candidate prompts to discover high-performing instructions.

Despite encouraging results across diverse tasks, most APO methods (Zhou et al., 2022; Yang et al., 2024; Fernando et al., 2023; Pryzant et al., 2023) rely on **reference-based supervision** (e.g., ground-truth labels, gold references, or label-derived rewards) to score candidates on validation sets. In many practical settings, however, obtaining such supervision at scale is costly and slow (Ratner et al., 2017). Still, prompt optimization remains essential; for example, in industrial LLM-based text classification, practitioners need reasonably good prompts to initiate deployment before large human-labeled datasets are available (Wagner, 2024). Such challenges raise a central question:

- *Can we optimize prompts without ground-truth label references?*

One direction is to reduce reliance on human annotation by adopting automatic evaluation of model outputs. Recent work suggests that LLMs can serve as judges of model outputs, including in reference-free settings (Zheng et al., 2023; Liu et al., 2023; Gu et al., 2024). As prompt quality is reflected in generated responses, evaluating prompts with an LLM judge is a natural approach. In this setting, rather than scoring each output independently, it is often preferable to rely on **pairwise preference feedback**: a judge compares outputs from two prompts on the same input and selects the preferred one. Pairwise comparisons typically yield a more reliable signal than direct pointwise scoring, which is prone to calibration issues (Liu et al., 2024).

LLM preference feedback introduces two challenges. *First*, LLM judges are noisy: calls

\*Work done during an internship at Meta.

†Code will be available at <https://github.com/meta-llama/prompt-ops>

are non-deterministic (He and Lab, 2025), judgments can exhibit position and verbosity biases (Zheng et al., 2023), and task complexity may amplify these effects. *Second*, preference-based evaluation scales quadratically with the number of candidates; each comparison requires an LLM API call and incurs monetary cost, making exhaustive evaluation infeasible.

To address these issues, we propose **Prompt Duel Optimizer (PDO)**, which treats preference-based prompt optimization as a *dueling bandit* problem. PDO uses *Double Thompson Sampling (D-TS)* (Wu and Liu, 2016) to adaptively choose which prompt pairs to compare. D-TS maintains posteriors over pairwise win probabilities and draws Thompson samples for both candidates, prioritizing informative comparisons. PDO also incorporates *Top-Performer Guided Mutation*, which periodically mutates top-performing prompts to generate local variants and prunes weaker ones. Aligning this expand-and-prune cycle with comparisons chosen by D-TS concentrates search on stronger regions of the prompt space without exhaustive enumeration.

We summarize our contributions as follows:

1. We formulate preference-based prompt optimization without ground-truth label references as a dueling bandit problem with LLM-judged pairwise comparisons.
2. We introduce Prompt Duel Optimizer (PDO) and evaluate it on both multiple-choice and open-ended tasks. PDO outperforms heuristic and label-free baselines, and is also competitive with supervised APO methods.
3. Through ablation studies, we show: (i) D-TS is sample-efficient, identifying high-quality prompts with far fewer comparisons than random sampling; (ii) mutation steers search toward stronger regions and yields better candidates; and (iii) pairwise preference signals are more reliable than pointwise LLM scoring.
4. PDO flexibly incorporates a small fraction of ground-truth labels when available, mitigating judge noise, accelerating convergence, and enabling practical human-in-the-loop deployment.

## 2 Preliminaries and Problem Setup

In this section, we establish the connection between preference-based prompt optimization and dueling bandits.

### 2.1 Background: Dueling Bandits

In the  $K$ -armed dueling bandit problem (Bengs et al., 2021), at each round the decision-maker selects two arms  $i$  and  $j$  to duel, and the comparison yields a stochastic preference outcome. Specifically, the probability that  $i$  is preferred to  $j$  is

$$\mu(i, j) = \Pr(i \succ j).$$

A *Condorcet winner* is an arm  $i^*$  such that  $\mu(i^*, j) > 0.5$  for all  $j \neq i^*$ . When no Condorcet winner exists, a standard choice rule is the Copeland criterion: the *Copeland score* of arm  $i$  counts the number of opponents it beats with probability greater than  $1/2$ :

$$C(i) = |\{j \neq i : \mu(i, j) > \frac{1}{2}\}|.$$

A *Copeland winner* is then an arm  $i^* \in \arg \max_i C(i)$ .

### 2.2 Prompt Optimization Through the Lens of Dueling Bandits

Building on these definitions, we formalize preference-based prompt optimization in the dueling bandits setting.

**Problem Setup.** Let  $\mathcal{X}$  denote the input space and  $\mathcal{P} = \{p_1, \dots, p_K\}$  a finite set of candidate prompts. Let  $D_{\mathcal{X}}$  be a distribution over  $\mathcal{X}$ . For  $p \in \mathcal{P}$  and  $x \sim D_{\mathcal{X}}$ , let  $f_p(x)$  denote the LLM output when applying prompt  $p$  to input  $x$ . To compare the two prompts  $p_i, p_j \in \mathcal{P}$  on the same input  $x$ , we query an LLM judge and record a binary preference:

$$\text{Judge}_x(p_i, p_j) = \begin{cases} 1, & \text{if } f_{p_i}(x) \succ f_{p_j}(x), \\ 0, & \text{otherwise.} \end{cases}$$

Given a set of unlabeled examples  $\{x_i\}_{i=1}^n$ , the empirical estimate of  $\mu(p_i, p_j)$  becomes

$$\hat{\mu}(p_i, p_j) = \frac{1}{n} \sum_{k=1}^n \mathbf{1}[\text{Judge}_{x_k}(p_i, p_j) = 1].$$

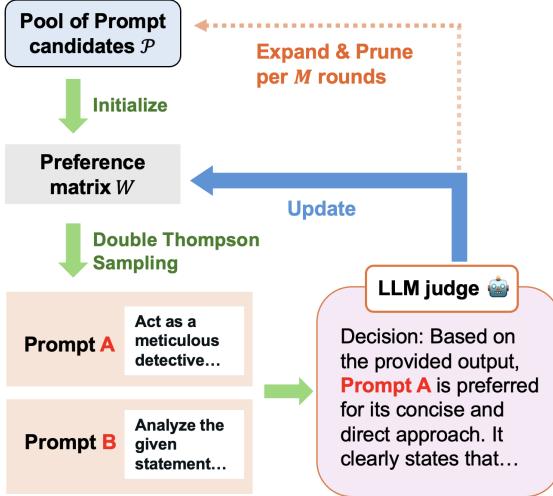


Figure 1: Workflow of the Prompt Duel Optimizer.

**Prompt Optimization Objective.** The goal of prompt optimization is to identify a prompt that maximizes task performance. In the absence of ground-truth references, we use pairwise preferences as a practical proxy for selecting high-quality prompts. Using empirical estimates  $\hat{\mu}(p_i, p_j)$ , we select the Condorcet winner when it exists, or otherwise the Copeland winner. In practice, limited API budgets make exhaustive estimation ( $O(n|\mathcal{P}|^2)$  comparisons) infeasible, motivating sample-efficient methods that adaptively target informative pairs while aligning with the Condorcet/Copeland objectives.

### 3 Prompt Duel Optimizer (PDO)

To address the above challenge, we propose the **Prompt Duel Optimizer (PDO)**, an algorithm designed to identify high-performing prompts under limited comparison budgets. PDO combines two complementary components: (1) *Double Thompson Sampling (D-TS)* (Wu and Liu, 2016), a Bayesian strategy for efficient pairwise evaluation that targets Copeland-optimal winners; and (2) *Top-Performer Guided Mutation*, which adaptively mutates top-performing prompts to expand coverage of the search space.

#### 3.1 Efficient Prompt Selection via Double Thompson Sampling

D-TS extends Thompson Sampling (Agrawal and Goyal, 2012) to the dueling bandit setting, where feedback comes from pairwise com-

parisons rather than scalar rewards. Each round uses two independent Thompson draws to concentrate queries on informative duels and guides the search toward a *Copeland-optimal* prompt.

**Notations.** For each pair  $(p_i, p_j)$ , let  $W_{ij}$  denote the current number of wins of  $p_i$  over  $p_j$ , and let  $N_{ij} = W_{ij} + W_{ji}$  be the total number of duels. The Bayesian posterior for the probability that  $p_i$  beats  $p_j$  is modeled as  $\theta_{ij} \sim \text{Beta}(W_{ij} + 1, W_{ji} + 1)$ . For  $\alpha > 0$ , the corresponding upper and lower confidence bounds are defined as

$$u_{ij}(t) = \frac{W_{ij}}{N_{ij}} + \sqrt{\frac{\alpha \log t}{\max\{1, N_{ij}\}}},$$

$$l_{ij}(t) = \frac{W_{ij}}{N_{ij}} - \sqrt{\frac{\alpha \log t}{\max\{1, N_{ij}\}}}.$$

**Per-round Procedure** ( $t = 1, 2, \dots, T$ ):

1. *First Prompt Selection.* Compute an optimistic Copeland score for each prompt:

$$\widehat{\zeta}_i(t) = \frac{1}{K-1} \sum_{j \neq i} \mathbf{1}\{u_{ij}(t) \geq 0.5\}$$

Keep only prompts with the maximum score in a set  $\zeta(t) = \{i \mid \widehat{\zeta}_i(t) = \max_k \widehat{\zeta}_k(t)\}$ . For each  $i \in \zeta(t)$ , draw independent samples  $\theta_{ij}^{(1)}$  and count

$$s_i = \sum_{j \neq i} \mathbf{1}\{\theta_{ij}^{(1)} \geq 0.5\}$$

Select prompt  $i^* = \arg \max_{i \in \zeta(t)} s_i$ .

