Becoming Experienced Judges: Selective Test-Time Learning for Evaluators

Seungyeon Jwa[1] **Daechul Ahn**[1] **Reokyoung Kim**[1] **Dongyeop Kang**[2] **Jonghyun Choi**[1]

[1]Seoul National University [2]University of Minnesota

{amyj97,daechulahn,reokyoungkim,jonghyunchoi}@snu.ac.kr dongyeop@umn.edu

Abstract

Automatic evaluation with large language models, commonly known as *LLM-as-a-judge*, is now standard across reasoning and alignment tasks. Despite evaluating many samples in deployment, these evaluators typically (i) treat each case independently, missing the opportunity to accumulate and reuse evaluation insights across cases, and (ii) rely on a single fixed prompt for all cases, neglecting the need for sample-specific evaluation criteria. We introduce **Learning While Evaluating** (LWE), a framework that enables evaluators to improve *sequentially* at inference time without requiring additional training or external signals. LWE maintains an evolving *meta-prompt* that (i) stores evaluation insights derived from self-generated feedback during sequential testing, and (ii) allows evaluators to leverage these insights to generate sample-specific evaluation instructions for accurate and consistent judgments. Furthermore, we propose *Selective* LWE, which updates the meta-prompt only on self-inconsistent cases, focusing computation where it matters most. This selective approach retains the benefits of sequential learning while being far more cost-effective. Across two pairwise comparison benchmarks, *Selective* LWE outperforms strong baselines, empirically demonstrating that evaluators can improve during sequential testing with a simple selective update—learning most from the cases they struggle with.[1]

1 Introduction

Large language models (LLMs) and vision-language models (VLMs) are increasingly used as automatic evaluators, commonly referred to as *(V)LLM-as-a-judge* (Zheng et al., 2023; Chen et al., 2024). Despite their growing importance, current evaluators remain surprisingly rigid: they typically apply a single evaluation prompt for all cases and evaluate each case in isolation, as shown in Fig. 1 ("Vanilla Judge"). This rigidity limits evaluators in two key ways: it prevents tailoring evaluation criteria to individual test cases and the reuse of insights gained from earlier decisions. This motivates our research question: *can evaluators adapt and improve during testing?*

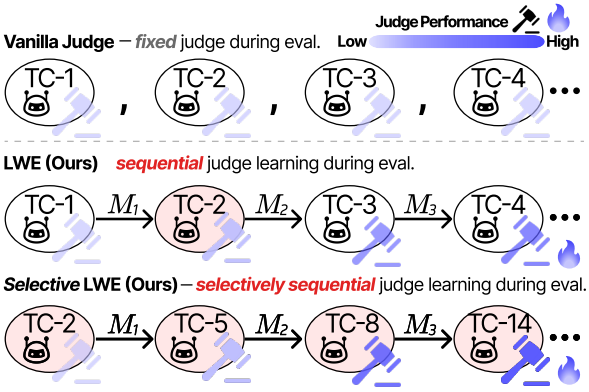

Figure 1: **Comparison of three evaluation approaches.** A vanilla judge evaluates each test case (TC) independently using a fixed prompt throughout evaluation (eval.). LWE (ours) employs a *meta-prompt* (M) that evolves sequentially as the evaluator progresses through test cases, enabling sample-specific tailoring and continual improvement during evaluation. *Selective* LWE (ours) further enhances efficiency by updating the meta-prompt only on challenging cases (*e.g.*, the red-highlighted TCs 2, 5, 8, 14), preserving performance gains while substantially reducing computational overhead. The color gradient illustrates progressive improvement of the judge's performance over time.

Humans routinely exhibit such test-time improvement. A student taking an exam refines their strategy as they progress, and a judge becomes more reliable as they accumulate experience across diverse cases (Flavell, 1979; Sternberg, 1985). Recent test-time scaling approaches move in a similar direction by allocating more inference computation per sample, but the extended reasoning they produce is discarded immediately afterward, leaving

[1]Code: https://github.com/snumprlab/lwe

the thought process inherently sample-independent. Thus, current automatic evaluators cannot retain or refine their reasoning *across* cases, raising a methodological question: how can we enable evaluators to learn from their evaluation experience?

To support such learning, an evaluator must be able to retain and use experience from one case to the next. Recent progress in non-evaluation settings (Wang et al., 2024b; Liu et al., 2025; Chen et al., 2025) demonstrates that models can improve through sequential inference based on gained information across test cases. In particular, Suzgun et al. (2025) introduce a memory prompt that reflects on similar past and current examples to generate the adaptive memory.

These approaches highlight the value of leveraging self-generated insights accumulated during sequential problem solving. However, evaluation requires a different form of knowledge: an evaluator should maintain stable, general evaluation principles while also adapting its judging criteria to each sample. To address this, we introduce a *meta-prompt* that explicitly separates general evaluation principles from sample-specific prompts that adapt these principles to each case.

We propose **Learning While Evaluating** (LWE), a framework that enables evaluators to maintain and continually refine their *evaluation insights* as they process test cases. At the core of LWE, a meta-prompt serves as a persistent repository of evaluation insights discovered during testing and generates customized evaluation criteria and steps for each new case (Sec. 3.1). Through self-feedback mechanisms, LWE continually improves and expands its meta-prompt during testing, enabling evaluators to learn—rather than merely repeat—their evaluation behavior (Fig. 2).

Furthermore, while LWE enables evaluators to learn from experience, updating every sample is computationally expensive and often unnecessary, since some test cases are already straightforward and can be accurately judged without additional reasoning. To focus updates on cases where more refined evaluation criteria are actually needed, we exploit a test-time–accessible signal that indicates when the evaluator's fixed judging criteria are insufficient. Prior work (Zheng et al., 2023; Shi et al., 2024; Koo et al., 2024) reports that positional bias can lead a model to give different judgments depending on which answer is presented first, revealing uncertainty or conflict in its internal criteria. Leveraging this inconsistency as a test-time signal, we propose *Selective* LWE, which updates the meta-prompt only on such inconsistent cases, retaining the benefits of sequential learning while substantially reducing computational overhead (Sec. 3.2).

Experiments on two pairwise comparison benchmarks (Li et al., 2025a; Yasunaga et al., 2025) show that our framework improves accuracy and consistency over strong baselines while reducing inference cost. Overall, we demonstrate that evaluators need not remain static: they can also *learn from experience*, especially from the cases that challenge them the most.

2 Related Work

2.1 LLM-as-a-Judge

LLM-as-a-judge (Zheng et al., 2023) has become a standard alternative to human evaluation, supporting improvements in model behavior during training (Glaese et al., 2022; Bai et al., 2022; Ouyang et al., 2022; Lee et al., 2023; Yuan et al., 2024) as well as inference (Madaan et al., 2023; Shinn et al., 2023; Zhang et al., 2024a). Beyond textual settings, VLM-as-a-judge plays a role in the automatic evaluation of multimodal tasks, including image–text alignment (Chen et al., 2024).

Pitfalls of LLM-as-a-judge have also been reported, including sensitivity to prompt permutations and various forms of bias, most notably position bias (Zheng et al., 2023; Shi et al., 2024; Koo et al., 2024; Park et al., 2024; Zhao et al., 2025; Bavaresco et al., 2025; Schilcher et al., 2025). To improve evaluation capability, existing approaches often rely on training (Kim et al., 2023; Vu et al., 2024; Yu et al., 2025; Whitehouse et al., 2025; Chan et al., 2025; Lee et al., 2024; Zang et al., 2025; Wang et al., 2025a) or inference-time schemes with multiple models (Jung et al., 2025; Li et al., 2025b). Related lines that learn meta-reward models or meta-reasoning procedures (Kim et al., 2025; Saha et al., 2025) aim to generate case-specific evaluation criteria or reasoning steps, but they also rely on additional training. In contrast, we pursue an inference-only approach that avoids any further training and requires no additional models, making it readily deployable in real-world settings.

2.2 Prompt Optimization

Prompt optimization aims to discover task-specific prompts that enhance model performance (Yang et al., 2023; Guo et al., 2024; Khattab et al., 2024;

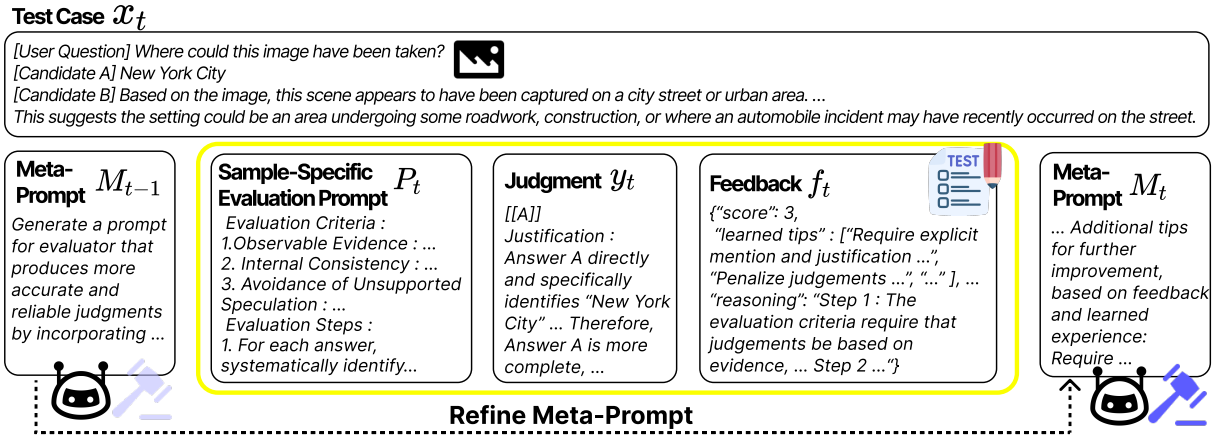

Figure 2: **Overview of proposed LWE.** Given a test case x_t, the meta-prompt M_{t-1} generates a sample-specific evaluation prompt P_t, which the evaluator uses to produce a judgment y_t. The evaluator then reflects on its decision to produce self-feedback f_t, which is incorporated into the meta-prompt to form M_t for subsequent cases.

