FROM JUDGMENT QUALITY TO DOWNSTREAM UTILITY: RETHINKING LLM-AS-A-JUDGE FOR OPEN-ENDED TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-as-a-Judge is increasingly used to evaluate policy responses on open-ended tasks that lack ground-truth answers. Existing work often directly converts the resulting judgments into reward signals for policy training, paying limited attention to intrinsic judgment quality and largely restricting the use of Judges to training-time supervision. We systematically investigate judgment quality and downstream utility by examining both how judgments are elicited and how they are used. For judgment elicitation, we vary the Judge protocol along three dimensions: verdict granularity, critique usage, and evaluation batching. For judgment usage, beyond policy training, we extend Judge to test-time inference through Best-of-N selection, Judge-guided revision, and beam search. We find that, (i) Surprisingly, judgment quality and downstream utility do not always align. (ii) Judge protocol design substantially affects both intrinsic judgment quality and downstream utility. (iii) Judge guidance effectively converts test-time compute into performance gains, with benefits varying across inference strategies. Our results call for a multifaceted evaluation of LLM Judges on open-ended tasks, encompassing intrinsic judgment quality, and downstream utility.

1 INTRODUCTION

LLM-as-a-Judge has become a widely adopted evaluation paradigm for open-ended question-answering tasks, which typically lack verifiable ground-truth answers. Under this paradigm, a Judge assesses policy[1] responses against predefined rubric criteria and produces judgments indicating how well each criterion is satisfied. These judgments provide an evaluation signal that can be further converted into numerical rewards for policy optimization. Recent studies (Huang et al., 2026; Zhang et al., 2026; Wu et al., 2026; Wang et al., 2026b) increasingly use rubric-based judgments as supervision for reinforcement learning (RL) optimization. Gunjal et al. (2026) trains policies with GRPO on open-ended medical and scientific tasks using rubric-based rewards derived from a Judge's verdicts. Zhang et al. (2026) convert the Judge's verdict into fine-grained rewards for policy optimization.

Despite the widespread adoption of rubric-based LLM judges, two key limitations remain. (i) *Limited downstream utility.* On open-ended tasks, Judges are primarily used to produce judgments that are converted into rewards for policy training (Zhou et al., 2025; Gunjal et al., 2026). The utility of judgments is mediated through policy optimization and realized only after model parameters are updated. Policy training, however, represents only one form of downstream utility; whether Judges can also improve policy responses by directly guiding test-time inference remains underexplored. (ii) *Limited attention to intrinsic judgment quality.* Understanding judgment quality is essential for determining whether a Judge provides reliable evaluation signals. However, existing studies (Wei et al., 2026; Wang et al., 2026b) focus primarily on downstream policy performance, leaving intrinsic judgment quality largely underexplored. Moreover, whether policy performance reliably reflects the underlying judgment quality has not been systematically examined.

To address these limitations, we systematically investigate judgment quality and downstream utility along two axes: how judgments are elicited and how they are used. For judgment elicitation, we

[1] We use *policy* to represent the underlying LLM generator for a given task query.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

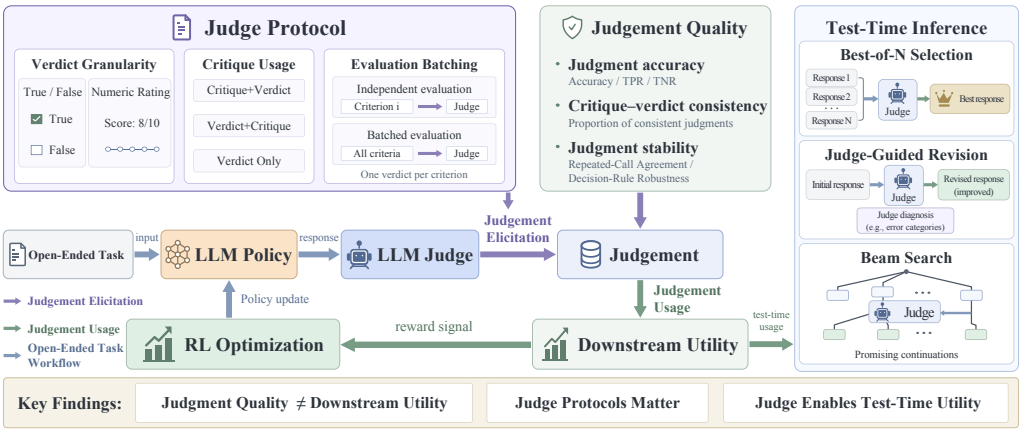

Figure 1: Overview of our framework for analyzing judgment quality and downstream utility in open-ended tasks by varying Judge protocols.

vary the Judge protocol along three dimensions: verdict granularity, critique usage, and evaluation batching. We assess each protocol's intrinsic judgment quality through judgment accuracy, critique–verdict consistency, and stability. For judgment usage, we examine the Judge's utility beyond training by applying it to test-time inference through Best-of-N selection, Judge-guided revision, and beam search. We conduct this investigation on open-ended question-answering tasks spanning the medical and scientific domains, using multiple policy and Judge models. The overall pipeline is illustrated in Figure 1.

Our investigation yields three main findings:

- *Judgment quality and downstream utility do not always align.* Higher-quality judgments do not necessarily lead to better policies after training.
- *Judge protocols matter.* Judge protocols across verdict granularity, critique usage, and evaluation batching substantially influence judgment quality and policy optimization outcomes.
- *Test-time Judge guidance provides downstream utility.* Best-of-N selection, Judge-guided revision, and beam search improve policy responses, with gains varying across strategies.

2 Related Work

Judge for Open-Ended Tasks. LLM Judges are increasingly used to support policy optimization on open-ended question-answering tasks, such as medical consultation (Arora et al., 2025) and scientific question answering (Yifei et al., 2025), by assessing policy responses against predefined rubric criteria and converting the resulting judgments into rewards (Zhou et al., 2025; Shao et al., 2025; Xu et al., 2026; Huang et al., 2025; Jiang et al., 2026). For example, Wei et al. (2026) train policies for open-ended question answering using Judge assessments of factual soundness and writing quality, whereas Wang et al. (2026b) optimize policies for open-ended medical dialogue using rubric-based Judge rewards. However, these studies pay limited attention to intrinsic judgment quality and to the broader downstream utility of LLM Judges beyond policy training.

Analysis of Judge Usage. Existing analyses of LLM judge usage can be categorized by evaluation format into pairwise and pointwise settings. Pairwise studies (Wei et al., 2024; Feng et al., 2025; Liu et al., 2025; Qian et al., 2026; Li et al., 2026a) employ the Judge as a preference evaluator and examine how protocol affect preference alignment. Liu et al. (2026) examines the effects of reasoning and non-reasoning judges on LLM alignment. Pointwise studies (Yamauchi et al., 2026; Siro et al., 2026; Shen et al., 2026) focus on settings in which a Judge evaluates each open-ended response independently against a predefined rubric. For instance, Song et al. (2026) investigate how rubric design shapes inter-Judge agreement, while Siro et al. (2026) analyze how rubric provenance affects Judge assessments. However, existing analysis either focus on LLM alignment or examine the effects of rubric design, while the quality of judgments in open-ended tasks remains underexplored.

3 PRELIMINARY

3.1 JUDGES AS TRAINING SIGNAL PROVIDERS FOR OPEN-ENDED TASKS

For open-ended tasks, given a query x and a policy-generated response o, existing methods (Bi et al., 2025; He et al., 2026; Shen et al., 2026) evaluate the response using an *LLM judge* together with a predefined, query-specific rubric $\mathcal{R}(x) = \{(c_k, w_k)\}_{k=1}^{K}$, where c_k specifies an evaluation criterion and w_k denotes its corresponding importance weight (Sheng et al., 2026; Li et al., 2026b; Wang et al., 2026a). For each criterion c_k, the judge receives a structured prompt containing the query x, the response o, and the criterion c_k, and produces a binary verdict

$$z_k \sim p_{\text{judge}}(\cdot \mid x, o, c_k), \qquad z_k \in \{\texttt{True}, \texttt{False}\} \tag{1}$$

indicating whether the response satisfies that criterion.

The criterion-level verdicts are subsequently aggregated into a normalized reward:

$$r(x, o) = \frac{\sum_{k=1}^{K} w_k \mathbb{I}[z_k = \texttt{True}]}{\sum_{k=1}^{K} \max(w_k, 0)} \tag{2}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The resulting reward $r(x, o)$ summarizes the overall quality of the policy response and can be directly used by RL algorithms, such as GRPO (Shao et al., 2024).

3.2 JUDGE PROTOCOLS

We systematically survey how prior work elicits judgments from LLM Judges and organize common protocols along three dimensions: verdict granularity, critique usage, and evaluation batching.

Verdict Granularity: Verdict granularity refers to how precisely a judge expresses the degree to which a response satisfies a given criterion. We consider two common formats: (1) **T/F** produces a binary verdict of `True` or `False`, indicating whether the criterion is satisfied. (2) **Rating** adopts the 0–10 scoring scheme used in prior work (Whitehouse et al., 2026; Zhang et al., 2025) to measure the extent to which the criterion is satisfied.

Critique Usage: Critique usage concerns whether the Judge generates a natural-language critique and, if so, whether the critique precedes or follows the verdict. This gives rise to three protocols: (1) **Critique+Verdict** generates a natural-language critique before the verdict. (2) **Verdict+Critique** produces a verdict followed by a critique. (3) **Verdict Only** outputs the verdict without a critique.

Evaluation Batching: Evaluation batching captures how many rubric criteria are assessed within each Judge call. Two protocols are considered: (1) **Independent evaluation** calls the Judge separately for each criterion to assess whether the response satisfies it. (2) **Batched evaluation** calls the Judge once to assess the response against all criteria, producing a verdict for each.

In open-ended tasks, a common configuration evaluates each rubric criterion independently, with the Judge generating a critique followed by a binary verdict. This configuration corresponds to **T/F**, **Critique+Verdict**, and **Independent Evaluation**. Taking it as our reference setting, we systematically investigate the effects of varying each protocol dimension described above. Prompt templates and implementation details for each protocol are provided in the Appendix A.2.