2. *Second Prompt Selection.* Restrict to only uncertain opponents for  $i^*$ ,

$$S_{i^*}(t) = \{j \neq i^* : l_{i^*j}(t) \leq 0.5\}$$

Draw independent samples  $\theta_{ji^*}^{(2)}$  and select  $j^* = \arg \max_{j \in S_{i^*}(t)} \theta_{ji^*}^{(2)}$

3. *Duel and update.* Judge prompts  $(p_{i^*}, p_{j^*})$ , record the winner, and update  $W_{i^*j^*}$ .

This two-step prompt selection procedure using D-TS achieves the following regret bounds in general Copeland settings.

#### Theorem 1 (PDO Copeland Regret)

Let  $\mathcal{P} = \{p_1, \dots, p_K\}$  be  $K$  prompts with pairwise preference probabilities  $\mu_{i,j} = \Pr(p_i \succ p_j)$ . Define the normalized Copeland score

$$\zeta_i = \frac{1}{K-1} \sum_{j \neq i} \mathbf{1}\{\mu_{i,j} > 1/2\}, \quad \zeta^* = \max_i \zeta_i.$$

If at round  $t$  the algorithm selects  $(a_t, b_t)$ , the regret is  $r_t = \zeta^* - \max\{\zeta_{a_t}, \zeta_{b_t}\}$  and  $R_T = \sum_{t=1}^T r_t$ . Then Double Thompson Sampling (D-TS) achieves (Wu and Liu, 2016)

$$\mathbb{E}[R_T] = O(K^2 \log T).$$

**Implications.** The theorem implies that  $\frac{\mathbb{E}[R_T]}{T} \rightarrow 0$  as  $T \rightarrow \infty$ , i.e., PDO with D-TS asymptotically converges to selecting Copeland-optimal prompts. If a Condorcet winner exists, D-TS converges to that unique prompt; otherwise, it converges to the set of Copeland winners.

### 3.2 Efficient Prompt Discovery via Top-Performer Guided Mutation

While D-TS identifies the Copeland-optimal prompt within a fixed pool, the broader goal of PDO is to locate the global optimum  $p^*$  over a combinatorially large prompt space. Since exhaustive search is infeasible, PDO incrementally expands the candidate pool through mutation of top-performing prompts.

**Mutation Procedure.** At round  $t$ , let the current pool be  $\mathcal{P}_t = \{p_1, \dots, p_{K_t}\}$  with the empirical Copeland scores  $\widehat{\mathbf{C}}_t := (\widehat{C}_t(p))_{p \in \mathcal{P}_t}$ . The procedure is:

1. *Selection:* Choose the top-performing prompt  $p_t^* = \arg \max_{p \in \mathcal{P}_t} \widehat{\mathbf{C}}_t$ .
2. *Mutation:* Generate a variant  $p_{\text{new}}$  of  $p_t^*$  via template edits, text-gradient guided changes, or LLM-assisted rewrites.
3. *Expansion:* Update pool  $\mathcal{P}_{t+1} = \mathcal{P}_t \cup \{p_{\text{new}}\}$ .

**Coverage-Theoretic Justification.** The efficiency of mutating top performers can be understood by assuming an underlying performance function  $C$  (e.g., expected Copeland score) that is  $L$ -Lipschitz under a prompt distance  $d(\cdot, \cdot)$  measuring semantic or functional similarity between prompts. The  $\epsilon$ -neighborhood of the optimum is  $B_\epsilon(p^*) = \{p \in \mathcal{P} : d(p, p^*) \leq \epsilon\}$ .

Drawing on classical results in Lipschitz bandits (Kleinberg et al., 2008), concentrating mutations around the current best prompt is analogous to “zooming in” on near-optimal regions. Mutating a weak prompt  $p_{\text{weak}}$  produces only local variants; unless  $p_{\text{weak}}$  lies close to  $p^*$ , reaching  $B_\epsilon(p^*)$  requires many steps, with

---

### Algorithm 1: Prompt Duel Optimizer

---

```

Require:  $\mathcal{P}_0$  (initial pool); judge  $\mathcal{J}$ ;
           batch size  $m$ ; total rounds  $T$ ;
           mutation period  $M$ 
1  $\mathcal{P} \leftarrow \mathcal{P}_0$ ; initialize  $W, N \in \mathbb{R}^{|\mathcal{P}| \times |\mathcal{P}|}$  to
   zeros;
2 for  $t \leftarrow 1$  to  $T$  do
3    $(p_i, p_j) \leftarrow \text{D-TS}(\mathcal{P}, W, N)$  ;
    // prompt selection via D-TS
4    $(w_i, w_j) \leftarrow \text{DUEL}(p_i, p_j, m, \mathcal{J})$  ;
    // get win counts
5    $W[i, j] \leftarrow W[i, j] + w_i$ ;
6    $W[j, i] \leftarrow W[j, i] + w_j$ ;
7    $N[i, j] \leftarrow N[i, j] + m$ ;
8    $N[j, i] \leftarrow N[j, i] + m$ ;
9   if  $t \bmod M = 0$  then
10     $p_t^* \leftarrow \text{CURRENTBEST}(\mathcal{P}, W, N)$  ;
     // Copeland rank
11     $p_{\text{new}} \leftarrow \text{MUTATE}(p_t^*)$  ;
     // generate new prompts
     // via top-performer guided
     // mutation
12     $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_{\text{new}}\}$ ;
13    Expand  $W, N$  with zero
         row/column for  $p_{\text{new}}$ ;
14 return  $\text{CURRENTBEST}(\mathcal{P}, W, N)$  ;
     // final Copeland winner

```

---

expected hitting time increasing in  $d(p_{\text{weak}}, p^*)$ . By contrast, mutating the best-known prompt  $p_{\text{top}}$  yields neighbors that satisfy

$$C(p) \geq C(p_{\text{top}}) - L \cdot d(p, p_{\text{top}}),$$

so they inherit relatively strong performance. This biases exploration toward high-performing regions and shortens convergence.

PDO thus combines the regret guarantees of D-TS with the coverage-theoretic efficiency of top-performer mutation, making it a sample-efficient approach for prompt optimization without ground-truth references (see Algorithm 1). In practice, we periodically prune the candidate set by removing prompts with low Copeland scores at the current iteration to accelerate convergence.