Fernando et al., 2023; Wang et al., 2024a; Yuksekgonul et al., 2025; Xu et al., 2025; Agrawal et al., 2025; Zehle et al., 2026). These methods iteratively refine prompts using validation sets or external feedback, after which a single optimized prompt is fixed and applied uniformly to all test samples. This paradigm is effective, yet inherently *static*: it cannot adapt to sample-specific variations during evaluation. Moreover, the reliance on pre-constructed validation sets for optimization can be impractical in real-world evaluation scenarios where constructing labeled data is costly or infeasible. To address these limitations, our framework performs meta-prompt evolution without any validation data and generates sample-specific prompts based on the evolving meta-prompt.

2.3 Test-Time Scaling

Per-instance test-time scaling. Recent studies report that allocating more computation at inference time can substantially enhance reasoning quality by increasing deliberation depth, exploring multiple sampled traces, or performing structured search (Snell et al., 2024; Muennighoff et al., 2025; Shen et al., 2025; Wang et al., 2025b). However, these approaches operate on a *per-instance* basis: once an instance is completed, its intermediate reasoning is discarded, limiting the model's ability to leverage accumulated experience across test cases.

Sequential test-time scaling. Beyond per-instance compute, several works explore how models can *learn across* test cases by accumulating and reusing experience during inference time (Wang et al., 2024b; Liu et al., 2025; Chen et al., 2025; Suzgun et al., 2025; Huang et al., 2025). These methods show that models can refine and reuse task-specific heuristics to improve future predictions. A related but distinct line of work, test-time training (Sun et al., 2020, 2024), instead adapts model weights at inference via self-supervised objectives. We instantiate these ideas for LLM-as-a-judge through an evolving meta-prompt that stores insights for tailoring sample-specific evaluation instructions and refines itself via self-feedback.

3 Approach

We propose that evaluators could benefit from insights accumulated across test cases. In this work, we focus on the pairwise evaluation setup, where the judge compares two candidate responses and identifies the better one. To this end, we introduce Learning While Evaluating (LWE), a test-time learning framework that allows evaluators to accumulate and draw on experience from earlier cases during evaluation, ultimately improving their decision-making.

3.1 Learning While Evaluating

Figure 2 and Algorithm 1 outline the overall procedure of LWE. Unlike traditional approaches that evaluate each sample independently, LWE processes samples *sequentially*, accumulating insights from previous judgments. It maintains an evolving meta-prompt that (i) produces sample-specific evaluation prompts and (ii) integrates self-generated feedback to improve subsequent evaluations.

Sequential meta-prompt evolution. Let $\mathcal{D} = \{x_1, x_2, \ldots, x_T\}$ denote a set of pairwise compar-

Algorithm 1 Learning While Evaluating (LWE)

Input: A base LLM `LLM`, a test set $D_{\text{test}} = \{x_t\}_{t=1}^{T}$, an initial meta-prompt M_0 and batch size b

Output: A set of evaluated results S and a final meta-prompt M

$M \leftarrow M_0$ ▷ Initialize a meta-prompt.
$S \leftarrow \emptyset$ ▷ Initialize a set of evaluated results.
$F \leftarrow \emptyset$ ▷ Buffer for feedback within a batch.
for $t = 1$ to T **do**
  $x_t \leftarrow D_{\text{test}}[t]$
  $P_t \leftarrow \texttt{BuildEvalPrompt}_{\texttt{LLM}}(M, x_t)$
  $y_t \leftarrow \texttt{Judge}_{\texttt{LLM}}(P_t, x_t)$
  $f_t \leftarrow \texttt{Feedback}_{\texttt{LLM}}(M, P_t, x_t, y_t)$
  $S \leftarrow S \cup \{(x_t, y_t)\}$
  $F \leftarrow F \cup \{f_t\}$
  if $|F| = b$ **or** $t = T$ **then**
    $M \leftarrow \texttt{RefineMetaPrompt}_{\texttt{LLM}}(M, F)$ ▷ Update the meta-prompt using batch feedback.
    $F \leftarrow \emptyset$
  end if
end for
return S, M

Algorithm 2 *Selective* Learning While Evaluating

Input: A base LLM `LLM`, a test set D_{test}, a vanilla prompt P, an initial meta-prompt M_0 and batch size b

Output: A set of evaluated results S and a final meta-prompt M

$S \leftarrow \emptyset$ ▷ Initialize a set of evaluation results.
$I \leftarrow \emptyset$ ▷ Initialize a set of inconsistent cases.
for $x \in D_{\text{test}}$ **do**
  $y^{(AB)} \leftarrow \texttt{Judge}_{\texttt{LLM}}(P, x)$
  $x' \leftarrow x$ with the response order swapped
  $y^{(BA)} \leftarrow \texttt{Judge}_{\texttt{LLM}}(P, x')$
  if $y^{(AB)} = y^{(BA)}$ **then**
    $S \leftarrow S \cup \{(x, y^{(AB)})\}$ ▷ Consistent case; skip the update.
  else
    $I \leftarrow I \cup \{x\}$ ▷ Collect inconsistent cases.
  end if
end for
$(S', M) \leftarrow \texttt{LWE}(\texttt{LLM}, I, M_0, b)$
$S \leftarrow S \cup S'$
return S, M

ison cases. Starting from an initial meta-prompt M_0, the evaluator sequentially updates its meta-prompt based on observed test cases. For each test sample x_t, the current meta-prompt M generates a sample-specific prompt P_t (`BuildEvalPrompt`) to produce a judgment y_t (selecting the better response) (`Judge`). The evaluator then reflects on this decision to generate feedback f_t (`Feedback`) for refining M (`RefineMetaPrompt`). These steps are all realized via LLM prompting.

To prevent overfitting to individual samples, we update the meta-prompt once per batch of b samples (with $b = 4$ in our experiments). We analyze the effect of different batch sizes in Figure 7. Through this sequential process, the meta-prompt captures transferable evaluation heuristics and progressively refines its evaluation strategy across the test set.

3.2 *Selective* Learning While Evaluating

While sequential updates allow leveraging experience across samples, updating the meta-prompt on *every* sample incurs non-trivial computational overhead. Moreover, not all samples require the same level of deliberation—straightforward comparisons are already judged accurately by the vanilla evaluator and thus gain little from additional information, whereas challenging cases demand extra guidance for reliable evaluation.

Motivated by prior observations that LLM judges are sensitive to position bias (Zheng et al., 2023; Shi et al., 2024; Koo et al., 2024; Zhang et al., 2024b) and by recent work using self-consistency as a training signal (Wen et al., 2025), we turn this vulnerability into a *test-time signal*: we treat order-swap *inconsistency* (*i.e.*, disagreement between A *vs.* B and B *vs.* A judgments) as a label-free proxy for evaluator uncertainty.

This choice is attractive for three reasons: (i) it is *available at test time* without ground-truth labels; (ii) it is *instance-specific*, flagging precisely those cases where the current evaluation policy is confused; and (iii) it *targets compute* to the examples that most need additional guidance, avoiding wasted updates on straightforward cases that already yield consistent decisions.

Concretely, as formalized in Algorithm 2, for each sample x, the evaluator makes two vanilla judgments with the response order swapped, and an update is triggered only if the two judgments disagree. This selective mechanism bypasses samples with consistent judgments and focuses computational resources on ambiguous cases, improving efficiency without sacrificing accuracy, as we show empirically in Sec. 5.

4 Experimental Setup

4.1 Baselines

We compare our proposed methods LWE and *Selective* LWE against representative inference approaches.

Fixed prompting. `Vanilla` applies a fixed prompt for all test cases. `Chain-of-Thought (CoT)` (Wei et al., 2022) augments the vanilla prompt with an explicit "step-by-step reasoning" instruction. `Majority Voting` takes the majority label from five independent stochastic judgments of the vanilla prompt (temperature 0.7). `TextGrad` (Yuksekgonul et al., 2025) iteratively modifies its prompt based on validation sets. The optimized prompt is then fixed and applied uniformly to all test cases. Among all baselines, it is the only method that requires additional labeled data prior to test-time evaluation.

Adaptive prompting. `Dynamic Cheatsheet` (Suzgun et al., 2025) (DC) maintains a memory prompt that is sequentially updated during test-time. We use the strongest variant, `DC-RS`, which retrieves similar examples and curates a memory prompt that conditions the model's response generation. It is a strong baseline but does not explicitly generate sample-specific evaluation prompts, and it performs updates on every sample unconditionally. `Sample-Specific Prompt` generates a tailored evaluation prompt for each test case from a fixed meta-prompt. Although it applies different prompts per sample, its fixed meta-prompt configuration provides a baseline that isolates the effect of sequence-level meta-prompt updates in our method.

We use `gpt-4.1-2025-04-14` (OpenAI, 2025) (gpt-4.1) as a base evaluator for experiments.

4.2 Benchmarks

We evaluate our method on two multimodal pairwise comparison benchmarks, where the evaluator selects the better response between two candidates.

VL-RewardBench (VLRewardBench, VL) (Li et al., 2025a) and **Multimodal RewardBench** (MMRewardBench, MM) (Yasunaga et al., 2025) each provide an image, a question, and two textual responses, and the evaluator must determine which response is better. Following prior work (Suzgun et al., 2025), we use representative subsets where necessary due to API budget constraints: we evaluate on the full 1,247 examples of VLRewardBench, and 1,000 out of 4,711 examples from MMRewardBench.