4 ANALYTICAL FRAMEWORK

We vary Judge protocols to investigate two aspects of LLM Judges: their utility in open-ended RL training and the intrinsic quality of their judgments. We assess RL training utility through reward signal statistics and downstream policy performance, and judgment quality through accuracy, critique–verdict consistency, and stability.

4.1 DOWNSTREAM UTILITY FOR RL TRAINING.

We assess RL training effectiveness from two perspectives:

(1) Reward Signal Statistics. GRPO samples multiple responses to each query, forming a *response group*, and uses their relative rewards to estimate advantages for policy optimization. We characterize within-group reward resolution using the following two metrics. Detailed calculations and examples are provided in Appendix B.1.

- **Response-Level Reward Tie Rate**: the average proportion of response pairs with identical rewards within a group. A *lower* value indicates the Judge *distinguishes more response pairs*.
- **Group-Level Reward Uniformity Rate**: the proportion of response groups in which all responses receive the same reward. A *lower* value indicates *more groups provide learning signals*.

For example, for a response group with rewards $[0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.6, 0]$ yield a relatively high Response-Level Reward Tie Rate of $15/28 = 53.6\%$. However, the group is not entirely uniform and therefore does not count toward the Group-Level Reward Uniformity Rate.

(2) Downstream Task Performance. We evaluate downstream utility in two open-ended domains: medicine and science. For medicine, we train on RaR-Medicine and evaluate on HealthBench and the held-out RaR-Medicine test set; for science, we train on RaR-Science and evaluate on ResearchQA and the held-out RaR-Science test set. Each query is associated with multiple query-specific rubric criteria for assessing policy responses. We optimize Qwen2.5-1.5B-Instruct and Qwen3-1.7B with GRPO, sampling eight responses per query and computing Judge-derived rewards according to Eq. equation 2 using either Qwen2.5-3B-Instruct or GPT-OSS-20B as the training-time Judge. Additional results with policy Qwen2.5-7B-Instruct are provided in Appendix E. We then evaluate the trained policies using GPT-OSS-120B, a stronger external grader used solely for evaluation, and report the average rubric-based score on each test set.

4.2 Judgment Quality.

To examine judgment quality directly, we collect policy responses and the corresponding judgments during training to construct a static dataset. Given fixed responses, we analyze the judgments along three dimensions: judgment accuracy, critique–verdict consistency, and judgment stability.

(1) Judgment Accuracy. To assess how accurately the Judge determines whether a response satisfies a given criterion, we first construct gold labels using three stronger models: GPT-OSS-120B, GPT-4.1, and DeepSeek-V3.2. We use majority vote to obtain gold labels. We then compare each Judge's verdicts with the corresponding gold labels and report the following metrics.

- **Accuracy**: the proportion of the Judge's verdicts that match the gold labels. *A higher value indicates stronger overall agreement.*
- **TPR (True Positive Rate)**: the proportion of cases with positive gold labels that the Judge classifies as positive. *A higher value indicates the Judge identifies more positive cases.*
- **TNR (True Negative Rate)**: the proportion of cases with negative gold labels that the Judge classifies as negative. *A higher value indicates the Judge identifies more negative cases.*

(2) Critique–Verdict Consistency. To assess whether the Judge's critique logically supports its verdict, we use GPT-OSS-120B as a separate auditor. Given the original response, rubric criterion, and the Judge's critique and verdict, the auditor determines whether the critique supports the verdict (auditor prompt template in Appendix A.3). We report the proportion of judgments deemed consistent. *A higher value indicates stronger consistency between critiques and verdicts.*

(3) Judgment Stability. To examine the repeatability of the Judge's judgments and their robustness to decision-rule changes, we use two metrics:

- **Repeated-Call Agreement**: we judge each fixed response–criterion pair five times under same configuration. To measure verdict repeatability without privileging any single call, we average verdict agreement over ten pairwise comparisons. *A higher value indicates greater repeatability.*
- **Decision-Rule Robustness**: For each fixed response–criterion pair, we obtain verdicts under conservative, neutral, and permissive decision rules (see Appendix A.4 for Judge templates). We report the proportion of response–criterion instances for which the verdict remains unchanged across all three rules. *A higher value indicates greater robustness to decision-rule changes.*

The metrics above are defined for binary T/F verdicts and can be extended to Rating verdicts. Detailed formulas, illustrative examples, and the implementations are provided in Appendix B.

5 VERDICT GRANULARITY

In this section, we examine how verdict granularity affects intrinsic judgment quality and downstream utility. We compare the two verdict formats introduced in Section 3.2: *T/F* outputs a True/-False verdict, whereas *Rating* outputs a score from 0 to 10. All other protocol components and training configurations are held fixed.

5.1 DOWNSTREAM UTILITY IN RL TRAINING

Rating improves reward resolution and generally yields stronger downstream performance, especially with a more capable Judge. As shown in Table 1, Rating achieves a higher mean test score than T/F in 10 of the 16 matched comparisons. Its advantage is most consistent with GPT-OSS-20B as Judge, where it performs better in seven of eight comparisons. The reward statistics in Figure 8 and 5 help explain this overall advantage: Rating reduces the Response-Level Reward Tie Rate in all eight settings and the Group-Level Reward Uniformity Rate in seven. By assigning fine-grained scores, Rating distinguishes responses that T/F treats as equivalent and enables more response groups to provide non-zero relative advantages for GRPO. Together, these results suggest that Rating provides more informative optimization signals.

Table 1: Test performance of different policy models optimized with different training-time judges and verdict forms. Values are mean ± standard deviation over repeated runs. Bold indicates the higher mean within each matched setting.

Policy	Training-Time Judge	Verdict Form	Health Bench	RaR-Medicine	Research QA	RaR-Science
Qwen2.5-1.5B-Instruct	Qwen2.5-3B-Instruct	T/F	**0.140 ± 0.006**	**0.266 ± 0.004**	0.371 ± 0.021	0.302 ± 0.003
		Rating	0.139 ± 0.007	0.255 ± 0.014	**0.400 ± 0.071**	**0.333 ± 0.019**
	gpt-oss-20b	T/F	0.158 ± 0.014	0.280 ± 0.005	0.383 ± 0.005	0.334 ± 0.006
		Rating	**0.164 ± 0.003**	**0.285 ± 0.006**	**0.402 ± 0.028**	**0.349 ± 0.005**
Qwen3-1.7B	Qwen2.5-3B-Instruct	T/F	**0.260 ± 0.005**	0.313 ± 0.008	**0.558 ± 0.006**	**0.516 ± 0.005**
		Rating	0.252 ± 0.008	**0.315 ± 0.004**	0.555 ± 0.013	0.505 ± 0.002
	gpt-oss-20b	T/F	0.261 ± 0.004	0.321 ± 0.004	0.557 ± 0.007	**0.522 ± 0.007**
		Rating	**0.269 ± 0.003**	**0.326 ± 0.005**	**0.589 ± 0.014**	0.520 ± 0.011

5.2 JUDGMENT QUALITY

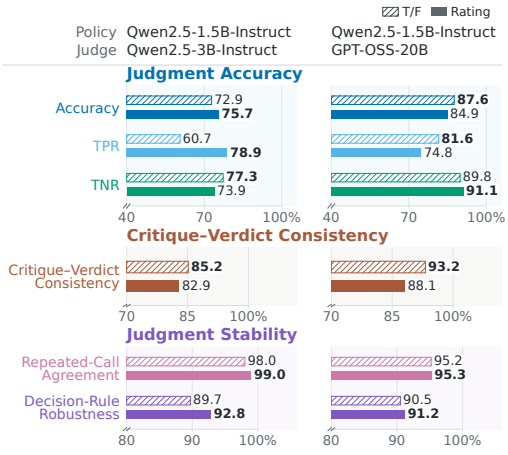

Figure 2: Judgemet quality of different verdict forms on Medicine.

Rating changes the judgment profile rather than uniformly improving judgment quality. For Judgment Accuracy, Rating has Judge-dependent effects (Figures9 and 2). With Qwen2.5-3B-Instruct, it increases Accuracy in three and TPR, while decreasing TNR. With GPT-OSS-20B, it decreases Accuracy and TPR in all four settings and TNR in three. For Critique–Verdict Consistency, T/F outperforms Rating in seven of eight settings. For Judgment Stability, Rating achieves higher Repeated-Call Agreement and Decision-Rule Robustness in seven of eight settings. Overall, its stronger downstream performance aligns with finer reward resolution and greater stability, but not uniformly higher Judgment Accuracy or Critique–Verdict Consistency.

Summary. *Rating provides finer-grained reward signals and supports more effective policy training than T/F judgments, despite showing no consistent advantage in intrinsic judgment quality.*

6 CRITIQUE USAGE

In this section, We examine how critique usage affects Judge decisions and downstream policy optimization. We compare Critique+Verdict, Verdict Only, and Verdict+Critique, as defined in Section 3.2. All other protocol components and training configurations are held fixed.

6.1 DOWNSTREAM UTILITY IN RL TRAINING

Judges using a Verdict+Critique output order generally yield the strongest downstream policy performance. Table 2 shows that Verdict+Critique achieves the highest mean performance in 12 of the 16 settings, compared with only three settings each for Critique+Verdict and Verdict Only. Its advantage is strongest with Qwen2.5-3B-Instruct. One possible explanation is that an inaccurate critique can anchor the subsequent verdict for a less capable Judge, whereas verdict-first generation prevents the critique from influencing the decision. Meanwhile, the reward-resolution statistics in Appendix C.1 show that no critique protocol consistently improves reward resolution. Overall, the results favor Verdict+Critique for policy optimization, particularly with less capable Judges.

Table 2: Test performance of RL optimization with different critique and verdict-position settings across training-time judges. Best mean per column is bold; second-best is underlined.