## 4 Experiment and Results

### 4.1 Experimental Setup

**Datasets.** In this section, we conduct experiments to evaluate PDO on both closed-

Method	Causal	Date	DisambigQA	Formal	Geometric	Hyperbaton	Logical-5	Logical-7
No prompt	<b>0.661</b> $\pm 0.044$	<b>0.854</b> $\pm 0.024$	<b>0.698</b> $\pm 0.047$	<b>0.739</b> $\pm 0.031$	<b>0.434</b> $\pm 0.036$	<b>0.900</b> $\pm 0.020$	<b>0.785</b> $\pm 0.018$	<b>0.739</b> $\pm 0.033$
CoT	<b>0.653</b> $\pm 0.042$	<b>0.877</b> $\pm 0.019$	<b>0.720</b> $\pm 0.039$	<b>0.725</b> $\pm 0.027$	<b>0.422</b> $\pm 0.028$	<b>0.891</b> $\pm 0.023$	<b>0.761</b> $\pm 0.027$	<b>0.726</b> $\pm 0.025$
PoS	<b>0.652</b> $\pm 0.037$	<b>0.878</b> $\pm 0.019$	<b>0.698</b> $\pm 0.043$	<b>0.739</b> $\pm 0.027$	<b>0.403</b> $\pm 0.030$	<b>0.896</b> $\pm 0.024$	<b>0.798</b> $\pm 0.032$	<b>0.750</b> $\pm 0.026$
SPO	<b>0.655</b> $\pm 0.033$	<b>0.884</b> $\pm 0.017$	<b>0.725</b> $\pm 0.057$	<b>0.738</b> $\pm 0.018$	<b>0.650</b> $\pm 0.069$	<b>0.886</b> $\pm 0.031$	<b>0.787</b> $\pm 0.031$	<b>0.721</b> $\pm 0.026$
PDO (ours)	<b>0.681</b> $\pm 0.040$	<b>0.918</b> $\pm 0.014$	<b>0.738</b> $\pm 0.050$	<b>0.744</b> $\pm 0.030$	<b>0.598</b> $\pm 0.073$	<b>0.910</b> $\pm 0.029$	<b>0.804</b> $\pm 0.034$	<b>0.711</b> $\pm 0.019$
Method	Navigate	Penguins	Salient	Snarks	Tracking-5	Tracking-7	Tracking-3	Web of Lies
No prompt	<b>0.869</b> $\pm 0.013$	<b>0.915</b> $\pm 0.018$	<b>0.698</b> $\pm 0.020$	<b>0.823</b> $\pm 0.027$	<b>0.695</b> $\pm 0.033$	<b>0.499</b> $\pm 0.020$	<b>0.890</b> $\pm 0.019$	<b>0.766</b> $\pm 0.020$
CoT	<b>0.878</b> $\pm 0.016$	<b>0.915</b> $\pm 0.023$	<b>0.709</b> $\pm 0.021$	<b>0.833</b> $\pm 0.023$	<b>0.724</b> $\pm 0.046$	<b>0.532</b> $\pm 0.025$	<b>0.904</b> $\pm 0.019$	<b>0.796</b> $\pm 0.022$
PoS	<b>0.866</b> $\pm 0.019$	<b>0.910</b> $\pm 0.027$	<b>0.693</b> $\pm 0.025$	<b>0.816</b> $\pm 0.026$	<b>0.725</b> $\pm 0.030$	<b>0.538</b> $\pm 0.034$	<b>0.888</b> $\pm 0.019$	<b>0.861</b> $\pm 0.019$
SPO	<b>0.874</b> $\pm 0.035$	<b>0.934</b> $\pm 0.025$	<b>0.662</b> $\pm 0.038$	<b>0.820</b> $\pm 0.046$	<b>0.692</b> $\pm 0.046$	<b>0.543</b> $\pm 0.026$	<b>0.826</b> $\pm 0.087$	<b>0.818</b> $\pm 0.043$
PDO (ours)	<b>0.900</b> $\pm 0.023$	<b>0.937</b> $\pm 0.034$	<b>0.681</b> $\pm 0.032$	<b>0.840</b> $\pm 0.039$	<b>0.796</b> $\pm 0.084$	<b>0.641</b> $\pm 0.089$	<b>0.930</b> $\pm 0.046$	<b>0.942</b> $\pm 0.040$

Table 1: Test results on 16 BBH tasks averaged over 10 runs. For PDO, we report the test accuracy of the prompt selected by the highest Copeland score. PDO is compared with baselines that assume **no access** to ground-truth labels. The best performance is shown in bold, and the second-best is underlined.

Method	Causal	Date	DisambigQA	Formal	Geometric	Hyperbaton	Logical-5	Logical-7
APE	<b>0.680</b> $\pm 0.044$	<b>0.892</b> $\pm 0.019$	<b>0.730</b> $\pm 0.043$	<b>0.747</b> $\pm 0.022$	<b>0.670</b> $\pm 0.072$	<b>0.940</b> $\pm 0.013$	<b>0.822</b> $\pm 0.041$	<b>0.721</b> $\pm 0.035$
OPRO	<b>0.682</b> $\pm 0.044$	<b>0.910</b> $\pm 0.022$	<b>0.734</b> $\pm 0.038$	<b>0.728</b> $\pm 0.026$	<b>0.569</b> $\pm 0.048$	<b>0.932</b> $\pm 0.021$	<b>0.774</b> $\pm 0.030$	<b>0.718</b> $\pm 0.031$
Breeder	<b>0.683</b> $\pm 0.027$	<b>0.898</b> $\pm 0.026$	<b>0.745</b> $\pm 0.038$	<b>0.746</b> $\pm 0.027$	<b>0.684</b> $\pm 0.048$	<b>0.932</b> $\pm 0.021$	<b>0.778</b> $\pm 0.022$	<b>0.724</b> $\pm 0.018$
PDO (ours)	<b>0.680</b> $\pm 0.033$	<b>0.915</b> $\pm 0.024$	<b>0.740</b> $\pm 0.040$	<b>0.754</b> $\pm 0.022$	<b>0.712</b> $\pm 0.058$	<b>0.948</b> $\pm 0.017$	<b>0.809</b> $\pm 0.028$	<b>0.733</b> $\pm 0.024$
Method	Navigate	Penguins	Salient	Snarks	Tracking-5	Tracking-7	Tracking-3	Web of Lies
APE	<b>0.899</b> $\pm 0.024$	<b>0.925</b> $\pm 0.025$	<b>0.686</b> $\pm 0.033$	<b>0.875</b> $\pm 0.033$	<b>0.803</b> $\pm 0.041$	<b>0.600</b> $\pm 0.050$	<b>0.911</b> $\pm 0.041$	<b>0.948</b> $\pm 0.020$
OPRO	<b>0.882</b> $\pm 0.032$	<b>0.919</b> $\pm 0.025$	<b>0.687</b> $\pm 0.023$	<b>0.888</b> $\pm 0.021$	<b>0.833</b> $\pm 0.083$	<b>0.662</b> $\pm 0.073$	<b>0.924</b> $\pm 0.035$	<b>0.938</b> $\pm 0.045$
Breeder	<b>0.908</b> $\pm 0.016$	<b>0.927</b> $\pm 0.018$	<b>0.698</b> $\pm 0.024$	<b>0.903</b> $\pm 0.019$	<b>0.859</b> $\pm 0.049$	<b>0.600</b> $\pm 0.042$	<b>0.927</b> $\pm 0.028$	<b>0.963</b> $\pm 0.019$
PDO (ours)	<b>0.890</b> $\pm 0.010$	<b>0.933</b> $\pm 0.027$	<b>0.695</b> $\pm 0.022$	<b>0.874</b> $\pm 0.036$	<b>0.847</b> $\pm 0.062$	<b>0.662</b> $\pm 0.084$	<b>0.946</b> $\pm 0.015$	<b>0.979</b> $\pm 0.024$

Table 2: Test results of PDO when selecting the prompt with the highest development set accuracy (from the same candidate pools as in Table 1). The results are compared with supervised APO methods.

ended multiple-choice tasks and open-ended QA tasks. For the multiple-choice setting, we select 16 tasks from Big-Bench-Hard (Suzgun et al., 2022), using accuracy as the evaluation metric. For the open-ended QA setting, we consider four task categories from MS-MARCO (Bajaj et al., 2016), where the final evaluation metric is an integer score between 1 and 5, assigned by an LLM judge comparing the model’s output with the ground-truth answer provided by the original dataset.

Across both task types, for the results reported in this section, we randomly split the data into development and test sets with a 50/50 split ratio. We report the test set performance averaged over 10 runs. The *Llama-3.3-70B-Instruct* model is used for prompt generation, preference judging, and final evaluation. Detailed descriptions of the datasets and the experimental setup of PDO are provided in Appendix C.

**LLM Judge Design.** For multiple-choice tasks, we require the LLM to produce both an answer and the accompanying reasoning given the instruction prompt. We then apply a dual-judge approach: if two prompts yield different answers, the LLM judge selects the

prompt with the correct answer; if they yield the same answer, the judge decides based on the quality of reasoning. For open-ended tasks, the LLM judge design is more straightforward: each pair of responses is evaluated on accuracy, completeness, relevance, and clarity, consistent with the criteria of the final evaluation metric. To mitigate position bias, the order of the two outputs in each prompt pair is randomized before being fed into the judge template. A detailed rationale and analysis of the LLM preference judge design are provided in Section 5 and Appendix A.

**Baselines.** PDO is designed for prompt optimization without access to ground-truth labels or external references. A directly comparable baseline is **SPO** (Xiang et al., 2025), which similarly uses an LLM judge to iteratively compare the outputs of two prompts and select the winning prompt. We also include classical prompting techniques including chain-of-thought **COT** (Wei et al., 2022) and plan-and-solve **PoS** (Wang et al., 2023).

For the Big-Bench-Hard (BBH) dataset, we also evaluate PDO under a supervised setting to enable comparison with popular supervised APO methods, including **APE** (Zhou

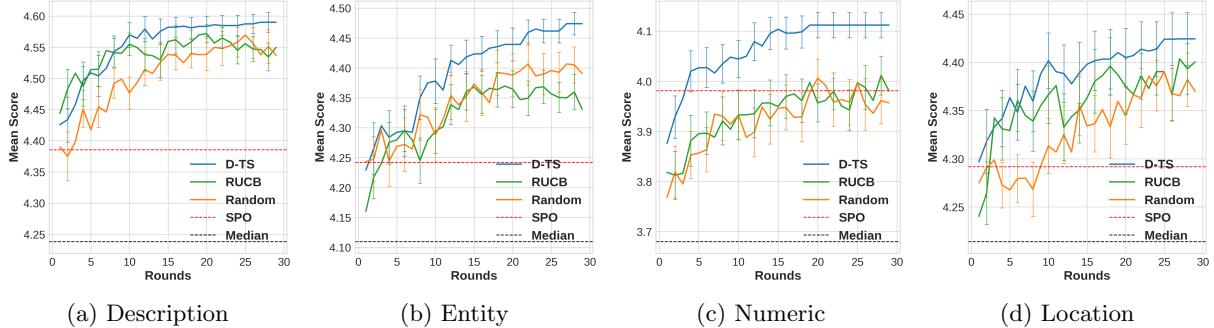


Figure 2: Test performance of the winning prompt on the four MS-MARCO tasks. Each curve shows the mean score of the current Copeland leader over rounds, with 50 duels per round. PDO with D-TS consistently outperforms RUCB and Random sampling, and surpasses the SPO baseline within a few rounds. Gray lines indicate the median test score across all prompts generated by PDO.

et al., 2022), **OPRO** (Yang et al., 2024), and **Breeder** (Fernando et al., 2023). The optimization itself remains label-free; labels are introduced only in the final stage to select the prompt that achieves the highest development-set accuracy among candidates generated through initial proposals and prompt mutations. Results for the without-label and with-label settings are reported in Tables 1 and 2, respectively.