4.3 Evaluation Metrics

We consider four metrics. `Accuracy` is measured on a single random ordering of the two responses. It reflects single-pass performance without accounting for position-bias effects. `Consistency` measures stability under response-order swapping. `Pair Accuracy`, a stricter metric than Accuracy, requires both the original and swapped predictions to be correct and therefore reflects robustness to position bias. This metric is particularly important for assessing the reliability of judge models, as it captures whether their decisions remain stable under input permutations. `Relative Inference Cost` computes the total character length of input and output normalized to the vanilla baseline, providing a quality–cost trade-off measure. All metrics are macro-averaged across benchmarks.

5 Results

Across our experiments, *Selective* LWE consistently improves evaluation quality while operating at substantially lower inference cost than existing adaptive methods.

Main results. Table 1 presents the overall results on the two benchmarks using gpt-4.1. *Selective* LWE not only matches or exceeds most baselines in accuracy but also achieves the highest consistency and pair accuracy across all benchmarks, operating at a substantially lower inference cost. Notably, *Selective* LWE delivers these gains with only 3.9× relative inference cost, compared to higher costs for other adaptive methods, *i.e.*, 4.4× for TextGrad and 12.9× for DC.

Learning during evaluation. Figure 3 visualizes the cumulative accuracy on the vanilla-inconsistent subsets of the two benchmarks. First, the Sample-Specific Prompt already outperforms the vanilla baseline, demonstrating the benefit of tailoring the evaluation criteria to each case. Building on this, *Selective* LWE achieves even larger gains. These results suggest that, as *Selective* LWE progressively refines its meta-prompt through targeted updates, it accumulates evaluation heuristics that extend beyond individual instances, yielding overall improvements in accuracy throughout the evaluation trajectory.

Method	VLRewardBench			MMRewardBench			Relative Inference Cost
	Acc. (↑)	Cons. (↑)	PairAcc. (↑)	Acc. (↑)	Cons. (↑)	PairAcc. (↑)	Input & Output Text (↓)
Vanilla	0.629	0.801	0.529	0.808	0.863	0.747	1.0×
CoT	0.651	0.808	0.553	0.808	0.874	0.749	1.2×
Majority Voting	0.627	0.810	0.537	0.828	0.891	0.769	5.0×
TextGrad*	0.730	0.749	0.615	0.821	0.836	0.741	4.4×
Dynamic Cheatsheet	0.698	0.868	0.629	0.811	0.901	0.764	12.9×
Sample-Specific Prompt	0.661	0.727	0.529	0.815	0.865	0.742	2.5×
LWE (**Ours**)	**0.745**	0.805	0.646	0.799	0.846	0.727	10.9×
Selective LWE (**Ours**)	0.676	**0.940**	**0.648**	**0.836**	**0.947**	**0.808**	3.9×

Table 1: **Performance of various inference strategies across benchmarks.** We evaluate two vision–language benchmarks, reporting accuracy (Acc.), consistency (Cons.), and pair accuracy (PairAcc.). The rightmost column shows the relative inference cost, measured by the total input and output character length normalized to the vanilla baseline. Since LWE and *Selective* LWE are inherently order-sensitive, we report the averaged results over three random-order runs (see Appendix H for full results). Bold denotes the best performance. TextGrad* relies on gold-labeled validation data and thus operates under a more favorable setting than the other methods (see Appendix B).

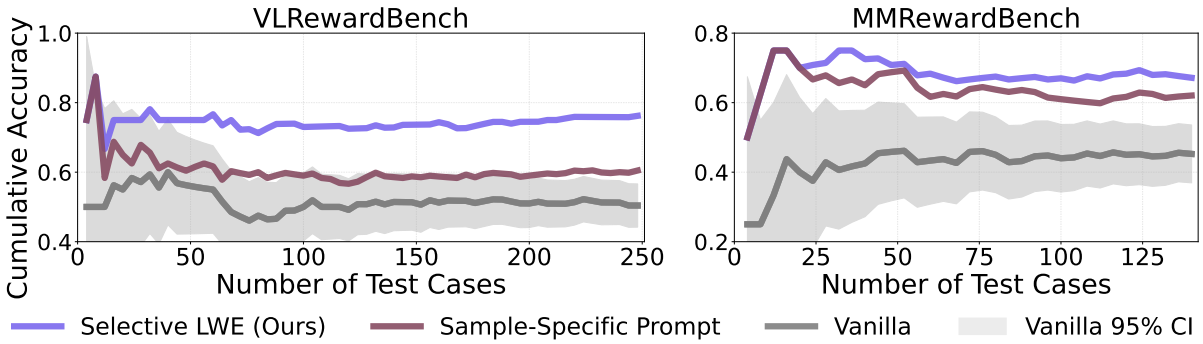

Figure 3: **Cumulative accuracy over test cases.** Curves are computed on the vanilla-inconsistent subsets of each benchmark, where *Selective* LWE performs updates. Gray-shaded areas indicate confidence intervals for the vanilla baseline, computed at each point using the binomial proportion method with significance level $\alpha = 0.05$.

Full updates vs. *Selective* updates. As shown in Table 1, the inference strategies with updating for all test cases, *i.e.*, DC and LWE, generally improve all three metrics but require substantially higher inference costs, 12.9× and 10.9× respectively, compared to the vanilla method. In contrast, *Selective* LWE preserves most of these gains, often even surpassing the full sequential variant, while operating at only 3.9× the inference cost (approximately 36% of LWE). This 3.9× budget reflects two vanilla passes for consistency checking and an additional 1.9× for the LWE process. This demonstrates that not every test sample requires equal inference effort: focusing updates on confusing cases provides a more cost-effective path to improved evaluation quality, achieving superior performance and efficiency compared to uniform test-time scaling that updates every sample indiscriminately.

Reliability of judgments. By focusing updates on inconsistent cases, *Selective* LWE gains a substantial increase in consistency (up to 0.947), while also yielding a large margin of pair accuracy. This improvement in pair accuracy is particularly meaningful, as it reflects position-invariant evaluation behavior. In real-world evaluation, judges typically assess responses in a random order, so higher pair accuracy implies that such one-shot judgments are more *reliable*.

6 Analysis

We further analyze the mechanisms underlying the effectiveness of *Selective* LWE, examining whether its selective sequential updates identify informative cases and how robust the resulting behavior is across settings.

Validity of the selection signal. We first examine whether the inconsistency-based signal reliably identifies beneficial samples for meta-prompt updates. As shown in Figure 4, the performance gains of LWE over the vanilla baseline increase monotonically with the proportion of inconsistent samples. Notably, LWE maintains stable performance even on subsets where the vanilla evaluator's pair accuracy is zero. This reveals an interesting pattern:

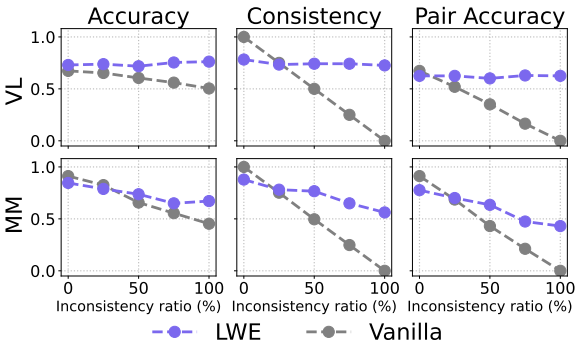

Figure 4: **Effect of inconsistency ratio on meta-prompt updates.** We evaluate LWE on subsets containing different proportions of inconsistent samples, where inconsistency is computed based on vanilla predictions. As the inconsistency ratio increases, LWE shows larger performance gains over the vanilla baseline (purple *vs.* gray), highlighting its effectiveness in handling inconsistent cases. The total number of samples is fixed: 248 for VLRewardBench and 137 for MMRewardBench.

the evaluator benefits most from the samples that confuse it, making inconsistency an effective guide for updates. Consequently, the *Selective* mechanism allocates computation where refinement is most impactful.

What the meta-prompt learns. To understand how selective updates improve evaluator behavior, we qualitatively examine how the meta-prompt changes during refinement. Figure 5 shows that, rather than accumulating ad-hoc advice, updates progressively distill more structured evaluation principles and sharper criteria for distinguishing subtle differences between responses. For example, directing the evaluator not only to detect errors but to compare cases where both answers contain issues, assess the impact of overstatements or omissions, and weigh these factors to reach a more balanced and fair judgment. This indicates that the meta-prompt is internalizing reliable and portable guidance for future evaluations. See Appendix F for evaluation examples.

Robustness to sample ordering. Since LWE performs sequential updates, we test sensitivity to sample ordering by evaluating each method on three random permutations of each test set. As shown in Figure 6, results remain consistent across three runs with small standard deviations; on MMRewardBench, the variance of pair accuracy for LWE is effectively zero and nearly zero for *Selective* LWE. This result suggests that learning effects arise from accumulated experience, rather than incidental ordering artifacts. The complete results are provided in Appendix H.

. . . **Additional Tips:**
- Instruct evaluators to systematically check for calculation or logical errors in each answer, not just the reasoning structure penalize both overstatements (e.g., exaggerating color, quantity, or significance) and omissions (e.g., failing to mention a visible but relevant feature), and to penalize answers with such errors even if the explanation appears logical. weigh the impact of these errors on the overall evaluation. . . .
- Require evaluators to ensure that their final justification is comprehensive, not only comprehensive but also balanced, addressing both the strengths and weaknesses of each answer in relation to every evaluation criterion and step, and not omitting any relevant aspect of the comparison.

Figure 5: **Illustration of how the meta-prompt evolves after a single refinement step.** Red and blue text denotes instructions removed and added during the update, reflecting the shift from loosely specified checks to clearer, more structured heuristics under LWE. The full meta-prompts are provided in Appendix G.