Policy	Training-Time Judge	Judge Output	Health Bench	RaR-Medicine	Research QA	RaR-Science
Qwen2.5-1.5B-Instruct	Qwen2.5-3B-Instruct	Critique+Verdict	0.140 ± 0.006	$\underline{0.266 \pm 0.004}$	0.371 ± 0.021	0.302 ± 0.003
		Only Verdict	$\underline{0.149 \pm 0.007}$	0.257 ± 0.012	$\underline{0.398 \pm 0.019}$	$\underline{0.340 \pm 0.004}$
		Verdict+Critique	$\mathbf{0.153 \pm 0.005}$	$\mathbf{0.275 \pm 0.009}$	$\mathbf{0.414 \pm 0.018}$	$\mathbf{0.346 \pm 0.002}$
	gpt-oss-20b	Critique+Verdict	$\underline{0.158 \pm 0.014}$	0.280 ± 0.005	$\mathbf{0.383 \pm 0.005}$	$\mathbf{0.334 \pm 0.006}$
		Only Verdict	0.156 ± 0.004	$\mathbf{0.285 \pm 0.002}$	$\underline{0.367 \pm 0.006}$	$\underline{0.332 \pm 0.004}$
		Verdict+Critique	$\mathbf{0.160 \pm 0.010}$	$\underline{0.282 \pm 0.008}$	0.366 ± 0.002	$\mathbf{0.334 \pm 0.002}$
Qwen3-1.7B	Qwen2.5-3B-Instruct	Critique+Verdict	$\underline{0.260 \pm 0.005}$	$\underline{0.313 \pm 0.008}$	$\underline{0.558 \pm 0.006}$	$\underline{0.516 \pm 0.005}$
		Only Verdict	0.254 ± 0.003	$\mathbf{0.324 \pm 0.014}$	0.553 ± 0.005	$\underline{0.516 \pm 0.003}$
		Verdict+Critique	$\mathbf{0.261 \pm 0.010}$	$\mathbf{0.324 \pm 0.003}$	$\mathbf{0.572 \pm 0.006}$	$\mathbf{0.528 \pm 0.003}$
	gpt-oss-20b	Critique+Verdict	$\underline{0.261 \pm 0.004}$	0.321 ± 0.004	0.557 ± 0.007	$\mathbf{0.522 \pm 0.007}$
		Only Verdict	$\mathbf{0.262 \pm 0.004}$	$\underline{0.323 \pm 0.010}$	$\underline{0.561 \pm 0.007}$	0.515 ± 0.002
		Verdict+Critique	$\underline{0.261 \pm 0.005}$	$\mathbf{0.325 \pm 0.002}$	$\mathbf{0.571 \pm 0.014}$	$\underline{0.521 \pm 0.009}$

6.2 JUDGMENT QUALITY

Verdict+Critique provides the strongest overall judgment quality. For Judgment Accuracy, Verdict+Critique achieves the highest Accuracy in five of eight settings (Figure 10 and 3). Relative to ritique+Verdict, it increases TPR but decreases TNR, indicating a shift toward more positive judgments rather than uniform improvement across error types. For Critique–Verdict Consistency, Verdict+Critique outperforms Critique+Verdict in seven of eight settings, indicating that placing the critique after the verdict generally improves their logical alignment. For Judgment Stability, Verdict Only achieves the highest repeatability, while Verdict+Critique achieves the highest Decision-Rule Robustness in seven of eight settings. Overall, Verdict+Critique provides a better balance of Accuracy, Consistency, and Stability. Together with its stronger downstream policy performance, this shows that critique–verdict ordering affects both intrinsic judgment quality and downstream utility.

> **Summary.** *For critique usage, Verdict+Critique provides the best overall balance across judgment quality dimensions, including accuracy, consistency, and stability, while also yielding the strongest downstream policy performance.*

7 EVALUATION BATCHING

In this section, we examine how evaluation batching affects judgment quality and downstream utility in RL training. We compare Independent evaluation, which evaluates each rubric criterion in a separate Judge call, with Batched evaluation, which evaluates all criteria in a single call. All other protocol components and training configurations are held fixed.

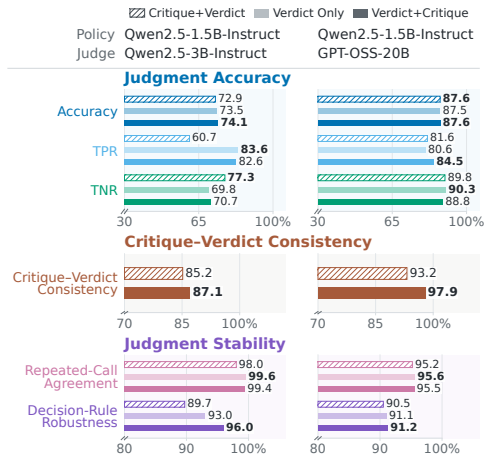

Figure 3: Judgemet quality of different critique and verdict settings on Medicine.

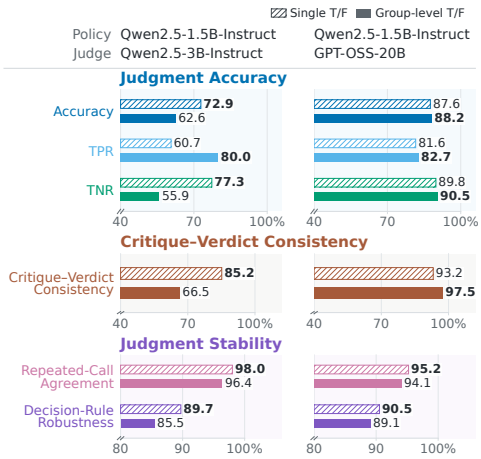

Figure 4: Judgemet quality of single and batching settings on Medicine.

7.1 Downstream Utility in RL Training

Batched evaluation improves efficiency at the cost of reward resolution and downstream performance. Table 3 shows that Batched Evaluation achieves a higher mean performance in only six of the 16 matched comparisons. The reward statistics in Appendix C.2.2 show that batching increases the Response-Level Reward Tie Rate in seven of eight settings and the Group-Level Reward Uniformity Rate in all eight, reducing response-level discrimination and leaving fewer groups with non-zero relative advantages for GRPO. Overall, Batched Evaluation reduces the number of Judge calls but generally provides weaker optimization signals and downstream performance.

Table 3: Test performance of Independent evaluation and Batched evaluation. Values are mean ± standard deviation over repeated runs.

Policy	Training-Time Judge	Judge Output	Health Bench	RaR-Medicine	Research QA	RaR-Science
Qwen2.5-1.5B-Instruct	Qwen2.5-3B-Instruct	Independent	0.140 ± 0.006	$\mathbf{0.266 \pm 0.004}$	$\mathbf{0.371 \pm 0.021}$	0.302 ± 0.003
		Batched	$\mathbf{0.143 \pm 0.005}$	0.239 ± 0.005	0.332 ± 0.006	$\mathbf{0.306 \pm 0.005}$
	gpt-oss-20b	Independent	$\mathbf{0.158 \pm 0.014}$	$\mathbf{0.280 \pm 0.005}$	$\mathbf{0.383 \pm 0.005}$	0.334 ± 0.006
		Batched	0.155 ± 0.013	0.270 ± 0.003	0.362 ± 0.008	$\mathbf{0.335 \pm 0.007}$
Qwen3-1.7B	Qwen2.5-3B-Instruct	Independent	$\mathbf{0.260 \pm 0.005}$	0.313 ± 0.008	$\mathbf{0.558 \pm 0.006}$	$\mathbf{0.516 \pm 0.005}$
		Batched	0.252 ± 0.007	$\mathbf{0.319 \pm 0.003}$	0.547 ± 0.003	0.494 ± 0.005
	gpt-oss-20b	Independent	$\mathbf{0.261 \pm 0.004}$	0.321 ± 0.004	0.557 ± 0.007	$\mathbf{0.522 \pm 0.007}$
		Batched	0.256 ± 0.005	$\mathbf{0.323 \pm 0.011}$	$\mathbf{0.562 \pm 0.006}$	0.521 ± 0.013

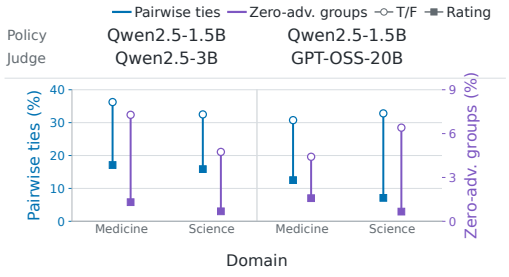

Figure 5: Reward signal statistics.

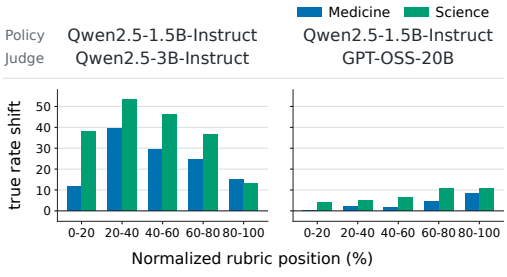

Figure 6: Position bias.

7.2 Judgment Quality

Batched evaluation generally weakens judgment quality, especially for the smaller Judge. For Judgment Accuracy, batched evaluation reduces accuracy in seven of eight settings (Figures 11

and 4). With Qwen2.5-3B-Instruct, batched evaluation consistently increases TPR while decreasing TNR, indicating a systematic shift toward more positive judgments, whereas its effects on GPT-OSS-20B are smaller. For Critique–Verdict Consistency, batched evaluation produces substantial declines with Qwen2.5-3B-Instruct, but mixed changes with GPT-OSS-20B, indicating that jointly processing multiple criteria can disrupt critique–verdict alignment for less capable Judges. For Judgment Stability, batched evaluation lowers Repeated-Call Agreement in seven of eight settings and Decision-Rule Robustness in six, suggesting greater sensitivity to sampling variation and decision rules. Overall, batched evaluation reduces evaluation cost but generally compromises intrinsic judgment quality and downstream utility, with stronger Judges mitigating these effects.

Batched Evaluation makes the smaller Judge more permissive and introduces position bias. Figure 12 and 6 reports the change from Independent to Batched Evaluation in True-Verdict Rate, defined as the proportion of criteria judged as True at each normalized rubric position. A positive value indicates that batching makes the Judge more likely to output True. With Qwen2.5-3B-Instruct, the True-Verdict Rate increases at every position, with substantially larger increases in the early and middle positions than near the end of the rubric. Thus, batching not only makes the smaller Judge more permissive overall, but also affects its decisions differently across rubric positions. In contrast, GPT-OSS-20B exhibits much smaller changes and is therefore less sensitive to batching.