## 4.2 Benchmark Results

**Multiple-choice Tasks Performance.** As shown in Table 1, using only preference signals from the LLM judge and selecting the winner prompt by Copeland scores, PDO achieves the highest evaluation accuracy on **13/16** tasks. Notable gains include *Tracking-7* (0.641 vs. 0.543, **+9.8pp**) and *Web of Lies* (0.942 vs. 0.861, **+8.1pp**). We examine the benefit of top-performer guided mutation in Appendix B.

In the labeled setting, when the final prompt is chosen by development-set accuracy, PDO remains competitive with state-of-the-art prompt optimization baselines, achieving the top score on **9/16** tasks. For the *Geometric* dataset, selecting the prompt based on development-set accuracy yields 0.712, outperforming all other methods, whereas selecting based on the LLM judge’s ranking results is only 0.598 accuracy, underperforming the label-free baseline SPO. In this particular case, this gap suggests that the LLM judge is less effective and noisier in identifying the best prompt on this dataset compared with others in BBH. We further examine the issue of LLM judge noise in Section 5.

**Open-ended QA Tasks.** We evaluate four MS-MARCO tasks (*Description*, *Entity*, *Numeric*, and *Location*), starting from a pool of  $|\mathcal{P}| = 50$  instructions. At each round  $t$ , we snapshot the win matrix  $W_t$ , compute Copeland scores, and report the test performance of the current Copeland winner. Figure 2 reports averages over 30 independent runs, comparing D-TS with the dueling-bandit alternative Relative Upper Confidence Bound (RUCB; Zoghi et al. 2013), uniform Random sampling, and the SPO baseline. Across all tasks, D-TS achieves the highest scores and converges more quickly than RUCB and Random. The horizontal reference lines mark the median score of all prompts generated by PDO (gray) and the SPO baseline (red). Within just a few rounds, D-TS surpasses both reference lines and maintains this advantage over RUCB and Random. These results highlight the *sample efficiency* of D-TS: it consistently identifies stronger prompts faster and more reliably than random sampling or alternative dueling-bandit methods.

## 5 LLM Preference Judge Analysis

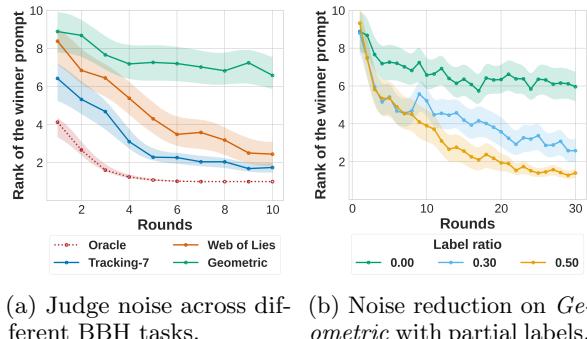
A fundamental bottleneck in PDO is that it depends on an LLM judge to approximate downstream performance through pairwise comparisons. In this section, we examine the factors that influence the effectiveness of the LLM judge in PDO.

**Correlation Between Judge Noise and Task Performance.** We proxy LLM-judge effectiveness for the 16 BBH tasks by the test accuracy gap between Table 1 (prompt cho-

sen by Copeland score, label-free) and Table 2 (prompt chosen by development-set accuracy). In our results, *Tracking-7* and *Web of Lies* show small gaps and achieve top performance under both selection criteria, whereas *Geometric* exhibits a large discrepancy of **11.4** percentage points.

To investigate further, for each task we fix  $|\mathcal{P}| = 20$  instruction prompts with varying accuracy, run D-TS with the LLM judge, and compare it against an *oracle* judge that always selects the higher-accuracy prompt in each duel. At round  $t$ , we record the ground-truth accuracy rank of the current Copeland leader. Figure 3a shows that the oracle converges to the best prompt by round 4; with the LLM judge, *Tracking-7* steadily improves to rank 2 and *Web of Lies* approaches rank  $\approx 2.5$ , while *Geometric* remains around ranks 6–8 across rounds. These trends confirm that judge reliability is closely related to the performance gaps observed in Tables 1 and 2.

**Reducing Judge Noise.** We hypothesize that a key source of judge noise in BBH tasks is the inability to select the correct prompt when two prompts yield different answers. As an extension to the original label-free setting, we test whether providing a small fraction of labeled examples can help identify the top-ranked prompt.



(a) Judge noise across different BBH tasks. (b) Noise reduction on *Geometric* with partial labels.

Figure 3: The accuracy of the LLM judge varies across tasks, with some tasks (e.g., *Geometric*) showing persistent misjudgments compared to an oracle judge. Introducing a fraction of true labels ( $r \in \{0.3, 0.5\}$ ) helps reduce the impact of noisy judgments and guide D-TS more quickly toward the top-ranked prompts.

Specifically, we assume that the ground-truth labels of an  $r \in \{0.3, 0.5\}$  fraction of examples are known in advance. On each round, if the

chosen example  $x_t$  belongs to this labeled subset, we determine the duel outcome using its true label: if prompt  $i$  is correct and  $j$  is incorrect, we update  $W[i, j] \leftarrow W[i, j] + 1$ , and symmetrically for the reverse case. If both are correct or both are wrong, we add 0.5 to each of  $W[i, j]$  and  $W[j, i]$ . When  $x_t$  is unlabeled, the LLM judge decides the duel as usual. Figure 3b shows that on *Geometric*,  $r = 0.3$  pulls the leader to roughly the top 3 by later rounds, while  $r = 0.5$  drives it close to ranks 1–2 with narrower ribbons. Overall, these results confirm our hypothesis: access to partial ground-truth labels mitigates systematic judge errors and steers D-TS toward the best prompt.

**Impact of Judge Model Size.** We evaluate MS-MARCO performance when using Llama3.1-8B as the preference judge compared to the larger Llama3.3-70B (default). Table 3 reports the test accuracy of the prompt with the highest Copeland score at round 30. The comparison shows mixed but stable results: 70B slightly outperforms 8B on *Entity* and *Numeric*, while 8B surpasses 70B on *Description* and *Location*. Overall, both judges perform similarly and consistently outperform the baseline SPO.

Method	Description	Entity	Numeric	Location
SPO	$4.385 \pm 0.16$	$4.242 \pm 0.24$	$3.981 \pm 0.25$	$4.292 \pm 0.23$
PDO (8B)	<b><math>4.602 \pm 0.09</math></b>	$4.473 \pm 0.13$	$4.076 \pm 0.20$	<b><math>4.451 \pm 0.11</math></b>
PDO (70B)	$4.590 \pm 0.08$	<b><math>4.477 \pm 0.10</math></b>	<b><math>4.112 \pm 0.13</math></b>	$4.432 \pm 0.12$

Table 3: Results on MS-MARCO using Llama-70B vs. 8B as the preference judge.

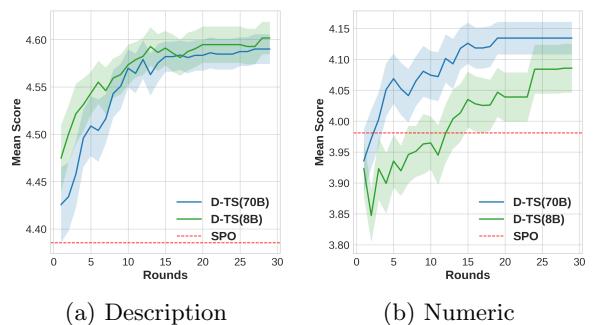


Figure 4: On the *Numeric* task, Llama-70B as judge enables D-TS to find strong prompts faster and reach higher scores, whereas on the *Description* task, Llama-8B slightly outperforms 70B overall.

**Pairwise Preference vs. Pointwise Scoring.** PDO relies on the *pairwise preferences* of an LLM judge to select the winning prompt. To highlight the benefit of this formulation, we compare it against a *pointwise scoring* judge, where each prompt is evaluated independently on the development set by assigning a numeric score (1–5) without access to ground-truth answers. The prompt with the highest average score is then selected. We evaluate both strategies on the same fixed pool of 50 candidates from MS-MARCO in 4.2 and report the test mean score of the selected prompt. Figure 5 shows that preference judgement consistently outperforms pointwise scoring in 7 of 8 model–task combinations across two judge models. Overall, these results are consistent with the LLM-as-judge literature: pairwise comparisons yield more stable decisions than direct pointwise scoring by reducing dependence on calibration (Liu et al., 2024).