Effect of batching. We analyze the number of samples processed per update (b), as this hyperparameter directly influences two key factors of LWE: evaluation performance and inference cost. Figure 7 shows that updating after every sample (b=1) incurs the highest computational cost, mirroring the overhead of DC, and does not necessarily yield the best performance. Conversely, overly large batches (b=8) degrade performance because the context used for meta-prompt updates becomes excessively long, making the refinement unstable. A moderate batch size (b=4) provides the best balance, sustaining strong accuracy while keeping inference cost relatively low, and we therefore adopt b=4 as the default configuration of LWE.

Generalization across evaluators. Lastly, we assess whether the benefits of *Selective* LWE extend across different backbone models. As shown in Table 2, with `gemini-2.5-pro` (Comanici et al., 2025), which already achieves strong vanilla performance, *Selective* LWE still yields measurable gains, indicating that the selective updates extract additional signal even when the base evaluator is already competent.

On the other hand, `claude-sonnet-4.5` (Anthropic, 2025) shows substantially larger improvements. This is explained by *Selective* LWE 's strength on vanilla-inconsistent cases: claude-sonnet-4.5 produces a large number of such confus-

Model	Method	VLRewardBench			MMRewardBench			Relative Inference Cost
		Acc. (↑)	Cons. (↑)	PairAcc. (↑)	Acc. (↑)	Cons. (↑)	PairAcc. (↑)	Input & Output Text (↓)
gemini-2.5-pro	Vanilla	0.754	0.888	0.696	0.858	0.925	0.820	1.0×
	Selective LWE	**0.768**	**0.955**	**0.744**	**0.865**	**0.969**	**0.852**	3.2×
claude-sonnet-4.5	Vanilla	0.473	0.490	0.327	0.540	0.482	0.419	1.0×
	Selective LWE	**0.703**	**0.881**	**0.648**	**0.823**	**0.879**	**0.771**	15.2×

Table 2: **Generalization across evaluators.** Results on gemini-2.5-pro and claude-sonnet-4.5 show that *Selective* LWE consistently improves evaluation performance across evaluators. The results of *Selective* LWE are averaged over three random-order runs (see Appendix H for full results).

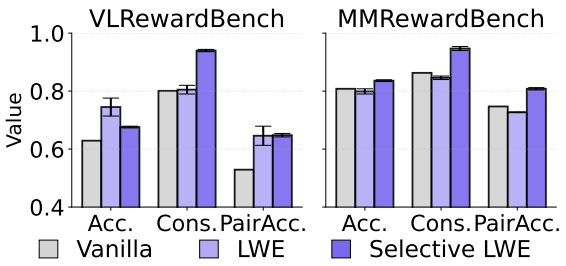

Figure 6: **Ablation on test case ordering.** We evaluate the sensitivity of LWE to input ordering using three random permutations of each test set. Bars report mean ± standard deviation across permutations. *Selective* LWE exhibits stable performance across different orderings.

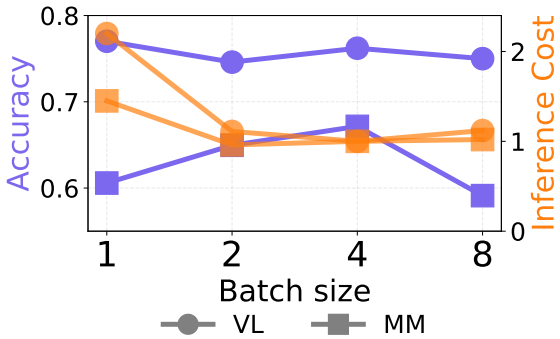

Figure 7: **Effect of batch size** (b) **on accuracy and inference cost.** Experiments are conducted on vanilla-inconsistent subsets of each benchmark. Inference cost is measured as the ratio of total input and output character counts, normalized to the cost of batch size 4.

ing instances (over 50% of the test sets), allowing the method to intervene more often and correct failures where the vanilla evaluator struggles. Its elevated inference cost (15.2×) naturally follows from this higher frequency of selective updates.

Despite these model-specific differences, the results reveal a consistent pattern of improvement, positioning our method as a readily deployable inference strategy across diverse generative evaluators.

7 Conclusion

In this work, we study sequential test-time learning for evaluators and ask whether they can *learn while testing*. We introduce **Learning While Evaluating** (LWE), a framework that maintains an evolving meta-prompt to generate sample-specific evaluation instructions and refine itself via self-generated feedback. We further propose *Selective* LWE, which updates only on samples exhibiting self-inconsistency. Across two pairwise comparison benchmarks, we show that *Selective* LWE achieves strong evaluation performance at only a fraction of the token cost required by full sequential updates. These results demonstrate that focusing compute on confusing cases yields reliable and cost-efficient evaluation, without training or external supervision. We hope our work encourages a shift from static judges that rely on a single fixed prompt for all samples toward *learning evaluators* that improve *during* evaluation—accumulating experience as they progress through test cases, tailoring sample-specific criteria, and delivering more reliable judgments within practical inference budgets.

Limitations

Our method relies on the base capability of the underlying model, making it less effective for relatively weak models. When applied to the open model `Qwen3-VL-235B-A22B-Instruct` (Qwen team, Alibaba Cloud, 2025), which achieves competent vanilla performance, the model often fails to produce valid evaluation prompts under meta-prompt updates.

We leverage internal inconsistency to identify cases for updates. However, some "consistent but wrong" cases remain indistinguishable without access to ground-truth labels.

Finally, while our study focuses on pairwise comparisons, the proposed LWE can extend to direct assessment and other evaluation settings.

Acknowledgments

The authors thank the members of SNUMPR, especially Seongwon Cho, San Kim, and Hyeonbeom Choi, for their valuable comments and support. This work was partly supported by the IITP grants (RS-2022-II220077, RS-2022-II220113, RS-2022-II220959, RS-2022-II220871, RS-2021-II211343 (SNU AI), RS-2025-25442338 (AI Star Fellowship-SNU)) funded by the Korea government (MSIT), a grant (No. RS-2025-25453780) funded By MOTIE, a grant of Korean ARPA-H Project through the Korea Health Industry Development Institute (KHIDI) funded by the Ministry of Health & Welfare, Republic of Korea (RS-2025-25424639), the BK21 FOUR program, SNU in 2025, and the AICA GPU support program.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Anthropic. 2025. Claude sonnet 4.5 system card. Technical report, Anthropic PBC. Accessed: 2025-11-17.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, and 1 others. 2025. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–255.

Chi-Min Chan, Chunpu Xu, Jiaming Ji, Zhen Ye, Pengcheng Wen, Chunyang Jiang, Yaodong Yang, Wei Xue, Sirui Han, and Yike Guo. 2025. J1: Exploring simple test-time scaling for llm-as-a-judge. *arXiv preprint arXiv:2505.11875*.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*.

Peter Baile Chen, Yi Zhang, Dan Roth, Samuel Madden, Jacob Andreas, and Michael Cafarella. 2025. Log-augmented generation: Scaling test-time reasoning with reusable computation. *arXiv preprint arXiv:2505.14398*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning*.

John H. Flavell. 1979. Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. *American Psychologist*, 34(10):906–911.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, and 1 others. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Tenghao Huang, Kinjal Basu, Ibrahim Abdelaziz, Pavan Kapanipathi, Jonathan May, and Muhao Chen. 2025. R2d2: Remembering, replaying and dynamic decision making with a reflective agentic memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30318–30330.

Jaehun Jung, Faeze Brahman, and Yejin Choi. 2025. Trust or escalate: LLM judges with provable guarantees for human agreement. In *The Thirteenth International Conference on Learning Representations*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, and 1 others. 2024. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and 1 others. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Zae Myung Kim, Chanwoo Park, Vipul Raheja, Suin Kim, and Dongyeop Kang. 2025. Toward evaluative thinking: Meta policy optimization with evolving reward models. *Preprint*, arXiv:2504.20157.

Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2024. Benchmarking cognitive biases in large language models as evaluators. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 517–545.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and 1 others. 2023. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *Forty-first International Conference on Machine Learning*.

Seongyun Lee, Seungone Kim, Sue Park, Geewook Kim, and Minjoon Seo. 2024. Prometheus-vision: Vision-language model as a judge for fine-grained evaluation. In *Findings of the association for computational linguistics ACL 2024*, pages 11286–11315.

Lei Li, Yuancheng Wei, Zhihui Xie, Xuqing Yang, Yifan Song, Peiyi Wang, Chenxin An, Tianyu Liu, Sujian Li, Bill Yuchen Lin, Lingpeng Kong, and Qi Liu. 2025a. Vl-rewardbench: A challenging benchmark for vision-language generative reward models. In *CVPR*.

Yuran Li, Jama Hussein Mohamud, Chongren Sun, Di Wu, and Benoit Boulet. 2025b. Leveraging llms as meta-judges: A multi-agent framework for evaluating llm judgments. *arXiv preprint arXiv:2504.17087*.

Yitao Liu, Chenglei Si, Karthik Narasimhan, and Shunyu Yao. 2025. Contextual experience replay for self-improvement of language agents. *arXiv preprint arXiv:2506.06698*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2023. Openai evals repository. https://github.com/openai/evals. Accessed: 2025-10-05.

OpenAI. 2025. GPT-4.1-2025-04-14. Accessed: 2025-10-05.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Trainfing language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Junsoo Park, Seungyeon Jwa, Ren Meiying, Daeyoung Kim, and Sanghyuk Choi. 2024. Offsetbias: Leveraging debiased data for tuning evaluators. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1043–1067.

Qwen team, Alibaba Cloud. 2025. Qwen3-vl. https://github.com/QwenLM/Qwen3-VL. Accessed: 2025-10-05.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. 2025. Learning to plan reason for evaluation with thinking-llm-as-a-judge. *Preprint*, arXiv:2501.18099.