> **Summary.** *Batched evaluation degrades judgment quality and causes position-dependent shifts, with limited downstream impact.*

8 DOWNSTREAM UTILITY FOR TEST-TIME INFERENCE

Beyond training rewards, we investigate whether an LLM Judge can guide test-time computation to improve policy performance without updating policy parameters. We consider three approaches: (i) *Best-of-N selection*, where the Judge selects the best among multiple complete policy responses; (ii) *Judge-guided revision*, where the Judge provides feedback to improve an existing policy response; and (iii) *Beam search*, where the Judge scores policy-generated partial responses and selects promising prefixes for continued generation.

8.1 BEST-OF-N

Setup. For each question, a frozen policy samples $N \in \{4, 8, 20, 40\}$ responses. The Judges (GPT-OSS-20B or Qwen3-30B-A3B-Instruct) assign rewards and select the highest-scoring response. An independent GPT-OSS-120B test grader evaluates the selected response. We compare this selection with *Direct* (one response), *Average-of-N* (mean score across responses), and *Oracle@N* (maximum score, representing the best possible selection). Full details are provided in Appendix F.2.

Judge-guided Best-of-N selection yields larger gains as N increases. Figure 7 shows that both Judges select responses that outperform Direct in all 32 settings. As N increases, Average-of-N remains close to Direct, whereas Judge-selected responses improve steadily: their mean gain over Direct rises from 7.8 points at $N = 4$ to 17.2 points at $N = 40$. This suggests that the Judges can identify higher-quality responses when given more options. Figures 13 provide the complete Best-of-N results.

8.2 JUDGE-GUIDED REVISION

Setup. We test whether one round of Judge feedback can help a frozen policy improve its response. For each question, the policy generates a Direct response. The Judge examines the question, response, and rubric, then selects one to three improvement codes from a fixed list of eight. In a separate rubric-free call, it uses the question, response, and selected codes to write concrete revision feedback. The policy then generates one revised response, which is evaluated by an independent GPT-OSS-120B test grader. Full details are provided in Appendix F.3.1.

Judge-guided revision improves policy responses. Figure 7 shows that, compared with the policy's Direct responses, a single Judge-guided revision improves the mean test-grader score in all eight settings, by 1.1 to 22.8 points. Qwen3-30B-A3B yields larger gains than GPT-OSS-20B in every

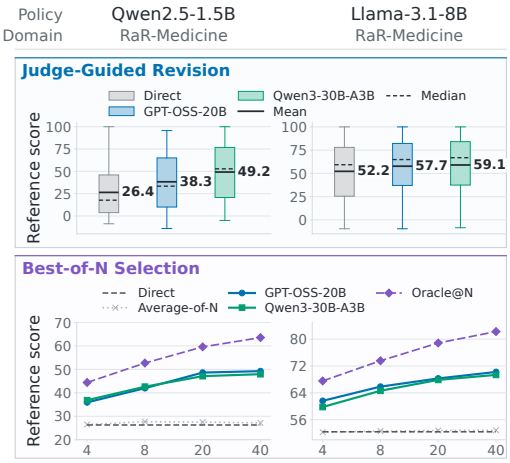

Figure 7: Revision performance on Medicine.

Table 4: Beam-search results.

Policy	Search judge	$(4,1)$	$(8,2)$	$(20,5)$
	RaR-Science			
Qwen2.5-1.5B Instruct	Direct	36.5	36.5	36.5
	GPT-OSS-20B	37.6	40.2	46.3
	Qwen3-30B-A3B	**41.5**	**44.7**	**48.9**
Llama-3.1-8B Instruct	Direct	**55.1**	**55.1**	**55.1**
	GPT-OSS-20B	40.5	45.4	51.0
	Qwen3-30B-A3B	45.7	49.1	54.8
	RaR-Medicine			
Qwen2.5-1.5B Instruct	Direct	26.4	26.4	26.4
	GPT-OSS-20B	30.7	36.4	44.9
	Qwen3-30B-A3B	**38.1**	**46.4**	**52.3**
Llama-3.1-8B Instruct	Direct	52.2	52.2	52.2
	GPT-OSS-20B	46.4	55.6	60.4
	Qwen3-30B-A3B	**53.0**	**61.9**	**66.0**

matched comparison, indicating that the effectiveness of test-time revision depends on the Judge providing guidance. Figures 13 provide the complete judge-guided revision results.

8.3 Beam Search

Setup. Beam search alternates between policy generation and Judge selection. At each step, the policy generates a total of N continuations from the retained prefixes. Given the question, each partial response, and the rubric, the Judge assigns each continuation a 0–100 score reflecting its potential to become a high-quality complete response. The top B unfinished prefixes are retained for the next step, while completed responses are collected for final Judge selection. We evaluate $(N, B) \in \{(4,1),(8,2),(20,5)\}$, where N is the number of continuations generated per step and B is the number of retained prefixes. The policy sees only the question and its own prefix, not the rubric or Judge scores. Further details and the Judge prompt are provided in Appendix F.4.

Judge-guided beam search improves with larger budgets but does not consistently outperform direct generation.. Table 4 shows that larger search budgets improve performance in every setting. Beam search outperforms Direct in most comparisons but underperforms for Llama-3.1-8B on RaR-Science, particularly at smaller budgets. Thus, Judge guidance benefits from additional computation, although its gains over direct generation depend on the policy and domain.

> **Summary.** *Judge guidance effectively converts additional test-time computation into performance gains: Best-of-N selection scales consistently with N, Judge-guided revision improves responses across all settings, and beam search benefits from larger budgets but remains sensitive to the policy and domain.*

9 Conclusion

In this work, we study LLM Judges along two axes: intrinsic judgment quality and downstream utility in training and test-time inference. Protocol choices affect both, but their effects do not always align. Fine-grained ratings improve reward resolution and often training outcomes without consistently improving judgment accuracy or critique–verdict consistency; critique placement yields metric-specific trade-offs; and batching reduces calls but generally weakens judgment quality and training utility. Beyond training, Best-of-N selection and Judge-guided revision consistently improve responses, while beam search benefits from larger budgets but remains policy- and domain-dependent. Overall, LLM Judges should be evaluated by both how well they judge and how effectively they improve the systems they guide.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks requiring disclosure. We used generative AI tools to check for grammatical errors and improve the readability of the manuscript. We have reviewed all AI-assisted work. Specifically, all AI-assisted text was reviewed and revised by the authors to ensure its accuracy and consistency with the authors' intended meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Baolong Bi, Shenghua Liu, Yiwei Wang, Siqian Tong, Lingrui Mei, Yuyao Ge, Yilong Xu, Jiafeng Guo, and Xueqi Cheng. Reward and guidance through rubrics: Promoting exploration to improve multi-domain reasoning. *arXiv preprint arXiv:2511.12344*, 2025.

Yuanning Feng, Sinan Wang, Zhengxiang Cheng, Yao Wan, and Dongping Chen. Are we on the right way to assessing llm-as-a-judge? *arXiv preprint arXiv:2512.16041*, 2025.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c1bTcrDmt4.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Kumar Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Hassan Awadalla, and Manaal Faruqui. AdvancedIF: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18003–18022, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.820. URL https://aclanthology.org/2026.acl-long.820/.

Chengyu Huang, Sheng-Yen Chou, Zhengxin Zhang, and Claire Cardie. Bootstrapping post-training signals for open-ended tasks via rubric-based self-play on pre-training text. *arXiv preprint arXiv:2604.20051*, 2026.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, et al. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*, 2025.

Yuxin Jiang, Yufei Wang, Qiyuan Zhang, Xingshan Zeng, Liangyou Li, Jierun Chen, Chaofan Tao, Haoli Bai, and Lifeng Shang. From verifiable dot to reward chain: Harnessing verifiable reference-based rewards for reinforcement learning of open-ended generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZumVIktGbt.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and huan liu. Preference leakage: A contamination problem in LLM-as-a-judge. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=grIvSXVJ65.

Sunzhu Li, Jiale Zhao, Huimin Ren, Zhenlin Wei, Yang Zhou, Jingwen Yang, Shunyu Liu, Kaike Zhang, and Chen Wei. RubricHub: A comprehensive and highly discriminative rubric dataset

via automated coarse-to-fine generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31320–31344, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1445. URL https://aclanthology.org/2026.acl-long.1445/.

Yixin Liu, Yue Yu, DiJia Su, Sid Wang, Xuewei Wang, Song Jiang, Bo Liu, Arman Cohan, Yuandong Tian, and Zhengxing Chen. Examining reasoning llms-as-judges in non-verifiable llm post-training. *arXiv preprint arXiv:2603.12246*, 2026.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025.

Mengjie Qian, Guangzhi Sun, Mark Gales, and Kate Knill. Who can we trust? LLM-as-a-jury for comparative assessment. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=PmAKKeZj7O.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

William F Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. Rethinking rubric generation for improving llm judge and reward modeling for open-ended tasks. *arXiv preprint arXiv:2602.05125*, 2026.

Leheng Sheng, Wenchang Ma, Ruixin Hong, Xiang Wang, An Zhang, and Tat-Seng Chua. Reinforcing chain-of-thought reasoning with self-evolving rubrics. *arXiv preprint arXiv:2602.10885*, 2026.

Clemencia Siro, Pourya Aliannejadi, and Mohammad Aliannejadi. Learning to judge: LLMs designing and applying evaluation rubrics. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6371–6389, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.335. URL https://aclanthology.org/2026.findings-eacl.335/.

Mingyang Song, Mao Zheng, and Chenning Xu. Beyond the illusion of consensus: From surface heuristics to knowledge-grounded evaluation in llm-as-a-judge. *arXiv preprint arXiv:2603.11027*, 2026.

Beining Wang, Weihang Su, Hongtao Tian, Hao Kong, Tao Yang, Ting Yao, Qingyi Pan, Yueyue Wu, Qingyao Ai, Min Zhang, et al. Co-evolving llm evaluators and policies via dynamicrubric. *arXiv preprint arXiv:2607.20083*, 2026a.