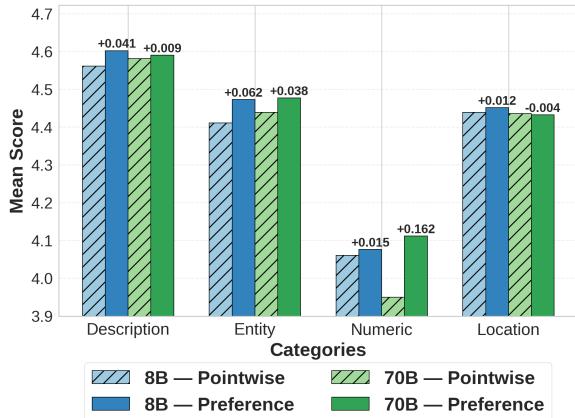


Figure 5: Comparison of pairwise preference used in PDO and pointwise scoring for prompt selection on MS-MARCO. Results indicate that pairwise preference leads to higher-performing prompts across model sizes (7 out of 8 cases).

## 6 Related Work

**APO Methods: from Supervised to Label-free.** Traditional prompt optimization methods rely heavily on supervised signals from labeled validation sets, where the typical pipeline involves generating, mutating, and scoring prompts (Zhou et al., 2022; Fernando et al., 2023; Pryzant et al., 2023; Guo et al., 2023; Schulhoff et al., 2024). More recently, studies have reduced supervision requirements (Xiang et al., 2025; Chen et al., 2024; Madaan

et al., 2023; Cheng et al., 2023; Dong et al., 2022). Notably, Self-Supervised Prompt Optimization (SPO) (Xiang et al., 2025) eliminates external references by using output-vs-output comparisons, iteratively generating, executing, and comparing prompts through pairwise LLM judgments. However, SPO follows a *greedy hill-climbing loop* without a principled exploration-exploitation strategy, limiting its ability to allocate comparisons efficiently and to robustly identify high-performing prompts.

**Bandit-Based Prompt Optimization.** The connection between prompt optimization and multi-armed bandits (MAB) has recently gained attention, as prompts can naturally be viewed as arms. OPTS (Ashizawa et al., 2025) formulates prompt strategy selection as a bandit problem with Thompson sampling. TRIPLE (Shi et al., 2024) casts prompt optimization as fixed-budget best-arm identification, assuming a predefined set of prompts with known scores. However, both approaches require labeled validation sets for scoring. APOHF (Lin et al., 2025) connects preference-based prompt optimization to dueling bandits, but assumes human-annotated pairwise preferences that are impractical at scale. PDO retains the dueling-bandit formulation while replacing human preferences with LLM-judge comparisons to align with realistic prompt optimization settings.

## 7 Conclusion

In this paper, we introduce PDO, a prompt optimization method that reduces dependence on costly supervision by formulating label-free optimization as a dueling bandit problem guided by LLM preference feedback. By combining Double Thompson Sampling (D-TS) with top-performer-guided mutation, PDO adaptively searches for informative prompt comparisons to identify high-performing prompts. Experiments on multiple-choice BBH tasks and the open-ended MS-MARCO dataset show that PDO consistently achieves competitive performance against baseline methods across tasks. We further demonstrate that LLM judges provide reliable preference feedback overall, while also analyzing the impact of judge noise and the role of partial labels in reducing it during optimization.

## Limitation

While our dueling-bandit framework provides a practical way to optimize prompts without ground-truth labels, it also has natural limitations. The quality of preference feedback depends on the meta prompt used to guide the LLM judge, and subtle differences in wording or criteria can influence its decisions. The judge’s own capability also matters: stronger foundation models tend to produce more reliable preferences, whereas smaller or less specialized models may introduce more noise, particularly in domain-specific tasks. Since the algorithm optimizes for the judge’s notion of quality rather than the true task metric, there is a risk of favoring stylistic patterns that the judge prefers. Although we include a thorough discussion and ablation studies of LLM judge design in Section 5 and Appendix A, due to computational resource constraints we were unable to conduct broader experiments across additional models. Despite these challenges, our formulation offers a principled and scalable starting point for label-free prompt optimization using preference feedback, and we hope it inspires future work on improving the alignment of LLM judge with task objectives.

## References

- Shipra Agrawal and Navin Goyal. 2012. [Analysis of thompson sampling for the multi-armed bandit problem](#). In *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 39.1–39.26, Edinburgh, Scotland. PMLR.
- Rin Ashizawa, Yoichi Hirose, Nozomu Yoshinari, Kento Uchida, and Shinichi Shirakawa. 2025. [Bandit-based prompt design strategy selection improves prompt optimizers](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20799–20817, Vienna, Austria. Association for Computational Linguistics.
- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. [Ms marco: A human generated machine reading comprehension dataset](#). *arXiv preprint arXiv:1611.09268*.
- Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. 2021. [Preference-based online learning with dueling bandits: A survey](#). *Journal of Machine Learning Research*, 22(7):1–108. Open access.
- Yongchao Chen, Jacob Arkin, Yilun Hao, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2024. [PRompt optimization in multi-step tasks \(PROMST\): Integrating human feedback and heuristic-based sampling](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3859–3920, Miami, Florida, USA. Association for Computational Linguistics.
- Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. [Black-box prompt optimization: Aligning large language models without model training](#). *ArXiv*, abs/2311.04155.
- Yihong Dong, Kangcheng Luo, Xue Jiang, Zhi Jin, and Ge Li. 2022. [Pace: Improving prompt with actor-critic editing for large language model](#). In *arXiv preprint arXiv:2212.XXXX*.
- Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. [Promptbreeder: Self-referential self-improvement via prompt evolution](#). *arXiv preprint arXiv:2309.16797*.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel M. Ni, and Jian Guo. 2024. [A survey on llm-as-a-judge](#). *arXiv preprint arXiv:2411.15594*.
- Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. [Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers](#). *arXiv preprint arXiv:2309.08532*.
- Horace He and Thinking Machines Lab. 2025. [Defeating nondeterminism in llm inference](#). *Thinking Machines Lab: Connectionism*. [Https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/](https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/).
- Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. 2008. [Multi-armed bandits in metric spaces](#). In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 681–690. ACM.
- Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. 2025. [Prompt optimization with human feedback](#).
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. [G-eval: NLG evaluation using gpt-4 with better human alignment](#). In *Proceedings of the 2023 Conference*

*on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2024. Aligning with human judgement: The role of pairwise preference in large language model evaluators. In *Proceedings of the First Conference on Language Modeling (COLM 2024)*, page –.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651.

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*, pages 9340–9366, Miami, Florida, USA. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with “gradient descent” and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968. 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, Balasubramanian Srinivasan, Qiaojing Yan, Yueyan Chen, Haibo Ding, and 2 others. 2025. A systematic survey of automatic prompt optimization techniques. *arXiv preprint arXiv:2502.16923v2*.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Samuel Schulhoff and 1 others. 2024. A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*.

Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. 2024. Efficient prompt optimization through the lens of best arm identification. In *Neural Information Processing Systems*.

Jiuding Sun, Chantal Shaib, and Byron C. Wallace. 2023. Evaluating the zero-shot robustness

of instruction-tuned language models. *arXiv preprint arXiv:2306.11270*.

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. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Patrick Wagner. 2024. Using llms for text classification. *Medium*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

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 *NeurIPS*.

Huasen Wu and Xin Liu. 2016. Double thompson sampling for dueling bandits. *Advances in neural information processing systems*, 29.

Jinyu Xiang, Jiayi Zhang, Zhaoyang Yu, Xining Liang, Fengwei Teng, Jinhao Tu, Fashen Ren, Xiangru Tang, Sirui Hong, Chenglin Wu, and Yuyu Luo. 2025. Self-supervised prompt optimization. *arXiv preprint arXiv:2502.06855*. Version v3, revised 21 August 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *ICLR*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

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*.

Masrour Zoghi, Shimon Whiteson, Rémi Munos, and Maarten de Rijke. 2013. Relative upper confidence bound for the k-armed dueling bandit problem. *arXiv preprint arXiv:1312.3393*. Version 2, submitted December 2013.

## A Further Discussion on LLM Judge Design

The design of an LLM preference judge can vary depending on the type of task PDO tackles. In this section, we present the rationale and provide a detailed analysis of our judge design for multiple-choice BBH tasks.