Patrick Schilcher, Dominik Karasin, Michael Schöpf, Haisam Saleh, Antonela Tommasel, and Markus Schedl. 2025. Characterizing positional bias in large language models: A multi-model evaluation of prompt order effects. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, and 1 others. 2025. Thinking vs. doing: Agents that reason by scaling test-time interaction. *arXiv preprint arXiv:2506.07976*.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2024. Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint arXiv:2406.07791*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Robert J. Sternberg. 1985. *Beyond IQ: A triarchic theory of human intelligence*. Cambridge University Press, New York.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, and 1 others. 2024. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*.

Yu Sun, Xiaolong Wang, Liu Zhuang, John Miller, Moritz Hardt, and Alexei A. Efros. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2025. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*.

Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. 2024. Foundational autoraters: Taming large language models for better automatic evaluation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17086–17105.

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

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. 2025a. Unified reward model for multimodal understanding and generation. *arXiv preprint arXiv:2503.05236*.

Yutong Wang, Pengliang Ji, Chaoqun Yang, Kaixin Li, Ming Hu, Jiaoyang Li, and Guillaume Sartoretti. 2025b. Mcts-judge: Test-time scaling in llm-as-a-judge for code correctness evaluation. *Preprint*, arXiv:2502.12468.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Jiaxin Wen, Zachary Ankner, Arushi Somani, Peter Hase, Samuel Marks, Jacob Goldman-Wetzler, Linda Petrini, Henry Sleight, Collin Burns, He He, and 1 others. 2025. Unsupervised elicitation of language models. *arXiv preprint arXiv:2506.10139*.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*.

Guowei Xu, Mert Yuksekgonul, Carlos Guestrin, and James Zou. 2025. metatextgrad: Learning to learn with language models as optimizers. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Michihiro Yasunaga, Luke Zettlemoyer, and Marjan Ghazvininejad. 2025. Multimodal rewardbench: Holistic evaluation of reward models for vision language models. *Preprint*, arXiv:2502.14191.

Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, and 1 others. 2025. Self-generated critiques boost reward modeling for language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11499–11514.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616.

Yuhang Zang, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Ziyu Liu, Shengyuan Ding, Shenxi Wu, Yubo Ma, Haodong Duan, Wenwei Zhang, and 1 others. 2025. Internlm-xcomposer2. 5-reward: A simple yet effective multi-modal reward model. *arXiv preprint arXiv:2501.12368*.

Tom Zehle, Timo Heiß, Moritz Schlager, Matthias Aßenmacher, and Matthias Feurer. 2026. promptolution: A unified, modular framework for prompt optimization. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 3: System Demonstrations)*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2024a. Aflow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.

Zheng Zhang, Fan Yang, Ziyan Jiang, Zheng Chen, Zhengyang Zhao, Chengyuan Ma, Liang Zhao, and Yang Liu. 2024b. Position-aware parameter efficient fine-tuning approach for reducing positional bias in llms. *arXiv preprint arXiv:2404.01430*.

Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. 2025. One token to fool llm-as-a-judge. *arXiv preprint arXiv:2507.08794*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Appendix

A Implementation Details

For meta-prompt updates, only one-sided judgments are utilized. The swapped counterparts are excluded to avoid nearly doubling the context length and to maintain inference efficiency.

As evaluation progresses, the meta-prompt accumulates insights and can become excessively long. Empirically, we observe that beyond a certain length, additional content becomes redundant and yields only marginal performance gains. To address this, we periodically summarize the meta-prompt once it exceeds a predefined length threshold (10,000 characters in our experiments). As shown in Table 3, our preliminary analysis suggests that summarization does not harm evaluation accuracy.

For Multimodal RewardBench, we subsampled 1,000 cases from the 4,711 samples, excluding the 500 samples from the "Hateful Memes" subset. This subset was included in the original paper but not provided in the benchmark's official implementation, so we used the data available in the official release.

For a fair comparison, Dynamic Cheatsheet, which is fully sequential, was evaluated using the same test-case order as LWE (run0 in Appendix H).

All experiments with gpt-4.1 (Table 1), gemini-2.5-pro, and claude-sonnet-4.5 (Table 2) were conducted with temperature set to 0, except for the majority-voting baseline for gpt-4.1, which used temperature 0.7.

B `TextGrad` Implementation Details

Since TextGrad requires a validation set for prompt optimization, we used 10 samples for training and 10 samples for validation on VLRewardBench. The reported results on VLRewardBench are based on the remaining 1,227 samples (out of 1,247), as the entire test set was used for other methods.

For Multimodal RewardBench, from the remaining 3,711 samples, we randomly selected 40 examples and split them into 20 for training and 20 for validation.

Following the default configuration used in the official TextGrad examples, we trained for 3 epochs with a batch size of 3. We then selected the prompts that achieved the highest validation scores for each benchmark and applied the corresponding final prompts uniformly to their test sets.

Benchmark	Setting	Acc.
VL (n=300)	w/o summarization	0.710
VL (n=300)	w/ summarization	0.733
VL (n=1,247)	w/ summarization (Table 1)	**0.745**
MM (n=400)	w/o summarization	0.790
MM (n=400)	w/ summarization	0.793
MM (n=1,000)	w/ summarization (Table 1)	**0.799**

Table 3: **Preliminary analysis of meta-prompt summarization.** As a preliminary check, we tested LWE with gpt-4.1 with and without summarization on subsets of VLRewardBench (VL) and MMRewardBench (MM), confirming no accuracy degradation from summarization.

Algorithm 3 `Sample-Specific Prompt`

Input: A base LLM `LLM`, a test set D_{test}, and an initial meta-prompt M_0
Output: A set of evaluated results S

$S \leftarrow \emptyset$ ▷ Initialize a set of evaluated results.
for $x \in D_{\text{test}}$ **do**
  $P \leftarrow \texttt{BuildEvalPrompt}_{\texttt{LLM}}(M_0, x)$
  $y \leftarrow \texttt{Judge}_{\texttt{LLM}}(P, x)$
  $S \leftarrow S \cup \{(x, y)\}$
end for
return S

Algorithm 4 `Vanilla`

Input: A base LLM `LLM`, a test set D_{test}, and a vanilla prompt P
Output: A set of evaluated results S

$S \leftarrow \emptyset$ ▷ Initialize a set of evaluation results.
for $x \in D_{\text{test}}$ **do**
  $y \leftarrow \texttt{Judge}_{\texttt{LLM}}(P, x)$
  $S \leftarrow S \cup \{(x, y)\}$
end for
return S

For TextGrad, the relative inference cost includes both test-time inference and the additional inference required during its prompt optimization stage.

C Pseudocode of Evaluation Methods

Algorithms 3 and 4 present the pseudocode of the Sample-Specific Prompt and Vanilla baselines. All procedures (`BuildEvalPrompt`, and `Judge`) are implemented through LLM prompting.

Figure 8: Image of the test case `hallucination_pair-4608` from VLRewardBench.

D Function for Extracting Answers

```
def extract_judgment(judgment):
    if "[[A]]" in judgment and "[[B]]"
        in judgment:
        return "Not judged in the proper
            format.  [[A,B]]"
    if "[[A]]" in judgment:
        return "A"
    elif "[[B]]" in judgment:
        return "B"
    elif "[A]" in judgment:
        return "A"
    elif "[B]" in judgment:
        return "B"
    else:
        return "Not judged in the proper
            format."
```

Code 1: A function used to extract judgments from model-generated responses. This version slightly modifies the original Multimodal RewardBench code, adding stricter formatting requirements.

E Prompt Templates

Figures 9–15 are prompt templates that are used for our experiments.

F Examples from the Vanilla Baseline and *Selective* LWE

Figures 16–20 illustrate the actual prompts generated while evaluating a test case `hallucination_pair-4608` from VLRewardBench. The corresponding input image is shown in Figure 8. While the Vanilla baseline incorrectly selects response A, *Selective* LWE correctly identifies response B as the better answer.

G Full Meta-Prompts from Figure 5

Figures 21 and 22 present the full meta-prompts corresponding to Figure 5. These examples illustrate how the meta-prompt is refined after a single update step, transitioning from an earlier version (Figure 21) to the updated version (Figure 22).

H Test Case Ordering Results

Tables 7–10 show the complete results of LWE and *Selective* LWE with gpt-4.1 on each benchmark with three random orders of test cases. Tables 11–14 provide the corresponding *Selective* LWE results using gemini-2.5-pro and claude-sonnet-4.5 on each benchmark.

I LWE on Inconsistent *vs*. Consistent Subsets

We investigate why LWE achieves higher accuracy than *Selective* LWE on VLRewardBench. An inspection of the inconsistent and consistent subsets shows that VLRewardBench contains a relatively large number of vanilla-consistent but incorrect examples (999 - 659 = 340 examples) compared to that of MMRewardBench (863 - 746 = 117 examples) (Table 5). We hypothesize that updates on these cases provide additional supervisory signal for the meta-prompt, which may contribute to the higher accuracy observed for LWE.

However, this effect appears to impact accuracy only, not pair accuracy. Across both benchmarks, *Selective* LWE consistently achieves higher pair accuracy, which reliably measures the evaluator's ability and robustness to position bias. Moreover, LWE exhibits substantially larger improvements on the vanilla-inconsistent subsets (Table 4) compared to the consistent subsets (Table 5). Taken together, these patterns suggest that, despite the presence of consistent but wrong cases, the vanilla inconsistency signal remains a useful and reliable criterion for determining when updates should be applied.

J Scalability to Large-Scale Distributed Evaluation

While our experiments use benchmark-scale test sets, real-world evaluation often spans millions of cases across parallel workers. *Selective* LWE already mitigates this bottleneck by focusing additional compute only on inconsistent cases, after parallelizable consistency checks. However, multi-worker distributed settings require additional strategies. We discuss two practical extensions of LWE to such settings.