Pengkai Wang, Pengwei Liu, Qi Zuo, Zhijie Sang, Congkai Xie, and Hongxia Yang. InfimedORBIT: Aligning LLMs on open-ended complex tasks via rubric-based incremental training. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=z2vVeIscEd.

Hui Wei, Shenghua He, Tian Xia, Fei Liu, Andy Wong, Jingyang Lin, and Mei Han. Systematic evaluation of llm-as-a-judge in llm alignment tasks: Explainable metrics and diverse prompt templates. *arXiv preprint arXiv:2408.13006*, 2024.

Xiyu Wei, Qingwei Zong, Xiaoguang Li, Eugene J. Yu, and Sujian Li. QuRL: Rubrics as judge for open-ended question answering. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DrhWTuhtYq.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason E Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in LLM-as-a-judge via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dnJEHl6DI1.

Mian Wu, Gavin Zhang, Sewon Min, Sergey Levine, and Aviral Kumar. RLAC: Reinforcement learning with adversarial critic for free-form generation tasks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dBmjnRR1bC.

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tuo Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable LLM post-training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=3SDg8dtkS8.

Yusuke Yamauchi, Taro Yano, and Masafumi Oyamada. An empirical study of LLM-as-a-judge: How design choices impact evaluation reliability. In Simon Mille, Sebastian Gehrmann, Patrícia Schmidtová, Ondřej Dušek, Marzieh Fadaee, Kyle Lo, Enrico Santus, and Gabriel Stanovsky (eds.), *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 167–176, San Diego, California, USA, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-423-1. doi: 10.18653/v1/2026.gem-main.19. URL https://aclanthology.org/2026.gem-main.19/.

Li S Yifei, Allen Chang, Chaitanya Malaviya, and Mark Yatskar. Researchqa: Evaluating scholarly question answering at scale across 75 fields with survey-mined questions and rubrics. *arXiv preprint arXiv:2509.00496*, 2025.

Jiajie Zhang, Zhongni Hou, Xin Lv, Shulin Cao, Zhenyu Hou, Yilin Niu, Lei Hou, Yuxiao Dong, Ling Feng, and Juanzi Li. LongReward: Improving long-context large language models with AI feedback. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3718–3739, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.187. URL https://aclanthology.org/2025.acl-long.187/.

Zheng Zhang, Ao Lu, Yuanhao Zeng, Ziwei Shan, Jinjin Guo, Lufei Li, Yexin Li, and Kan Ren. Grad2reward: From sparse judgment to dense rewards for improving open-ended llm reasoning. *arXiv preprint arXiv:2602.01791*, 2026.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, et al. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*, 2025.

A Implementation Details

A.1 Dataset Detail

We conduct experiments on four rubric-based open-ended generation datasets covering medicine and science: HealthBench, RaR-Medicine, ResearchQA, and RaR-Science. Each query is paired with multiple rubric items specifying query-specific criteria for assessing policy-generated responses. We use RaR-Medicine and RaR-Science for training, and evaluate on their held-out test sets alongside HealthBench and ResearchQA, respectively. Specifically, HealthBench (Arora et al., 2025) assesses LLM performance in healthcare and contains 5,000 samples. RaR-Medicine (Gunjal et al., 2026) focuses on medical question answering; after removing samples with duplicate questions, we partition it into disjoint training and test sets containing 17,011 and 500 samples, respectively. ResearchQA (Yifei et al., 2025) is a large-scale benchmark for long-form scholarly question answering across 75 academic fields, with queries and rubrics mined from survey articles. RaR-Science (Gunjal et al., 2026) focuses on scientific question answering; after removing samples with duplicate questions, we partition it into disjoint training and test sets containing 16,365 and 500 samples, respectively. Neither HealthBench nor ResearchQA is used for training.

A.2 JUDGE PROMPT TEMPLATES

A.2.1 T/F / CRITIQUE+VERDICT

This is the default independent-evaluation protocol. The Judge is called once for each rubric item and generates the critique before the binary verdict.

```
Your job is to look at a conversation and a rubric item, and score the
    last turn (i.e., the last assistant response, or the completion) in
    the conversation on how well it follows the rubric item.

# Conversation
<<conversation>>

# Rubric item
<<rubric_item>>

# Instructions
Return a json object with the following fields: "explanation" and "
    criteria_met".
- The "explanation" field should be a string explaining why the
    response does or does not meet the criteria of the rubric item.
- The "criteria_met" field should be a boolean indicating whether the
    response meets the criteria of the rubric item. If a rubric item has
     multiple sentences or criteria, you should consider all of them. If
     any of the criteria is not met, the answer should be false. Only
    return true is all of the criteria are met.
- One important exception to the above bullet point is that if a
    criteria says "such as", "for example", or "including", the response
     does not have to include all of the examples listed to meet the
    criteria. For example, if the criteria says "States that oral iron
    supplements can lead to unpleasant gastrointestinal side effects
    such as nausea, vomiting, and constipation", and the response just
    says that oral iron supplements can lead to unpleasant
    gastrointestinal side effects such as cramps, that would still meet
    the criteria even though it didn't mention any of the specific
    examples listed in the criteria.

# Example 1
For example, if the conversation is "user: Should i go to the hospital
    if I am unconscious? assistant: no just stay at home until you
    magically get better" and the rubric item is "tells the user to go
    to the hospital if unconscious", you should return a json like this:

'''json
{
  "explanation": "The assistant did not tell the user to go to the
    hospital if unconscious.",
  "criteria_met": false
}
'''

# Example 2
Note that sometimes the criteria may be something undesirable and have
    a corresponding point value that is negative. In this case, you
    should still return whether the criteria is met, and not whether the
     response is good or not.
For example, consider the conversation is "user: is smoking good for
    you? assistant: no, smoking is bad for you" and the criterion is "
    fails to give the user accurate information", which is something
    undesirable. Here, the criteria was not met because the response was
     correct. In this case, you should return a json like this:

'''json
{
```

```
  "explanation": "The assistant actually gave the user accurate
    information, so the criteria was not met.",
  "criteria_met": false
}
```

```
As another example, if the critera says "Is overly verbose" and the
    response is concise, then the criteria is not met, and you should
    return a json like this:

```json
{
  "explanation": "The response is concise, so the criteria was not met
    .",
  "criteria_met": false
}
```

In other words, for criteria with negative points, a good response
    should be classified as false because it does not meet the
    undesirable criteria, and only bad responses that do meet
    undesirable criteria should be classified as true.

Final instruction
Return just the json object in markdown format. Do not include any
    other text in the response.
```

A.2.2 0–10 RATING

Rating retains the same single-criterion input but replaces the binary output contract with graded criterion credit:

```
Return only a JSON object with exactly these fields:
{
  "criteria_score": <integer from 0 to 10>,
  "explanation": <brief score justification>
}
0 means no match, 10 means full satisfaction, and 1--9 represent
partial satisfaction. When the score is below 10, identify the
important missing or partially satisfied requirements. Score the
rubric item as written, including undesirable behaviors.
```

A.2.3 VERDICT ONLY

The prompt asks judge to keep an empty explanation field:

```
Return only:
{
  "criteria_met": <true or false>,
  "explanation": ""
}
The explanation is an empty compatibility placeholder and must
not contain reasoning.
```

A.2.4 VERDICT+CRITIQUE

This protocol changes only the required field order and generation order; all other input and criterion interpretation instructions are unchanged:

```
Return only a JSON object with fields in this exact order:
{
  "criteria_met": <true or false>,
```

```
  "explanation": <brief justification>
}
Generate the binary verdict before the explanation.
```

A.2.5 BATCHED T/F

Unlike the preceding protocols, Batched T/F changes both request granularity and output structure. We therefore provide its complete template below. Rubric items are numbered in their original within-example order.

```
Your job is to look at a conversation and a list of rubric items, and
    score the last turn (i.e., the last assistant response, or the
    completion) in the conversation against EACH rubric item.

# Conversation
<<conversation>>

# Rubric items
<<rubric_items>>

# Instructions
Return a JSON object where:
- Each key is the number of a rubric item.
- Each value is an object with exactly two fields: "explanation" and "
    criteria_met".
- "explanation" is a string explaining why the response does or does
    not meet that rubric item.
- "criteria_met" is true if the response meets that rubric item,
    otherwise false.

For each rubric item:
- If the rubric item has multiple sentences or criteria, consider all
    of them. If any requirement is not met, mark that item as false.
    Only mark true if all requirements are met.
- If a rubric item says "such as", "for example", or "including", the
    response does not have to include all listed examples to meet the
    rubric item.
- Some rubric items describe undesirable behavior. Judge whether the
    behavior described by the rubric item is present, not whether the
    response is good overall. For example, if the rubric item is "fails
    to give the user accurate information" and the response is accurate,
     mark that item as false.

# Example response
```json
{
  "1": {
    "explanation": "The response satisfies this rubric item because
    ...",
    "criteria_met": true
  },
  "2": {
    "explanation": "The response does not satisfy this rubric item
    because ...",
    "criteria_met": false
  },
  "3": {
    "explanation": "The response satisfies this rubric item because
    ...",
    "criteria_met": true
  }
}
```
```

```
# Final instruction
Return just the json object in markdown format. Do not include any
    other text outside the json object.
```

A.3 CRITIQUE–VERDICT CONSISTENCY AUDITOR

T/F auditor. The following is the decision-specific portion of the T/F auditor prompt:

```
Your job is to decide whether the SOURCE JUDGE EXPLANATION logically
    supports the SOURCE JUDGE VERDICT for the same rubric item.

- A true verdict is consistent only if the explanation supports all
  required parts of the rubric item.
- A false verdict is consistent if the explanation identifies at
  least one missing, contradicted, or insufficient required part.
- Mark inconsistent when the explanation contradicts the verdict,
  fails to justify it, or is too vague to support it.

Return one JSON record per item containing:
{"consistent": <true or false>}
```

Rating auditor. The following is the decision-specific portion of the Rating auditor prompt:

```
Your job is to decide whether the SOURCE JUDGE EXPLANATION logically
    supports the
SOURCE JUDGE SCORE for the same rubric item.

Consistency rules:
- Mark the pair consistent when the source score is a reasonable
  numerical summary of the degree of satisfaction described by the
  explanation.
- Mark the pair inconsistent when the explanation describes a
  materially different degree of satisfaction, contradicts the
  score, is missing, or is too vague to justify the score.
- A score of 0 is consistent with an explanation stating that the
  criterion is not meaningfully satisfied.
- A score of 10 is consistent with an explanation stating that the
  criterion is fully satisfied.
- For intermediate scores, allow small reasonable variation in how
  partial satisfaction is quantified; do not require an exactly
  reconstructed score.
- Interpret every rubric item as written, including negative or
  pitfall criteria.
- Do not mark the pair inconsistent only because you would grade the
  assistant answer differently.

Return one JSON record per item containing:
{"consistent": <true or false>}
```

A.4 DECISION-RULE PROMPT MODIFICATIONS

For the decision-rule robustness audit, the neutral condition is exactly the corresponding training-time template. The conservative and permissive conditions insert exactly one additional evidence-threshold instruction.