Since outputs in BBH are tied to fixed multiple-choice answers, the judging process naturally splits into two cases. (i) When the two prompts produce different answers, the judge’s role is straightforward: a capable judge should be able to identify the correct answer and select the corresponding prompt. (ii) When the two prompts produce the same answer, the situation is less clear. In this case, the answer alone cannot distinguish between the prompts, so we instead ask the judge to compare their reasoning chains. This design is well suited to BBH tasks, which are explicitly constructed to stress reasoning (Suzgun et al., 2022): the datasets involve multi-step deduction, object tracking, logical rules, or numerical reasoning, where strong performance correlates with the quality of intermediate steps. Our hypothesis is that examining the reasoning trace can reveal whether the model reached the correct answer through a coherent process or by shortcuts or luck.

This formalizes our dual-judge approach, as discussed in Section 5.1, where the judge evaluates either the answer or the reasoning depending on the case. To enable this, we prompt the model to always produce both a reasoning chain and a final answer using JSON-guided decoding.

### A.1 Validating the Dual-Judge Approach

To further probe the dual-judge approach, we conduct the following experiments. For each dataset, we select two prompts with different accuracy levels: prompt  $A$  as the **higher-accuracy** prompt and prompt  $B$  as the **lower-accuracy** prompt. We then test the judge in two scenarios: (i) whether prompt  $A$  is preferred when  $A$  is correct and  $B$  is wrong, and (ii) whether prompt  $A$  is preferred when both  $A$  and  $B$  produce correct answers. The results are reported in Figure 6.

**Result Analysis.** For most BBH tasks, we find strong evidence that the LLM judge reliably selects the higher-accuracy prompt when  $A$  is correct and  $B$  is wrong. We also observe moderate evidence that when both prompts produce the same answer, the judge prefers  $A$  based on the quality of its reasoning. Datasets where this preference is consistent under both **Answer** and **Reasoning** conditions (*Hyperbaton*, *Tracking-5*, *Tracking-7*, and *Web of Lies*) show superior performance in Table 1. On the other hand, when the judge is inconsistent in identifying the higher-accuracy prompt  $A$  in either condition, performance degrades (*Geometric*, *Salient*).

### A.2 Weighted Preference Matrix Update

Motivated by the above analysis in Figure 6, we consider a modification of the PDO algorithm for BBH tasks in which reasoning-based decisions, which are noisier than answer-based ones, are down-weighted so that they contribute less to the preference counts. Let  $W_{ij}$  and  $W_{ji}$  be the win counts used to form the Beta posteriors  $\text{Beta}(W_{ij}+1, W_{ji}+1)$  in D-TS. For a duel decided by source  $y \in \{\text{Answer}, \text{Reasoning}\}$ , we introduce a margin parameter  $\gamma_y \in [0, 0.5]$  and update

$$(W_{ij}, W_{ji}) \leftarrow (W_{ij} + 0.5 + \gamma_y, W_{ji} + 0.5 - \gamma_y)$$

whenever  $i$  is preferred, with the symmetric rule when  $j$  is preferred. This preserves one effective comparison per duel (sum increment = 1) while shrinking the winner–loser gap to  $2\gamma_y$  under noisier judgments. We fix  $\gamma_{\text{Answer}} = 0.5$  and ablate  $\gamma_{\text{Reasoning}} \in \{0.0, 0.2, 0.5\}$ . Figure 7 reports, for each dataset, the ground-truth accuracy *rank* of the Copeland leader over rounds. The results suggest that smaller values of  $\gamma_{\text{Reasoning}}$  (e.g., 0.2) often yield more stable progress than the full update  $\gamma_{\text{Reasoning}} = 0.5$  on tasks with noisier reasoning judgments, while tasks with consistent reasoning judgments remain robust across settings.

**Takeaway.** For BBH tasks, we use discounted updates as a simple mechanism to handle differences in judge reliability between answer-based and reasoning-based decisions in preference-based prompt optimization. More generally, for other tasks, LLM judges could be

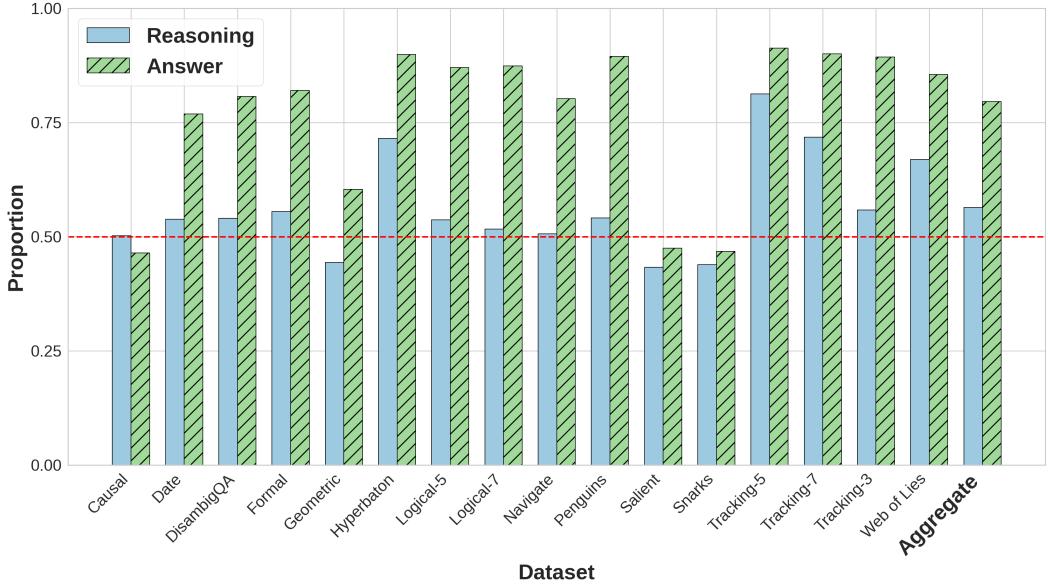


Figure 6: For each BBH dataset, we construct a prompt pair  $(A, B)$ , where  $A$  is the higher-accuracy prompt and  $B$  is the lower-accuracy prompt. We report the proportion of comparisons in which the judge prefers  $A$ . **Answer** (green) covers cases where  $A$  is correct and  $B$  is wrong; the judge evaluates by checking answers. **Reasoning** (blue) covers cases where both prompts produce the correct answer; the judge compares their reasoning chains. **Aggregate** pools all examples across datasets.

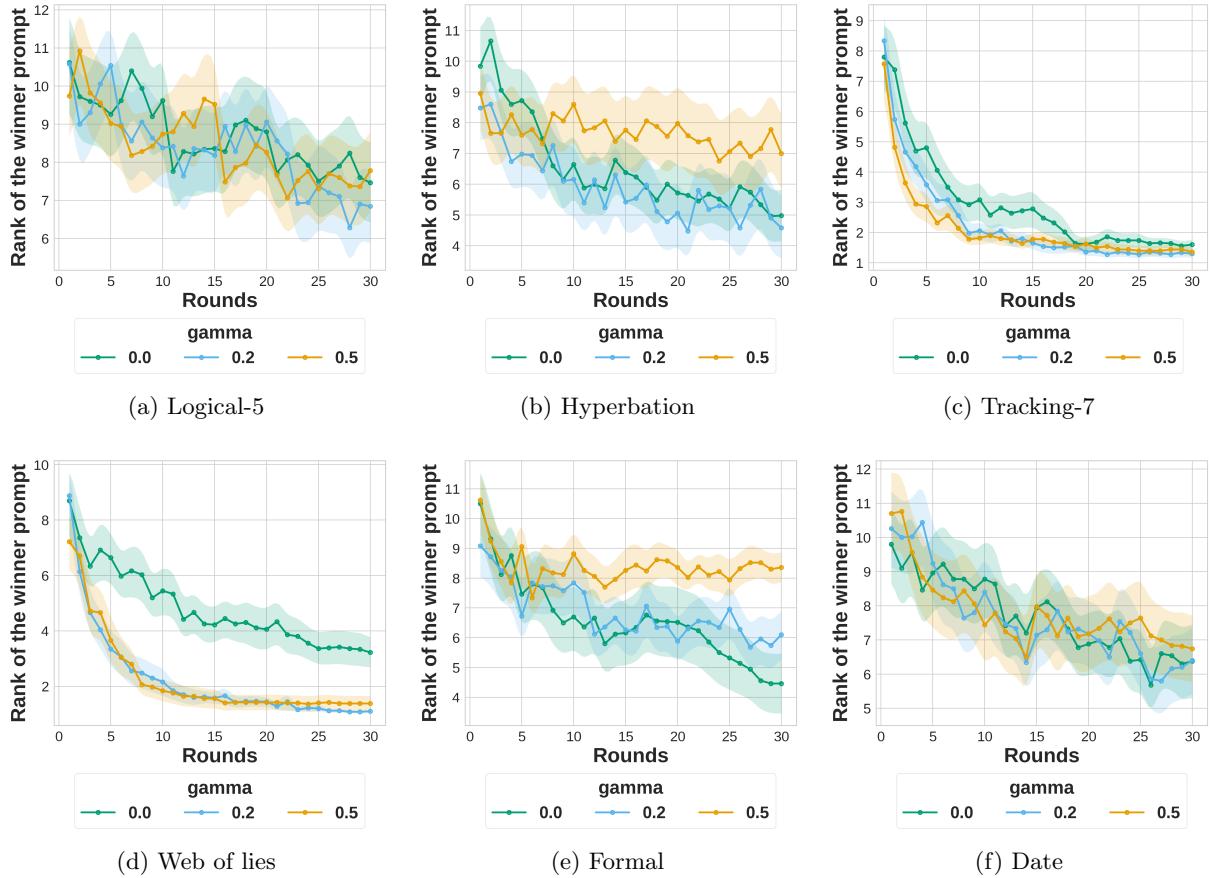


Figure 7: Effect of the reasoning-discount  $\gamma$  in the D-TS update across six BBH datasets. Each plot shows the ground-truth accuracy rank (lower is better) of the current Copeland leader over rounds. We fix  $\gamma_{\text{Answer}} = 0.5$  and ablate  $\gamma_{\text{Reasoning}} \in \{0.0, 0.2, 0.5\}$ . Results indicate that introducing a mild discount at  $\gamma_{\text{Reasoning}} = 0.2$  generally accelerates convergence and produces better final ranks overall compared to the undiscounted case  $\gamma_{\text{Reasoning}} = 0.5$ .

extended with adaptive adjustments that reflect reliability or with additional mechanisms to better capture uncertainty in judgments. Exploring such extensions is an interesting direction for future work.