A *warm-up-then-broadcast* strategy would run *Selective* LWE on a representative subset to obtain a mature meta-prompt, then broadcast it to all workers as an improved fixed prompt for parallel evaluation of the remaining cases. Since LWE is robust

Method	VLRewardBench	MMRewardBench
Vanilla	0.504 (125/248)	0.453 (62/137)
Selective LWE	0.762 (189/248)	0.672 (92/137)

Table 4: **Accuracy on the inconsistent subsets.** *Selective* LWE leads to more accurate judgments. Results correspond to a single run (run0) from Tables 9 and 10.

Method	VLRewardBench	MMRewardBench
Vanilla	0.660 (659/999)	0.864 (746/863)
LWE	0.734 (733/999)	0.846 (730/863)

Table 5: **Accuracy on the consistent subsets.** As an ablation, we apply LWE to consistent examples; the resulting performance gains are modest compared to those observed on inconsistent cases.

Method	VLRewardBench			MMRewardBench		
	Acc.	Cons.	PairAcc.	Acc.	Cons.	PairAcc.
TextGrad*	**0.730**	0.749	0.615	0.821	0.836	0.741
GEPA*	0.682	0.807	0.586	**0.861**	0.903	0.805
Selective LWE	0.676	**0.940**	**0.648**	0.836	**0.947**	**0.808**

Table 6: **Comparison with prompt optimization methods with gpt-4.1.** TextGrad and GEPA (*) require gold-labeled validation data, while *Selective* LWE uses no labels at any stage.

to sample ordering (Appendix H), warm-up on an arbitrary subset is expected to yield a similarly effective meta-prompt. A *sequential-parallel hybrid*, where workers periodically synchronize their local meta-prompts into a shared repository, is also possible, preserving sequential learning while scaling across workers.

K Comparison with Previous Works

Table 15 summarizes the comparison with previous studies.

Beyond TextGrad, we additionally compare against GEPA (Agrawal et al., 2025), a recent reflective prompt optimizer integrated in DSPy (Khattab et al., 2024). We use the same train, validation, and test splits as TextGrad (Appendix B), and apply the resulting optimized prompts uniformly to the test sets. We use the `auto="medium"` configuration of `dspy.GEPA` with gpt-4.1 as both the task and reflection model [2].

The relative inference cost of GEPA is 4.6×, which includes the one-time prompt-optimization (compile) cost; accounting for test-time inference alone, GEPA operates at 3.1×.

[2] https://github.com/gepa-ai/gepa

L Data Licensing

We use publicly available datasets under research-only and permissive licenses. VLRewardBench is released for research use only. Multimodal RewardBench is licensed under the Creative Commons Attribution-NonCommercial (CC BY-NC)[3], Copyright (c) Meta Platforms, Inc. and affiliates. TextGrad and Dynamic Cheatsheet are released under the MIT License[4]. All datasets and codebases were used solely for non-commercial, academic research in compliance with their respective licenses.

[3] https://creativecommons.org/licenses/by-nc/4.0/

[4] https://opensource.org/licenses/mit

LWE	run0	run1	run2	mean	std.
Acc.	0.746	0.713	0.776	**0.745**	**0.031**
Cons.	0.808	0.789	0.818	**0.805**	**0.015**
PairAcc.	0.650	0.610	0.677	**0.646**	**0.033**

Table 7: Performance of LWE under different orderings of VLRewardBench. The table reports the mean and standard deviation across three independent runs.

LWE	run0	run1	run2	mean	std.
Acc.	0.803	0.806	0.789	**0.799**	**0.009**
Cons.	0.839	0.850	0.850	**0.846**	**0.006**
PairAcc.	0.727	0.727	0.727	**0.727**	**0.000**

Table 8: Performance of LWE under different orderings of MMRewardBench. The table reports the mean and standard deviation across three independent runs.

Selective LWE	run0	run1	run2	mean	std.
Acc.	0.680	0.674	0.674	**0.676**	**0.003**
Cons.	0.945	0.938	0.937	**0.940**	**0.004**
PairAcc.	0.653	0.649	0.642	**0.648**	**0.006**

Table 9: Performance of *Selective* LWE under different orderings of VLRewardBench. The table reports the mean and standard deviation across three independent runs.

Selective LWE	run0	run1	run2	mean	std.
Acc.	0.838	0.837	0.832	**0.836**	**0.003**
Cons.	0.940	0.954	0.946	**0.947**	**0.007**
PairAcc.	0.806	0.813	0.805	**0.808**	**0.004**

Table 10: Performance of *Selective* LWE under different orderings of MMRewardBench. The table reports the mean and standard deviation across three independent runs.

Selective LWE	run0	run1	run2	mean	std.
Acc.	0.770	0.767	0.767	**0.768**	**0.001**
Cons.	0.955	0.964	0.946	**0.955**	**0.009**
PairAcc.	0.746	0.748	0.739	**0.744**	**0.005**

Table 11: Performance of *Selective* LWE under different orderings of VLRewardBench with gemini-2.5-pro. The table reports the mean and standard deviation across three independent runs.

Selective LWE	run0	run1	run2	mean	std.
Acc.	0.865	0.862	0.868	**0.865**	**0.003**
Cons.	0.975	0.959	0.972	**0.969**	**0.009**
PairAcc.	0.857	0.846	0.853	**0.852**	**0.006**

Table 12: Performance of *Selective* LWE under different orderings of MMRewardBench with gemini-2.5-pro. The table reports the mean and standard deviation across three independent runs.

Selective LWE	run0	run1	run2	mean	std.
Acc.	0.705	0.704	0.701	**0.703**	**0.002**
Cons.	0.878	0.885	0.881	**0.881**	**0.003**
PairAcc.	0.646	0.647	0.650	**0.648**	**0.002**

Table 13: Performance of *Selective* LWE under different orderings of VLRewardBench with claude-sonnet-4.5. The table reports the mean and standard deviation across three independent runs.

Selective LWE	run0	run1	run2	mean	std.
Acc.	0.841	0.837	0.791	**0.823**	**0.028**
Cons.	0.912	0.893	0.831	**0.879**	**0.042**
PairAcc.	0.795	0.782	0.736	**0.771**	**0.031**

Table 14: Performance of *Selective* LWE under different orderings of MMRewardBench with claude-sonnet-4.5. The table reports the mean and standard deviation across three independent runs.

Method	Updates Text	Selectively Updates Text	Tests Sequentially	Explicitly Generates Sample-specific Prompts
TextGrad / DSPy / GEPA / promptolution	on a *validation* set	✗	✗	✗
Dynamic Cheatsheet	on a *test* set	✗	✓	✗
Selective LWE	on a *test* set	✓	partially ✓	✓

Table 15: Comparison with previous works.

```
Please act as an impartial judge and evaluate the quality of the responses provided by two
AI assistants to the user question displayed below. You should choose the assistant that
follows the user's instructions and answers the user's question better. Your evaluation
should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and
level of detail of their responses. Begin your evaluation by comparing the two responses and
provide a short explanation. Avoid any position biases and ensure that the order in which
the responses were presented does not influence your decision. Do not allow the length of
the responses to influence your evaluation. Do not favor certain names of the assistants.
Be as objective as possible. After providing your explanation, output your final verdict by
strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better.

[User Question]
{question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]
```

Figure 9: `Vanilla` evaluation prompt. We follow the prompt of (Yasunaga et al., 2025).

```
Please act as an impartial judge and evaluate the quality of the responses provided by two
AI assistants to the user question displayed below.  You should choose the assistant that
follows the user's instructions and answers the user's question better.  Your evaluation
should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and
level of detail of their responses. Begin your evaluation by comparing the two responses and
provide a short explanation.  Avoid any position biases and ensure that the order in which
the responses were presented does not influence your decision.  Do not allow the length of
the responses to influence your evaluation.  Do not favor certain names of the assistants.
Be as objective as possible.  After providing your explanation, output your final verdict by
strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better.

[User Question]
{question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]

Please reason in a step-by-step manner before giving a response. (You now have an opportunity
to reason privately; your next response will not be evaluated.)
```

Figure 10: `Chain-of-Thought (CoT)` evaluation prompt. We adopt the Chain-of-Thought (CoT) evaluation prompt from the OpenAI Evals repository (OpenAI, 2023).

```
Generate a prompt for an evaluator that includes example-specific evaluation criteria and
step-by-step evaluation procedures. The evaluator will base their evaluation on the prompt and
it will help the evaluator accurately judge the correctness and reasoning quality of given
responses, and identify which answer is better.

Generated evaluation prompt MUST include
- evaluation criteria specific for the given example
- evaluation steps specific for the given example to induce an accurate and reliable judgment.

Additionally, the evaluation prompt MUST instruct the evaluator that their final judgment must
be expressed only in one of the following two formats:
'[[A]]' or '[[B]]'.

Output exactly ONLY THE PROMPT.
```