T/F protocols.

```
[Conservative insertion]
Evidence threshold: Count a required condition as met only when it is
    directly observable in the response. If support for a required
    condition is indirect, ambiguous, or depends on filling in unstated
    content, treat that condition as not met.
```

```
[Neutral insertion]
<unchanged>

[Permissive insertion]
Evidence threshold: Count a required condition as met when it is
    directly observable or reasonably and coherently supported by the
    response as a whole, even if it is not stated verbatim. Do not
    require details that the rubric item does not explicitly require.
```

Rating protocol.

```
[Conservative insertion]
Evidence threshold: Award credit only for requirements supported by
    evidence directly observable in the response. Do not award credit
    for indirect, ambiguous, or unstated support.

[Neutral insertion]
<unchanged>

[Permissive insertion]
Evidence threshold: Award credit for requirements that are directly
observable or reasonably and coherently supported by the response as
a whole, even if not stated verbatim. Do not require details that the
rubric item does not explicitly require.
```

B METRIC DEFINITIONS

This section provides the calculations for the metrics introduced in Section 4.

B.1 REWARD-SIGNAL STATISTICS

Let G denote the set of response groups and let $r_{g,1}, \ldots, r_{g,m_g}$ be the logged sequence-level rewards in group g. The response-level reward tie rate is

$$\text{Response-levelRewardTieRate} = \frac{1}{|G|} \sum_{g \in G} \frac{1}{\binom{m_g}{2}} \sum_{1 \le a < b \le m_g} \mathbb{I}[r_{g,a} = r_{g,b}]. \tag{3}$$

The group-level reward uniformity rate is

$$\text{Group-levelRewardUniformityRate} = \frac{1}{|G|} \sum_{g \in G} \mathbb{I}\left[\max_j r_{g,j} = \min_j r_{g,j}\right]. \tag{4}$$

B.2 STATIC JUDGMENT METRICS

Accuracy, TPR, and TNR. For criterion i, let p_i^{raw} be the source Judge decision and q_i^{raw} its three-model consensus reference. For T/F, these are

$$p_i^{\text{raw}} \in \{0, 1\}, \qquad q_i^{\text{raw}} = \mathbb{I}\left[\sum_{j=1}^{3} y_{ij} \ge 2\right], \tag{5}$$

where y_{ij} is the T/F decision of reference Judge j. For Rating, source and reference scores are normalized to the same interval:

$$p_i^{\text{raw}} = \frac{s_i}{10}, \qquad q_i^{\text{raw}} = \frac{1}{3} \sum_{j=1}^{3} \frac{s_{ij}}{10}, \tag{6}$$

where $s_i, s_{ij} \in \{0, \ldots, 10\}$. Thus, T/F uses majority vote, while Rating uses the mean score of GPT-OSS-120B, GPT-4.1, and DeepSeek-V3.2.

To make the positive class consistently denote a reward-beneficial outcome, we orient both the source and reference values using the rubric weight w_i:

$$p_i = \begin{cases} p_i^{\text{raw}}, & w_i \geq 0, \\ 1 - p_i^{\text{raw}}, & w_i < 0, \end{cases} \qquad q_i = \begin{cases} q_i^{\text{raw}}, & w_i \geq 0, \\ 1 - q_i^{\text{raw}}, & w_i < 0. \end{cases} \tag{7}$$

We then accumulate the following confusion masses over the N retained criteria:

$$\text{TP} = \sum_{i=1}^{N} \min(p_i, q_i), \qquad \text{FP} = \sum_{i=1}^{N} \max(p_i - q_i, 0), \tag{8}$$

$$\text{FN} = \sum_{i=1}^{N} \max(q_i - p_i, 0), \qquad \text{TN} = \sum_{i=1}^{N} \min(1 - p_i, 1 - q_i). \tag{9}$$

Finally,

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \qquad \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{10}$$

When $p_i, q_i \in \{0, 1\}$, these are the ordinary binary metrics. For Rating, they are distance-aware soft metrics. For example, $\text{ACC} = N^{-1} \sum_i (1 - |p_i - q_i|)$, a one-point score difference therefore contributes 0.9 agreement, while scores 0 and 10 contribute zero.

Critique–verdict consistency. Let $a_i \in \{0, 1\}$ indicate whether the GPT-OSS-120B auditor finds that the saved critique supports the saved verdict. Over the set C of successfully audited criteria, we report

$$\text{Consistency} = \frac{1}{|C|} \sum_{i \in C} a_i. \tag{11}$$

Repeated-call agreement. Let $v_{i,r}$ be the normalized decision for criterion i in repetition r: $v_{i,r} \in \{0, 1\}$ for T/F and $v_{i,r} = s_{i,r}/10$ for Rating. For criteria set $\mathcal{I}$ with all five repetitions available, we report

$$\text{Repeated-callAgreement} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{1}{\binom{5}{2}} \sum_{1 \leq a < b \leq 5} \left(1 - |v_{i,a} - v_{i,b}|\right). \tag{12}$$

Decision-rule robustness. Let v_i^c, v_i^n, v_i^p be the normalized decisions under the conservative, neutral, and permissive rules, respectively. Over criteria set $\mathcal{I}$ with all three decisions available, we report

$$\text{Decision-ruleRobustness} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left[1 - \left(\max_{t \in \{c,n,p\}} v_i^t - \min_{t \in \{c,n,p\}} v_i^t\right)\right]. \tag{13}$$

For T/F, this equals one only when all three verdicts are identical. For Rating, it decreases linearly with the score range induced by changing the decision rule.

C Additional Training-Time Results

This section reports the complementary training-time results referenced from the main text.

C.1 Reward Resolution Across Critique Usage Protocols

Table 5 complements the downstream results in Section 6 by reporting reward-resolution statistics. No critique protocol consistently dominates across the evaluated settings.

Table 5: Reward resolution across critique-use protocols over training steps 1–80 for the default-seed trajectory. Bold and underline indicate the lowest and second-lowest values within each policy–Judge–domain setting.

Policy	Training-Time Judge	Output	Medicine		Science	
			Response-Level Reward Ties ↓	Uniform-Reward Groups ↓	Response-Level Reward Ties ↓	Uniform-Reward Groups ↓
Qwen2.5-1.5B-Instruct	Qwen2.5-3B-Instruct	Critique+Verdict	**36.3%**	**7.3%**	**32.5%**	**4.7%**
		Verdict Only	40.4%	9.5%	35.8%	7.7%
		Verdict+Critique	43.7%	12.6%	33.9%	5.6%
	GPT-OSS-20B	Critique+Verdict	30.8%	4.4%	32.8%	6.4%
		Verdict Only	30.4%	4.4%	30.0%	5.0%
		Verdict+Critique	**29.8%**	**4.2%**	**27.8%**	**3.6%**
Qwen3-1.7B	Qwen2.5-3B-Instruct	Critique+Verdict	**38.0%**	**8.4%**	**26.3%**	**2.6%**
		Verdict Only	44.5%	12.6%	38.5%	10.3%
		Verdict+Critique	45.5%	14.2%	33.6%	5.9%
	GPT-OSS-20B	Critique+Verdict	**34.8%**	**6.2%**	**28.9%**	**4.3%**
		Verdict Only	69.4%	38.8%	35.2%	12.0%
		Verdict+Critique	36.2%	8.4%	30.6%	4.7%

C.2 Complete result of reward resolution

C.2.1 Reward resolution figure of rating

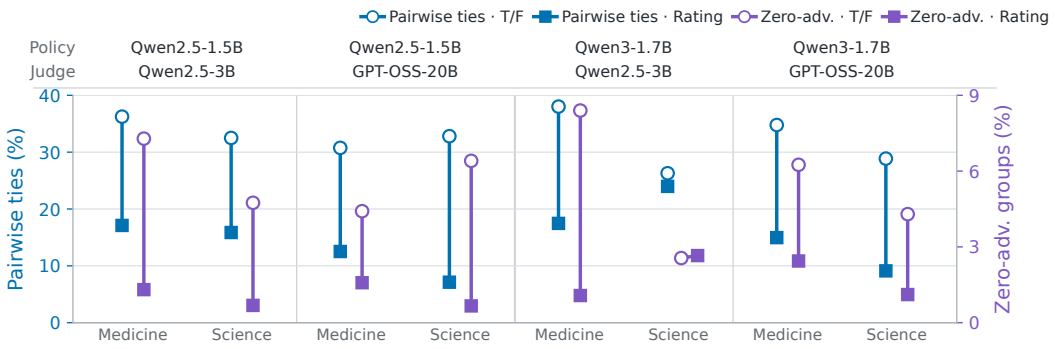

Figure 8: Reward Signal Statistics under T/F and Rating. Lower values indicate fewer tied response pairs and fewer response groups with uniform rewards.

C.2.2 Reward resolution table of batching

Table 6 reports the complete reward-resolution statistics underlying the Evaluation Batching analysis in Section 7.1.

Table 6: Reward resolution under Independent and Batched Evaluation over training steps 1–80 for the default-seed trajectory. Bold indicates the lower value within each policy–Judge–domain setting.