### A.3 Case Study

We conduct a case study of the judge’s decisions in Table 4 on the BBH *Web of Lies* task by comparing a lower-accuracy Prompt A (0.852) with a higher-accuracy Prompt B (1.000). Two representative scenarios explain why the judge tends to prefer B: (i) when both prompts produce the correct answer, the reasoning-based judge compares the reasoning chains and favors B’s more concise and internally consistent rationale; (ii) when the prompts disagree and B’s answer is correct, the answer-based judge correctly selects B.

## B The Effect of Prompt Mutation

PDO extends the standard dueling-bandit framework with top-performer-guided mutation, which incrementally expands the initial candidate pool by mutating top-performing prompts. We demonstrate its effectiveness on two BBH datasets: *Web of Lies* and *Tracking-7*. Starting from  $|\mathcal{P}| = 20$  instructions, D-TS runs pairwise duels. For the *mutate* setting, at rounds 10 and 20 we prune the 10 lowest-Copeland-score prompts and generate 10 new candidates by mutating the current Copeland leader, then continue dueling with the updated pool. In contrast, the *non\_mutate* baseline keeps the initial 20 prompts fixed and selects leaders only from this pool. At each round we report the ground-truth accuracy of the current Copeland leader. As shown in Figure 8, the two curves coincide before round 10, after which mutation consistently improves accuracy on both *Web of Lies* and *Tracking-7*, moving beyond the limits of the original pool and steering the search toward stronger regions of the prompt space.

## C Experiment Details

### C.1 Hyperparameters

For the main experiments in Section 4.2, we initialize with 20 candidate prompts for BBH and 50 candidate prompts for MS-MARCO. Each experiment runs for 30 rounds, with 50

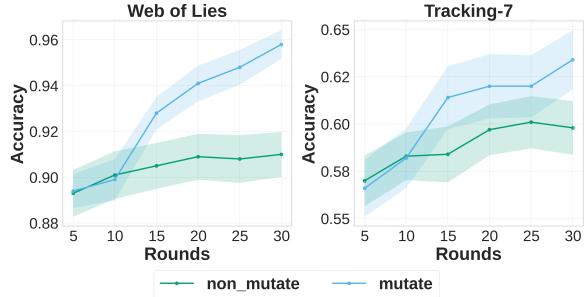


Figure 8: Effect of top-performer-guided mutation on *Web of Lies* (left) and *Tracking-7* (right).

duels per round. At rounds 10 and 20, we apply prompt mutation by selecting the top-3 prompts ranked by Copeland scores and generating 10 new prompts. At the same time, we prune the 10 lowest-ranked prompts by Copeland scores. We always select the prompt with the highest Copeland score as the winner, using the average win rate as a tiebreaker when multiple prompts share the same score. The D-TS parameter is fixed at  $\alpha = 1.2$  throughout all experiments.

### C.2 Prompt generation strategies

To generate the initial candidate prompts, we follow the approach in MiPROV2 (Opsahl-Ong et al., 2024), which first uses an LLM to construct a dataset summary from demonstration examples. The prompt generation process then incorporates this dataset summary, additional unlabelled demonstration examples, and randomly selected prompt tips from a predefined set to encourage diversity. In the mutation stage, we create new variants of the top-performing prompt by applying prompt tips to guide local edits, such as changing the tone, adjusting the wording, or appending synthetic few-shot examples.

### C.3 Baselines

**SPO** follows an iterative optimize-execute-evaluate loop, where new prompts are generated, executed, and then compared through pairwise LLM judgments of their outputs. For supervised APO methods, the typical pipeline involves generating, mutating, and scoring prompts. **APE** frames prompt design as black-box optimization, generating candidate instructions from input-output demonstrations and refining them

through iterative search. **OPRO** treats the LLM itself as the optimizer: a meta-prompt with task descriptions and prior results guides the model to propose and evaluate new candidates in an iterative loop. **Breeder** applies an evolutionary approach, jointly evolving task-prompts and mutation-prompts through LLM-driven mutation and selection.

#### C.4 Datasets

**BIG-bench Hard (BBH).** BBH (Suzgun et al., 2022) is a curated subset of BIG-bench consisting of 23 reasoning-intensive tasks. It is commonly used to stress multi-step reasoning and symbolic manipulation. In our experiments, we evaluate on 16 BBH multiple-choice tasks, where LLaMA-3.3-70B shows non-trivial sensitivity to instruction prompts.

**MS MARCO.** MS MARCO (Bajaj et al., 2016) is a large-scale question answering dataset built from real Bing search queries, paired with human-written answers and linked passages. It supports QA, passage ranking, and related IR/NLP tasks. In our setting, we focus on four task categories—Description, Entity, Numeric, and Location—and adopt a 1–5 integer scoring scheme from an LLM judge that compares model outputs against the dataset’s ground-truth answers.

## D Computational Complexity in # of LLM Calls

We analyze complexity purely in terms of the number of LLM calls, assuming each call has identical unit cost. All previously generated prompts’ predictions are cached and reused across rounds.

**Notation.**  $P$ : total number of prompts produced (including the initial baselines);  $P_0$ : number of initial baselines (constant);  $R$ : number of optimization rounds (one new prompt per round, so  $P = P_0 + R$ );  $B$ : number of examples used to score a *new* prompt per round (for caching its predictions);  $M$ : number of duels executed per round;  $N$ : number of labeled examples used to score each prompt in supervised APO. We treat mutation cost as  $\mathcal{O}(1)$  and omit it.

**Supervised APO with labels.** Each generated prompt is scored once on  $N$  labeled

examples:

$$\# \text{Calls}_{\text{sup}} = \mathcal{O}(P N).$$

**PDO (duels + caching).** At each round, we (i) predict for the new prompt on  $B$  examples, incurring  $B$  calls, and (ii) run  $M$  duels using cached predictions for older prompts, incurring  $M$  calls. Thus, the per-round cost is  $B + M$  calls. Over  $R$  rounds (equivalently,  $P - P_0$  new prompts),

$$\# \text{Calls}_{\text{PDO}} = \mathcal{O}(R(B+M)) = \mathcal{O}(P(B+M)).$$

**Comparison.**

Method	# LLM Calls
Supervised APO (labels)	$\mathcal{O}(P N)$
PDO (duels + caching)	$\mathcal{O}(P(B+M))$

**Remarks.** If  $B \ll N$ , PDO can require fewer calls than supervised APO because each new prompt is only partially evaluated and preferences guide the search. If  $B = N$  (new prompts fully evaluated as in supervised APO), then

$$\# \text{Calls}_{\text{PDO}} = \mathcal{O}(P(N+M)),$$

which is less efficient than supervised APO due to the additional  $M$  duels per round. Hence, PDO is advantageous in label-scarce regimes ( $B < N$ ), while supervised APO is preferable in label-rich regimes ( $B = N$ ).

Table 4: Case study of Reasoning-based and Answer-based judging on Prompt A (lower accuracy) and Prompt B (higher accuracy) for selected questions from the *Web of Lies* dataset in BBH.