Figure 11: Initial meta prompt. We use this prompt as the static meta prompt in `Sample-Specific Prompt` and the initial meta prompt in LWE and *Selective* LWE (`BuildEvalPrompt`).

```
You will be given:
An evaluation prompt — the instructions or criteria for judging
A given example — a case where you have already made a judgment based on that prompt

Your task:
Evaluate the correctness and reasoning quality of the given judgment.
Based on the inspection, update your [[Learned tips for future prompt optimization]] based on
the current "meta_prompt".

Your output must have three parts:
[[[Score (1-5)]]]
5 = Judgment is entirely correct, thorough, and follows the evaluation prompt exactly
4 = Mostly correct, with only minor reasoning gaps
3 = Partially correct, but with noticeable flaws or missing justification
2 = Largely incorrect, due to poor reasoning or serious omissions
1 = Fundamentally wrong, fails to follow the evaluation prompt

[[[Binary label]]]
"Absolutely confident the judgment is correct" — if reasoning is strong, evidence-based, and
follows the criteria without major gaps
"Not sure" — if there are reasoning flaws, possible alternative interpretations, or insufficient
evidence

[[[Learned tips for future prompt optimization]]]
After reading the given example and its judgment, identify specific points to improve in the
reasoning, coverage, or clarity.
Reflect on these findings to propose concrete adjustments to future evaluation prompts (e.g.,
clarifying ambiguous criteria, adding explicit checks for evidence, requiring certain logical
structures)
Capture any patterns that might cause repeated errors or inconsistencies so they can be addressed
in later prompts.
You might refer to the current meta prompt and identify additional essential tips that are not
addressed in the current meta prompt.
```

Figure 12: Prompt for feedback generation (Feedback) (Part 1).

```
**Evaluation Steps (You must follow these before scoring)**:
Step 1: Restate the evaluation criteria in your own words to ensure you understand them.
Step 2: Read the given example judgment carefully and identify its main claim(s) and supporting reasoning.
Step 3: Compare the judgment's reasoning with the evaluation criteria; look for matches, missing points, contradictions, or misinterpretations.
Step 4: Inspect for logical fallacies, including:
- Unsupported generalizations
- False cause (confusing correlation with causation)
- Strawman misrepresentations of the criteria
- Circular reasoning
- Irrelevant evidence
Step 5: Pinpoint exact areas for improvement in the judgment's reasoning or evidence use.
Step 6: Assign a score based on the completeness and accuracy of the reasoning relative to the prompt.
Step 7: Decide on the binary confidence label.
Step 8: Reflect on how these identified weaknesses could be prevented through clearer or more detailed future evaluation prompts, and write down 1-3 learned tips.

[[[The Start of Current Meta Prompt]]]
{meta_prompt}
[[[The End of Current Meta Prompt]]]

[[[The Start of Evaluation Prompt & Judgment]]]
{evaluation_prompt}

[Start of Judgment]
{judgment}
[The End of Judgment]
[[[The End of Evaluation Prompt & Judgment]]]

Strictly follow the **Output format**:
{{"score": score, "label": label, "learned tips": tips, "reasoning": reasoning_or_explanation}}
```

Figure 12: Prompt for feedback generation (Feedback) (Part 2).

```
Optimize your current meta prompt. You need to refer to feedback. You need to improve the
meta prompt. The goal of meta prompt is to generate an evaluation prompt that helps more
accurate and reliable judgments. A good evaluation prompt needs better example-specific
evaluation rubrics and evaluation steps. Make sure not to repeat any tips that are
already provided. Instead, write only additional tips that are general, reusable, and useful
for evaluation. Focus on refining and improving the quality of your learned tips and experiences.

[[[The Start of Current Meta Prompt]]]
{meta_prompt}
[[[The End of Current Meta Prompt]]]

[[[The Start of Batch examples & Meta Feedback]]]
{batch}
[[[The End of Batch examples & Meta Feedback]]]

Again, optimize your current meta prompt based on the feedback.
[[[Optimized Meta Prompt]]]
```

Figure 13: Prompt template for refining a meta prompt (`RefineMetaPrompt`). Each "batch" includes the current evaluation prompts, examples, judgments, and feedback.

```
[User Question]
{question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]
```

Figure 14: Example placeholder. The above placeholder with an example is appended at the end of each evaluation prompt to ensure stable generation.

```
You are a helpful assistant that summarizes and shortens text. Your current meta prompt is too
long.
[[[The Start of Current Meta Prompt]]]
{meta_prompt}
[[[The End of Current Meta Prompt]]]

Condense the meta prompt so it is roughly HALF as long, but still structured and keeping the
key details.
ONLY output the shortened meta prompt.
```

Figure 15: Summarization prompt. As described in Appendix A, when a meta prompt exceeds a predefined length, it is summarized by the model itself.

Example with the vanilla evaluation prompt

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: `"[[A]]"` if assistant A is better, `"[[B]]"` if assistant B is better.

`[User Question]`
Describe this image in detail.

`[The Start of Assistant A's Answer]`
The image shows a group of friends sitting on a couch in a living room, enjoying each other's company. There are five people in total, including a pretty young lady and four men. They appear to be dressed in casual or pajama attire, suggesting a relaxed atmosphere. Snacks or food can be seen on the table, adding to the cozy vibe. As they share a funny story, some drinks can also be observed in the scene, with three cups and a bottle placed around the room, likely adding to the merriment. On a table, there are two laptops and a mouse, hinting that the group may be engaged in some kind of electronic activity or entertainment as well.
`[The End of Assistant A's Answer]`

`[The Start of Assistant B's Answer]`
The image shows a group of friends sitting on a couch in a living room, enjoying each other's company. There are five people in total, including a pretty young lady and four men. They appear to be dressed in casual or pajama attire, suggesting a relaxed atmosphere.

As they share a funny story, some drinks can be observed in the scene, with three cups and a bottle placed around the room, likely adding to the merriment. On a table, there are two laptops and a mouse, hinting that the group may be engaged in some kind of electronic activity or entertainment as well.
`[The End of Assistant B's Answer]`

Figure 16: Example with the vanilla evaluation prompt used for pairwise judgment.

Example of a judgment with the vanilla evaluation prompt

Assistant A and Assistant B provide very similar descriptions of the image, both noting the group of five friends, the casual or pajama attire, the presence of drinks, laptops, and the relaxed, social atmosphere. However, Assistant A's response is slightly more detailed, mentioning the snacks or food on the table and describing the overall vibe as "cozy." This adds a bit more depth and context to the scene, making the description more complete and helpful. Both responses are accurate and relevant, but Assistant A's answer offers a touch more detail and completeness.

`[[A]]`

Figure 17: Example of a vanilla judgment by the prompt in Figure 16.

Example of a meta prompt

[[[Optimized Meta Prompt]]]

Meta Prompt for Generating High-Quality Evaluation Prompts

Generate an evaluation prompt for an evaluator that produces more accurate and reliable judgments by incorporating example-specific rubrics and detailed evaluation steps. The evaluation prompt you generate MUST:

- Include evaluation criteria tailored to the specific example, focusing on observable evidence, internal consistency, and avoidance of unsupported speculation or irrelevant details.
- Provide step-by-step evaluation procedures that guide the evaluator to systematically compare each answer against the example, cross-checking all claims with the available evidence and justifying their choice with reference to specific elements.
- Instruct the evaluator to penalize both the omission of significant details and the inclusion of non-existent or speculative information.
- Emphasize the importance of directness, clarity, and relevance, and require explicit reference to the evidence supporting their decision.
- Require that the evaluator's final judgment be expressed only in one of the following two formats: '[[A]]' or '[[B]]'.

In addition, based on learned experience and feedback, incorporate these further general, reusable tips to improve evaluation quality. Do NOT repeat any tips already provided above; instead, add only new, generalizable, and actionable tips that further refine evaluation quality:

Figure 18: Example of a meta prompt (Part 1).

Additional Tips for Further Improvement:

- Require evaluators to **explicitly restate the observable evidence at the start of their evaluation** to anchor all subsequent judgments in the provided data.
- Instruct evaluators to **explicitly check for and penalize misinterpretation of labels, diagrams, or visual cues** (e.g., confusing radius and diameter, or misreading image features), as these are common sources of error.
- Require evaluators to **explicitly verify that the selected answer is at least partially relevant to the actual example and prompt**, not just internally consistent or well-reasoned.
- Instruct evaluators to **call out and penalize even minor speculative or fabricated details in both answers**, not just omissions, and to reward concise answers that avoid unsupported elaborations.
- Add a step for evaluators to **explicitly check for off-topic or irrelevant answers**; if both answers are off-topic, reward the one that at least attempts to address the prompt or acknowledge when neither is correct.
- Require evaluators to **briefly justify why the rejected answer is incorrect or less preferred**, especially when visual or factual evidence is available.
- Encourage evaluators to **distinguish between errors due to plausible but unsupported assumptions and outright fabrications**, and to penalize accordingly.
- Instruct evaluators to **reward transparent, evidence-limited reasoning**—for example, when an answer acknowledges the limits of the available evidence or the possibility that neither option is fully correct.
- Remind evaluators to **explicitly reference the evidence in their reasoning, not just in the final answer**, and to ensure that all preferences and penalizations are justified by direct reference to the example.

Meta-Reminders:
- Do not judge the example contents themselves.
- Do not include or reference the example contents in your evaluation prompt.
- Follow these steps and criteria strictly for every evaluation.

Goal:
The resulting evaluation prompt should enable evaluators to make more accurate, reliable, and evidence-grounded judgments, with explicit, example-specific rubrics and procedures, and with all preferences and penalizations justified by direct reference to the evidence.

Figure 18: Example of a meta prompt (Part 2).

Example of a sample-specific evaluation prompt generated by a meta prompt

"**Evaluation Prompt Template**

Step 1: Restate the Observable Evidence
Begin your evaluation by explicitly restating the observable evidence from the example. List the key visual or factual elements that are directly present and verifiable.

Step 2: Apply Example-Specific Evaluation Criteria
Evaluate each answer using the following criteria, tailored to the example:
- **Accuracy and Completeness:** Does the answer accurately describe all significant observable details? Are any important elements omitted or misrepresented?
- **Internal Consistency:** Is the answer logically consistent within itself, without contradictions?
- **Relevance:** Does the answer focus only on details that are directly observable and relevant to the example, avoiding speculation or irrelevant information?
- **Evidence-Based Reasoning:** Are all claims in the answer directly supported by the observable evidence? Is there any unsupported speculation or fabrication?
- **Clarity and Directness:** Is the answer clear, concise, and free from unnecessary elaboration or ambiguity?