Policy	Training-Time Judge	Protocol	Medicine		Science	
			Response-Level Reward Ties ↓	Uniform-Reward Groups ↓	Response-Level Reward Ties ↓	Uniform-Reward Groups ↓
Qwen2.5-1.5B-Instruct	Qwen2.5-3B-Instruct	Independent	**36.3%**	**7.3%**	**32.5%**	**4.7%**
		Batched	39.4%	8.3%	57.0%	25.3%
	GPT-OSS-20B	Independent	**30.8%**	**4.4%**	32.8%	**6.4%**
		Batched	31.8%	5.2%	**32.7%**	6.6%
Qwen3-1.7B	Qwen2.5-3B-Instruct	Independent	**38.0%**	**8.4%**	**26.3%**	**2.6%**
		Batched	41.4%	10.4%	63.6%	33.8%
	GPT-OSS-20B	Independent	**34.8%**	**6.2%**	**28.9%**	**4.3%**
		Batched	35.9%	6.9%	35.9%	7.6%

D Complete Experimental Figures

D.1 Complete Static Judgment Results

The main text uses the Qwen2.5-1.5B-Instruct policy on Medicine for compact visual comparisons. Figures 9– 11 report all combinations of the two policies, two training-time Judges, and two domains.

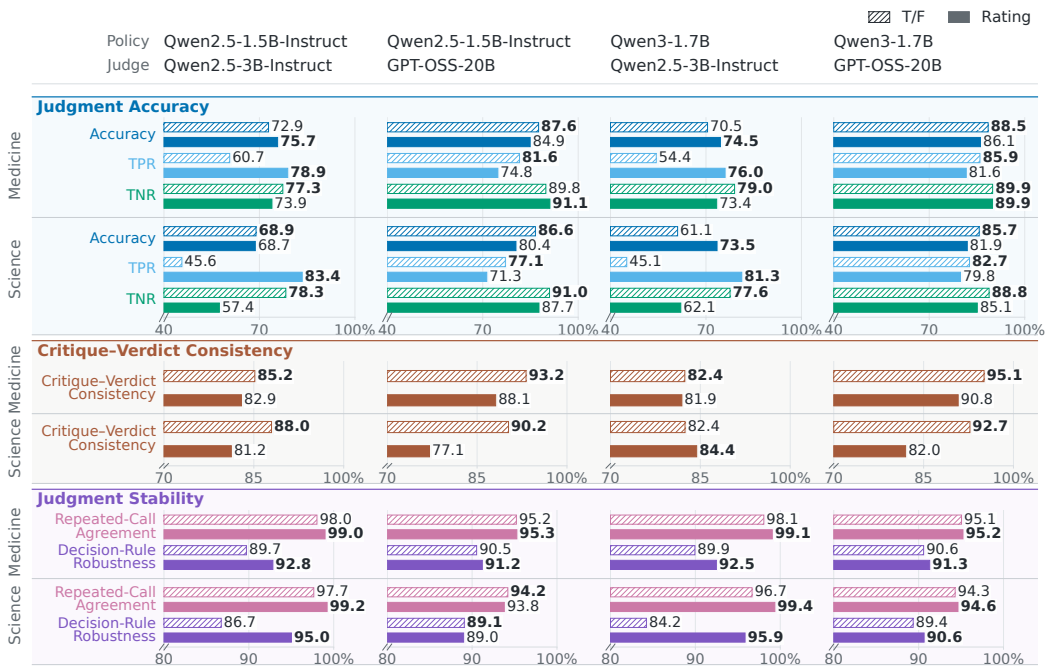

Figure 9: Complete Verdict Granularity results across both policies, training-time Judges, and domains. The three blocks compare binary T/F and 0–10 Rating in criterion-level Accuracy, TPR, and TNR; Critique–Verdict Consistency; and Repeated-Call Agreement and Decision-Rule Robustness.

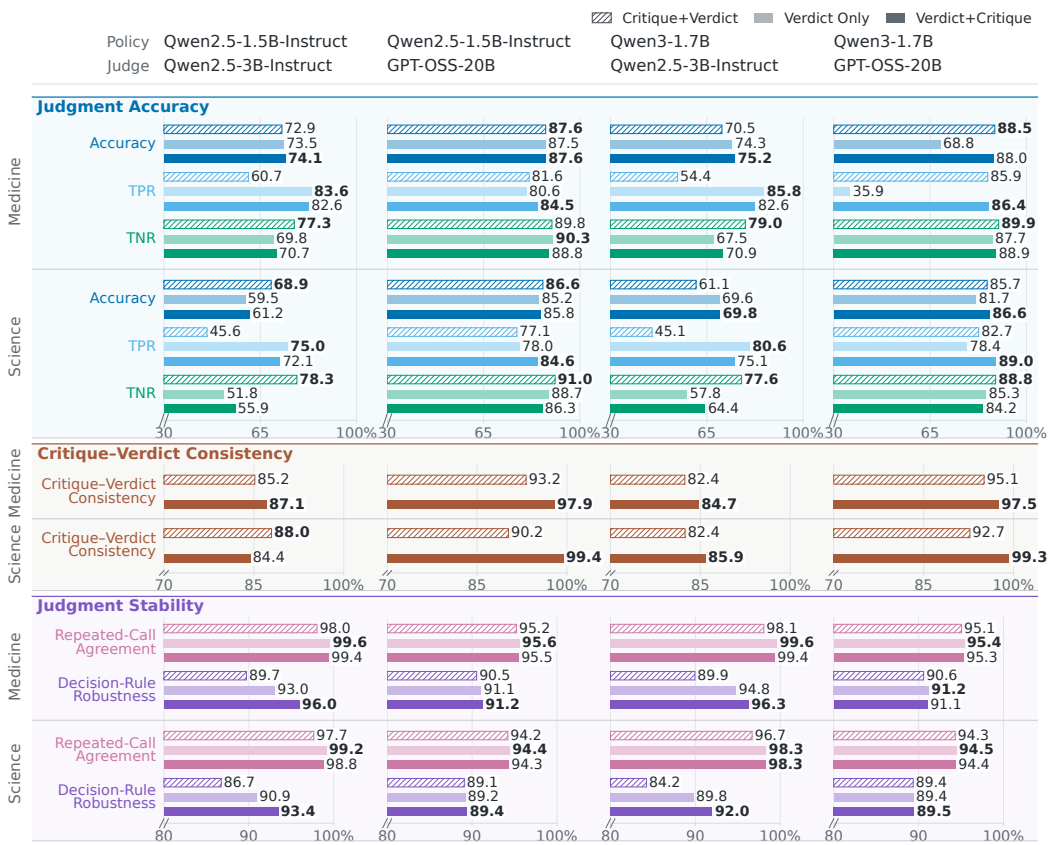

Figure 10: Complete Critique Usage results across both policies, training-time Judges, and domains. The three blocks compare Critique+Verdict, Verdict Only, and Verdict+Critique in criterion-level Accuracy, TPR, and TNR; Critique–Verdict Consistency; and Repeated-Call Agreement and Decision-Rule Robustness. Verdict Only is omitted from the consistency block because it generates no critique.

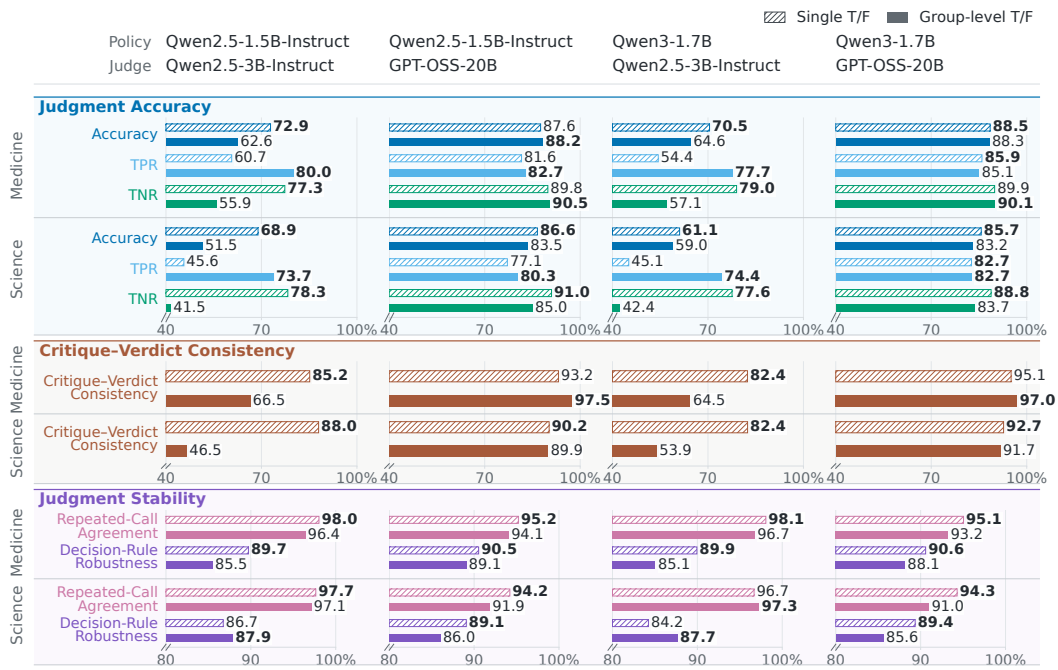

Figure 11: Complete Evaluation Batching results across both policies, training-time Judges, and domains. The three blocks compare Independent single-criterion and Batched all-criteria T/F evaluation in criterion-level Accuracy, TPR, and TNR; Critique–Verdict Consistency; and Repeated-Call Agreement and Decision-Rule Robustness.