Item	Content
<b>Prompt definitions</b>	
Prompt A (accuracy = 0.852)	Analyze the given statements about individuals telling the truth or lying, and determine the truthfulness of a specific person by iteratively applying logical deductions, stating assumptions before solving, and considering the chain of statements to arrive at a conclusion.
Prompt B (accuracy = 1.000)	Analyze the given statements about individuals telling the truth or lying, and determine the truthfulness of the person in question by following the chain of statements and applying logical reasoning. For example, if we have the statements: ‘Fidel lies. Fletcher says Fidel tells the truth. Yoland says Fletcher lies. Raymond says Yoland lies. Leda says Raymond lies,’ we can conclude that Leda tells the truth by tracing the statements: since Fidel lies, Fletcher’s statement that Fidel tells the truth is false, meaning Yoland’s statement that Fletcher lies is true, which in turn means Raymond’s statement that Yoland lies is false, and thus Leda’s statement that Raymond lies is true. Another example is: ‘Willian tells the truth. Phoebe says Willian lies. Alejandro says Phoebe lies. Lorine says Alejandro tells the truth. Christie says Lorine tells the truth,’ where we can deduce Christie tells the truth by following the chain of statements in a similar manner. Apply this reasoning process to the given question to determine the truthfulness of the person in question.
<b>Case I: Both prompts output the same correct answer → Reasoning-based judge</b>	
Question	Kristian lies. Sherrie says Kristian lies. Delbert says Sherrie lies. Jerry says Delbert tells the truth. Shalonda says Jerry tells the truth. <i>Does Shalonda tell the truth?</i>
Correct answer	No
Prompt A reasoning	Since Kristian lies and Sherrie says Kristian lies, Sherrie tells the truth. Delbert says Sherrie lies, so Delbert lies, which means Jerry lies when saying Delbert tells the truth, and consequently, Shalonda lies when saying Jerry tells the truth.
Prompt A answer	No
Prompt B reasoning	Since Kristian lies, Sherrie’s statement that Kristian lies is true. This means Delbert’s statement that Sherrie lies is false, so Jerry’s statement that Delbert tells the truth is false, and thus Shalonda’s statement that Jerry tells the truth is false.
Prompt B answer	No
Judge reasoning	Prompt B’s reasoning is more concise and directly addresses the truthfulness of each statement in a straightforward manner. It clearly states the implications of each person’s statement without introducing unnecessary complexity, making it easier to follow and understand. The logical flow from Kristian’s lie to Shalonda’s falsehood is more streamlined in B, enhancing clarity and coherence.
<b>Judge decision</b>	<b>B</b>

---

**Case II: Prompts output *different* answers (B correct, A incorrect)**

→ Answer-based judge

Question	Vina tells the truth. Helene says Vina lies. Kandi says Helene tells the truth. Jamey says Kandi lies. Ka says Jamey lies. <i>Does Ka tell the truth?</i>
Correct answer	No
Prompt A reasoning	If Ka lies, then Jamey tells the truth, which means Kandi lies, and thus Helene lies, contradicting Vina's truthfulness. Therefore, Ka must tell the truth to maintain the consistency of the statements.
Prompt A answer	Yes
Prompt B reasoning	Since Vina tells the truth, Helene's statement that Vina lies is false. This makes Kandi's statement that Helene tells the truth also false, which in turn makes Jamey's statement that Kandi lies true, and thus Ka's statement that Jamey lies is false, meaning Ka lies.
Prompt B answer	No
Judge reasoning	The correct answer is determined by tracing the statements back to Vina's truthfulness. Response B correctly identifies the chain: Vina true → Helene false → Kandi false → Jamey true → Ka's statement about Jamey false (Ka lies). This matches the task requirement.
Judge decision	<b>B</b>

---

**BBH: Answer-based Preference Judge Template**

```
## Role ##
You are a specialized judge focused on evaluating answer correctness when two responses give
different answers.
## Task ##
{question}
## Response from Prompt X ##
**Reasoning:** {reasoning_X}
**Answer:** {answer_X}
## Response from Prompt Y ##
**Reasoning:** {reasoning_Y}
**Answer:** {answer_Y}
## Your Task ##
The responses above give different answers: "{answer_X}" vs "{answer_Y}".
Your job is to determine which answer is more correct for the given task.
## Evaluation Criteria ##
Focus primarily on:
1. **Factual Accuracy** - Which answer better matches reality and task requirements?
2. **Task Alignment** - Which answer better fulfills the specific question asked?
## Output Format ##
{{
  "reasoning": "Your detailed justification explaining why prompt X or Y provided the more
               correct answer (~100 words).",
  "winner": "X or Y"
}}
```

## BBH: Reasoning-based Preference Judge Template

```
## Role ##
You are a specialized judge focused on evaluating reasoning quality when two responses give
the same answer.
## Task ##
{question}
## Response from Prompt X ##
**Reasoning:** {reasoning_X}
**Answer:** {answer_X}
## Response from Prompt Y ##
**Reasoning:** {reasoning_Y}
**Answer:** {answer_Y}
## Your Task ##
The responses above give the same answer: "{answer_X}".
Since both arrive at the same conclusion, your job is to determine which reasoning process
is better.
## Evaluation Criteria ##
Focus primarily on:
1. **Logical Coherence** - Is the reasoning chain clear and well-structured?
2. **Completeness** - Does the reasoning address all key aspects of the problem?
3. **Clarity** - Is the reasoning easy to follow and understand?
4. **Accuracy** - Are the intermediate steps and assumptions correct?
## Output Format ##
{{
  "reasoning": "Your detailed justification explaining why prompt X or Y provided better
    reasoning (~100 words).",
  "winner": "X or Y"
}}
```

## MS-MARCO: Preference Judge Template

```
## Role ##
You are a meticulous, impartial referee evaluating two competing answers to determine which
better answers the given question based on the provided context.
## Query ##
{query}
## Context ##
{context}
## Answer from Prompt X ##
{answer_X}
## Answer from Prompt Y ##
{answer_Y}
## Evaluation Criteria ##
Compare both answers based on:
1. **Accuracy** - How factually correct is each answer based on the context?
2. **Completeness** - Does the answer address all aspects of the question?
3. **Relevance** - How well does the answer stay focused on answering the question?
4. **Clarity** - How clear and well-articulated is the answer?
## Output Format ##
{{
  "reasoning": "Your detailed justification explaining why answer X or Y is better (~100
words).",
  "winner": "X or Y"
}}
```

## MS-MARCO: final evaluations with ground-truth references

```
"""
Begin your evaluation by carefully comparing the AI-generated answer with the reference
solution. Identify any outputs that are not true, as well as omissions or deviations,
and simulate human judgments in explaining how these impact the overall quality of the
response. Ensure that your assessment is objective and consistent.
At the end of your evaluation, assign a score from 1 to 5 based on the following scale:
- 1: Very poor - does not meet the requirements or is significantly incorrect.
- 2: Poor - contains major errors or omissions.
- 3: Fair - adequate but with notable flaws.
- 4: Good - meets the requirements with minor errors.
- 5: Excellent - fully accurate and well-articulated.

[User Question]
{question}

[Reference Solution]
{ground_truth}

[AI-Generated Answer]
{prediction}

Your response should be a valid JSON string (no backticks) following this schema:
{{ "explanation": "{{Detailed reasoning based on the comparison}}"
  "score": {{1-5}}
}}
"""

```

## Template for generating initial prompts

```
"""
You are an expert prompt-engineer. Your task is to generate **exactly 1** *high-quality* *
system-level instruction** for the target reasoning task.

# Dataset Snapshot
Below is an *LM-written summary* of the unlabeled question pool:
{dataset_summary}

# Sample Inputs (do NOT answer them)
{questions}

# Prompt-Engineering Tip
{tip}

# Output Format (STRICT)
Return **exactly** 1 item in a JSON array, *and nothing else*:
[ "Your single high-quality instruction here" ]
"""

```

## Template for mutating on top-performing prompts

```
"""
You are an expert prompt-engineer specializing in prompt optimization.
Your task is to generate 1 *diverse, high-quality* **mutation** of the currently **BEST
PERFORMING** instruction for the target reasoning task.

# BEST PERFORMING Instruction (Current Champion)
{instructions}

# CRITICAL: Follow This Prompt-Engineering Tip
{tip}

# Output Format
Return ***exactly** 1 mutated instruction in a JSON object, *and nothing else*:
{ "mutated_prompt": "Your mutated instruction here, following the tip." }
"""
```

## Prompt engineering tips for prompt generations

```
# Mutation tips for top-performer-guided prompt mutations
MUTATION_TIPS = {
    "expansion": "Keep the current champion instruction exactly as is, but expand on it by
    adding additional helpful guidance or clarifications. The result should be the original
    instruction plus new supplementary content.",
    "minimal": "Make very minimal changes to the current champion instruction. Keep it
    around the same length and modify only a few words through paraphrasing while preserving
    the core meaning.",
    "few_shot": "Add a few concrete examples to the current champion instruction to
    demonstrate the expected reasoning process or output format. Include 1-3 brief example
    cases that show how to apply the instruction.",
    "emphasis": "Adjust the tone, emphasis, or directional focus of the current champion
    instruction to create different reasoning patterns.",
}

# Initial instruction generation tips
INITIAL_INSTRUCTION_TIPS = {
    "framing": "Set the context for the task by framing it as a concrete creative scenario
    .",
    "simple": "Keep the instruction clear and concise.",
    "description": "Make sure your instruction is very informative and descriptive.",
    "persona": "Provide the LM with a creative persona that is relevant to the task.",
    "edge_cases": "List tricky cases the instruction must handle.",
    "assumptions": "Have the model state assumptions before solving.",
}
```