Step 3: Systematic Comparison and Cross-Checking
For each answer:
- Systematically compare every claim or detail against the observable evidence.
- Explicitly check for and penalize any misinterpretation of visual cues, labels, or features (e.g., miscounting objects, misidentifying items).
- Penalize both the omission of significant details and the inclusion of non-existent or speculative information.
- Explicitly verify that the answer is at least partially relevant to the actual example and prompt, not just internally consistent.
- Call out and penalize even minor speculative or fabricated details, and reward concise answers that avoid unsupported elaborations.
- Check for off-topic or irrelevant content; if both answers are off-topic, reward the one that at least attempts to address the prompt or acknowledge when neither is correct.

Step 4: Justification and Preference
- For the answer you prefer, justify your choice by referencing specific elements of the observable evidence and explaining how the answer meets the criteria above.
- Briefly justify why the rejected answer is incorrect or less preferred, especially when visual or factual evidence is available.
- Distinguish between errors due to plausible but unsupported assumptions and outright fabrications, and penalize accordingly.
- Reward transparent, evidence-limited reasoning (e.g., when an answer acknowledges the limits of the available evidence or the possibility that neither option is fully correct).
- Ensure that all preferences and penalizations are justified by direct reference to the evidence, not just in the final answer but throughout your reasoning.

Figure 19: Example of a sample-specific evaluation prompt generated by the meta prompt in Figure 18 (Part 1).

Step 5: Final Judgment
Express your final judgment in one of the following two formats only:
- [[A]]
- [[B]]

Meta-Reminders:
- Do not judge the example contents themselves.
- Do not include or reference the example contents in your evaluation prompt.
- Follow these steps and criteria strictly for every evaluation.

[User Question]
Describe this image in detail.

[The Start of Assistant A's Answer]
The image shows a group of friends sitting on a couch in a living room, enjoying each other's company. There are five people in total, including a pretty young lady and four men. They appear to be dressed in casual or pajama attire, suggesting a relaxed atmosphere. Snacks or food can be seen on the table, adding to the cozy vibe. As they share a funny story, some drinks can also be observed in the scene, with three cups and a bottle placed around the room, likely adding to the merriment. On a table, there are two laptops and a mouse, hinting that the group may be engaged in some kind of electronic activity or entertainment as well.
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
The image shows a group of friends sitting on a couch in a living room, enjoying each other's company. There are five people in total, including a pretty young lady and four men. They appear to be dressed in casual or pajama attire, suggesting a relaxed atmosphere.

As they share a funny story, some drinks can be observed in the scene, with three cups and a bottle placed around the room, likely adding to the merriment. On a table, there are two laptops and a mouse, hinting that the group may be engaged in some kind of electronic activity or entertainment as well.
[The End of Assistant B's Answer]

Figure 19: Example of a sample-specific evaluation prompt generated by the meta prompt in Figure 18 (Part 2).

Example of a judgment generated by a sample-specific evaluation prompt

Step 1: Restate the Observable Evidence
- Five people are present: four sitting on a couch (three men, one woman) and one person (gender unclear from this angle) sitting on the floor with their back to the camera.
- The people are dressed in casual or pajama-like attire.
- The setting is a living room with a couch, a wooden coffee table, and a rug.
- On the coffee table: two open laptops, a computer mouse, a candle, some cables, and a drink bottle.
- Several people are holding red cups; one person on the floor has a blue can or bottle.
- The group appears to be smiling and engaged in conversation or laughter.
- The room has a relaxed, informal atmosphere.

Step 2: Apply Example-Specific Evaluation Criteria
- **Accuracy and Completeness:** Both answers mention five people, casual/pajama attire, a living room, laptops, a mouse, drinks, and a relaxed atmosphere. Assistant A uniquely mentions snacks/food on the table, which is not clearly visible in the image. Both mention a "pretty young lady," which is subjective and not strictly observable.
- **Internal Consistency:** Both answers are internally consistent.
- **Relevance:** Both focus on observable details, but both include some speculation (e.g., "likely adding to the merriment," "may be engaged in some kind of electronic activity").
- **Evidence-Based Reasoning:** Both answers are mostly evidence-based, but A includes a fabricated detail about snacks/food, which is not supported by the image.
- **Clarity and Directness:** Both are clear and concise, but A is slightly more verbose due to the extra (unsupported) detail.

Step 3: Systematic Comparison and Cross-Checking
- Both answers correctly identify the number of people, the general attire, the presence of laptops, and the drinks.
- Both mention a "pretty young lady," which is subjective and not strictly evidence-based.
- Both mention drinks, cups, and a bottle, which are visible.
- Both mention the possibility of electronic activity, which is a reasonable inference from the laptops.
- Assistant A adds "snacks or food can be seen on the table," which is not supported by the image—there is a candle, but no visible food or snacks.
- Both answers are relevant to the prompt, but A includes a minor fabrication.

Step 4: Justification and Preference
- **Preferred Answer:** Assistant B is preferred because it avoids the unsupported claim about snacks/food, sticking more closely to the observable evidence. While both answers include some minor speculation and subjective language, B does not fabricate any details about the scene.
- **Rejected Answer:** Assistant A is less preferred due to the inclusion of a fabricated detail (snacks/-food) that is not visible in the image. This is a minor but clear deviation from evidence-based reasoning.

Step 5: Final Judgment
[[B]]

Figure 20: Example of a judgment generated by the sample-specific evaluation prompt in Figure 19.

Example of a meta prompt before an update

[[[Optimized Meta Prompt]]]

Generate an evaluation prompt for an evaluator that includes example-specific evaluation criteria and step-by-step evaluation procedures. The evaluator will use this prompt to accurately judge the correctness and reasoning quality of given responses, and to identify which answer is better.

The evaluation prompt you generate MUST include:
- Evaluation criteria tailored to the specific example, covering all aspects required for a thorough and precise judgment.
- Step-by-step evaluation procedures that are specific to the example and guide the evaluator to a reliable and accurate decision.

Additionally, the evaluation prompt MUST instruct the evaluator that their final judgment must be expressed only in one of the following two formats:
'[[A]]' or '[[B]]'.

When generating the evaluation prompt, incorporate the following additional tips to further improve accuracy and reliability (do NOT repeat tips already provided in the current meta prompt or previous feedback):

Additional Tips:
- Require evaluators to explicitly verify the presence or absence of each described or claimed feature, step, or detail in the source material (e.g., image, code output, data), and to document any mismatches.
- Instruct evaluators to systematically check for calculation or logical errors in each answer, not just the reasoning structure, and to penalize answers with such errors even if the explanation appears logical.
- Encourage evaluators to reward answers that acknowledge ambiguity or uncertainty when the evidence is inconclusive, rather than making unsupported definitive claims.
- Require evaluators to compare all differences between answers, including minor or subtle distinctions, and to justify whether these differences materially affect the evaluation outcome.
- Instruct evaluators to reference authoritative sources or documentation (when applicable) to resolve disputes about standards, definitions, or expected outputs.
- Prompt evaluators to explicitly state any assumptions made during the evaluation and to consider whether those assumptions are justified by the evidence or context.
- Encourage evaluators to identify and explain any gaps in reasoning, missing steps, or lack of explicit justification in the answers, and to penalize accordingly.
- Require evaluators to ensure that their final justification is comprehensive, addressing every evaluation criterion and step, and not omitting any relevant aspect of the comparison.

Output exactly ONLY THE PROMPT.

Figure 21: Example of a meta prompt before an update from Figure 5.

Example of a meta prompt after an update

Generate an evaluation prompt for an evaluator that includes example-specific evaluation criteria and step-by-step evaluation procedures. The evaluator will use this prompt to accurately judge the correctness and reasoning quality of given responses, and to identify which answer is better.

The evaluation prompt you generate MUST include:
- Evaluation criteria tailored to the specific example, covering all aspects required for a thorough and precise judgment.
- Step-by-step evaluation procedures that are specific to the example and guide the evaluator to a reliable and accurate decision.

Additionally, the evaluation prompt MUST instruct the evaluator that their final judgment must be expressed only in one of the following two formats:
'[[A]]' or '[[B]]'.

When generating the evaluation prompt, incorporate the following additional tips to further improve accuracy and reliability (do NOT repeat tips already provided in the current meta prompt or previous feedback):

Additional Tips:
- Require evaluators to explicitly verify the accuracy of arrangement, sequence, or spatial relationships described in each answer, especially when such details are visually or contextually important.
- Instruct evaluators to penalize both overstatements (e.g., exaggerating color, quantity, or significance) and omissions (e.g., failing to mention a visible but relevant feature), and to weigh the impact of these errors on the overall evaluation.
- Encourage evaluators to note when both answers contain errors or inaccuracies, and to justify the relative impact of those errors on the final decision, rather than defaulting to the answer with fewer mistakes.
- Prompt evaluators to reward answers that avoid unnecessary or unsupported inferences, especially when the evidence is ambiguous or incomplete.
- Require evaluators to check for and penalize the inclusion of details that are not supported by the source material, even if they do not directly contradict it.
- Instruct evaluators to briefly justify why the chosen answer best matches the evidence, even in cases where the answer is a single word or letter.
- Encourage evaluators to cite or reference authoritative sources, standards, or guidelines when resolving disputes about terminology, definitions, or expected outputs, and to document the source used.
- Remind evaluators to consider the clarity and explicitness of reasoning, rewarding answers that make their logic and assumptions transparent and penalizing those that leave reasoning steps implicit or unclear.
- Require evaluators to ensure that their final justification is not only comprehensive but also balanced, addressing both the strengths and weaknesses of each answer in relation to every evaluation criterion.

Output exactly ONLY THE PROMPT.

Figure 22: Example of a meta prompt after an update from Figure 5.