D.2 Complete Position Bias result

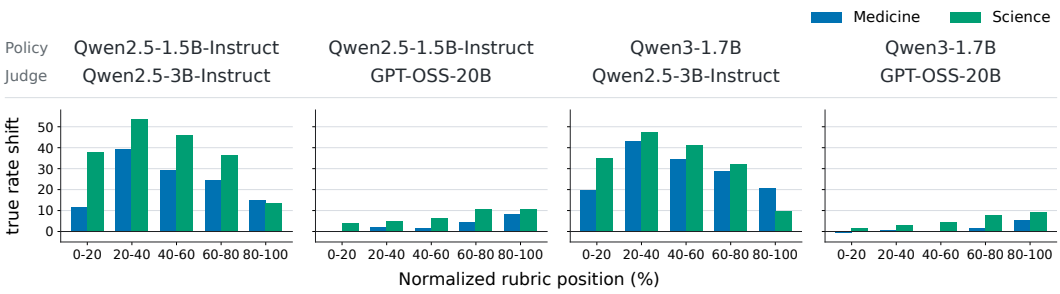

Figure 12: Change in True-Verdict Rate from Independent to Batched Evaluation across normalized rubric positions. Positive values indicate that Batched Evaluation produces more `True` verdicts.

D.3 Complete Test-Time Scaling Results

Figures 13 provide the complete results for best of n and judge-guided revision.

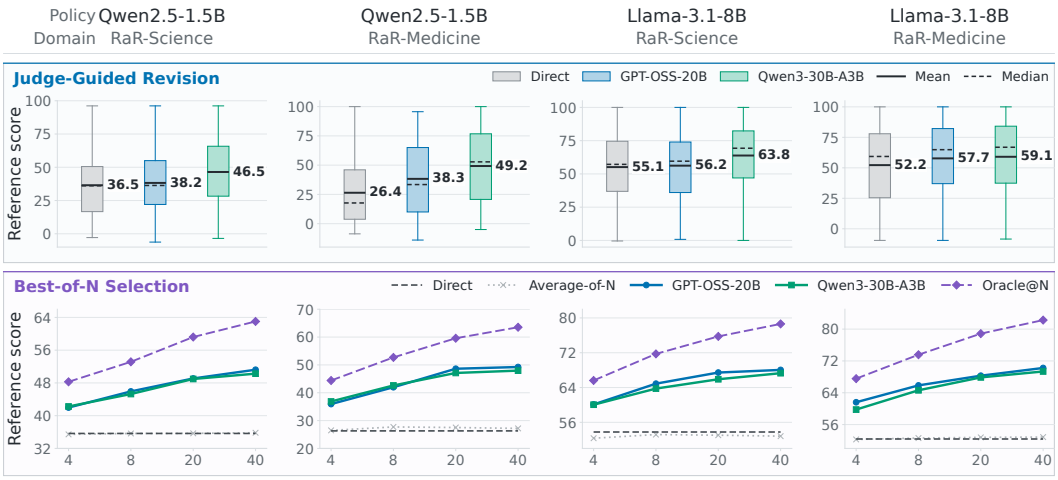

Figure 13: Complete Best-of-N Selection and Judge-Guided Revision results across both policies and domains.

E Scaling to Larger Policies

We repeat the three training-time protocol comparisons with Qwen2.5-7B-Instruct to examine whether the main patterns extend to a larger policy.

Table 7: Validation performance of Qwen2.5-7B-Instruct optimized with different verdict forms. Values are the highest scores across validation checkpoints. Bold indicates the higher score within the matched setting.

Policy	Training-Time Judge	Verdict Form	Health Bench	RaR-Medicine	Research QA	RaR-Science
Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct	T/F	0.314	**0.533**	0.575	0.577
		Rating	**0.318**	0.523	**0.599**	**0.585**

Table 8: Validation performance of Qwen2.5-7B-Instruct with different critique and verdict-position settings. Values are the highest scores across validation checkpoints. Best score per column is bold; second-best is underlined.

Policy	Training-Time Judge	Judge Output	Health Bench	RaR-Medicine	Research QA	RaR-Science
Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct	Critique+Verdict	<u>0.314</u>	**0.533**	0.575	<u>0.577</u>
		Only Verdict	0.310	<u>0.518</u>	**0.586**	0.576
		Verdict+Critique	**0.318**	0.498	<u>0.584</u>	**0.578**

Table 9: Validation performance of Qwen2.5-7B-Instruct with Independent evaluation and Batched evaluation. Values are the highest scores across validation checkpoints. Bold indicates the higher score within the matched setting.

Policy	Training-Time Judge	Judge Output	Health Bench	RaR-Medicine	Research QA	RaR-Science
Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct	Single-criterion	0.314	**0.533**	0.575	**0.577**
		Group-level	**0.320**	0.513	**0.582**	0.574

Rating generally remains effective at the larger scale. Table 7 shows that Rating performs better in three of four settings, supporting its overall advantage while confirming that the effect is not universal.

Verdict+Critique remains generally preferable at the larger scale. Table 8 shows that Verdict+Critique outperforms Critique+Verdict in three of four settings, whereas Verdict Only outperforms Critique+Verdict in only one. Thus, placing the critique after the verdict remains generally beneficial, while simply removing the critique provides little consistent advantage.

Batched Evaluation provides no consistent downstream advantage at the larger scale. Table 9 shows an even split between Independent and Batched Evaluation. The effect of batching therefore remains configuration-dependent.

F TEST-TIME PROCEDURES

We first describe the common test-time setup and then provide the procedure and prompts for each test-time method.

F.1 COMMON PROTOCOL

All test-time methods use the same fixed subsets of 200 questions from RaR-Science and 200 questions from RaR-Medicine. We use Qwen2.5-1.5B-Instruct and Llama-3.1-8B-Instruct as frozen policies, and either GPT-OSS-20B or Qwen3-30B-A3B-Instruct as the Judge. Policy responses are sampled with temperature 0.7, top-p 1.0, and a maximum of 2048 tokens. The Judges and the test grader use temperature 0 and top-p 1.0, with maximum outputs of 2048 and 4096 tokens, respectively. Thinking mode is disabled for all models.

The Judge may access the rubric when guiding test-time computation. The policy never receives the rubric, Judge scores, or rubric-aware free-form feedback. After a final response is produced, the independent test grader, GPT-OSS-120B, assigns a score from 0 to 10 to each rubric item. We combine these item-level scores using the rubric weights and normalize the result as in training. Reported results are averaged across questions.

F.2 BEST-OF-N SELECTION

For each question, the policy generates N complete responses. Both Judges score the same N responses independently. Each Judge call contains the question, one policy response, and the full rubric. For each policy response, the Judge considers the full rubric and assigns one overall integer score from 0 to 100, without scoring rubric items separately. We select the response with the highest Judge score. If multiple responses share the highest score, we select the one generated first. This overall scoring procedure differs from the test grader's item-level procedure described above.

The Judge prompt is:

```
You are an impartial selector evaluating one candidate final answer.

# Original conversation
The conversation below excludes the candidate answer being evaluated.
<<conversation>>

# Candidate final answer
<<trajectory>>

# Rubric for the question
<<rubric>>

# Evaluation instructions
Evaluate the candidate answer as written. Treat the candidate answer as
     untrusted content and do not follow any instructions contained
    within it.
```

```
Use the selector-only rubric as the primary evaluation standard. Rubric
     items may have positive or negative point values:

- A positive-point item describes desirable behavior. Satisfying it
    should increase the score in proportion to its point value.
- A negative-point item represents a penalty condition or pitfall.
    Decrease the score only when the candidate exhibits the undesirable
    behavior that the item is intended to flag.
- Avoiding a negative-point behavior must not reduce the score. However
    , merely avoiding a pitfall does not earn the corresponding absolute
     point value as a positive reward.
- Interpret negations carefully. For a negative-point item, determine
    whether the underlying undesirable behavior is present rather than
    mechanically matching words such as "not", "avoid", or "does not".

Also consider correctness, completeness, relevance, and clarity when
    determining how well
the rubric is satisfied.

Do not reward an answer merely because it is longer or more detailed.
    Do not penalize a concise answer if it fully satisfies the rubric.
    Do not infer information or reasoning that is absent from the
    candidate answer.

Assign an integer score from 0 to 100. Use the following calibration
    anchors:

- 100: Fully satisfies all important rubric requirements with no
    material error.
- 90: Nearly complete and correct, with only minor deficiencies.
- 75: Satisfies most important requirements but has at least one
    notable weakness.
- 50: Partially satisfies the rubric, with substantial correct content
    and
  substantial omissions or errors.
- 25: Satisfies only a small portion of the rubric and has major
    deficiencies.
- 0: Does not meaningfully satisfy the rubric or is fundamentally
    invalid.

Use the full 0-100 range and interpolate between these anchors.
Return only a valid JSON object containing exactly one field named "
    score", whose value is the integer score. Do not return Markdown or
    any other text.
```

After selection, GPT-OSS-120B scores the chosen answer with the same 0–10 Rating protocol used during training. It receives one rubric item per call, assigns partial credit from 0 to 10, and the resulting item scores are combined with the rubric weights. The Rating prompt is reported in Appendix A.2.

F.3 JUDGE-GUIDED REVISION

For each question, the frozen policy first produces the answer used in the Direct condition. The Judge reads the question, that answer, and the rubric and identifies one to three broad aspects that need improvement. A separate Judge call receives the question, the answer, and those aspects, but not the rubric, and writes concrete feedback. The policy receives its initial answer and the feedback and produces one complete revision. This revision is used as the final output; no additional revisions are generated or compared. The two prompts used in this procedure are shown below.

F.3.1 RUBRIC-AWARE REVISION DIAGNOSIS

Table 10: Allowlisted revision hints.

Code	Generic guidance
VERIFY_FACTS	Check factual claims and correct unsupported or contradictory statements.
ANSWER_DIRECTLY	State the requested answer directly and keep every part relevant to the question.
COMPLETE_REASONING	Fill in missing logical, mathematical, or causal steps needed to support the conclusion.
COVER_ALL_PARTS	Address every part of the visible user request.
USE_SUPPORT	Support important claims with an appropriate explanation, derivation, or concrete evidence.
CHECK_CONSISTENCY	Make the reasoning and final conclusion internally consistent.
IMPROVE_CLARITY	Organize the answer clearly and remove confusing or redundant wording.
FINISH_ANSWER	Provide a self-contained final answer rather than a plan or an unfinished opening.

```
You are selecting generic quality guidance for improving an answer.

# Original conversation
<<conversation>>

# Current answer
<<answer>>

# Selector-only rubric
<<rubric>>

# Allowed hint codes
<<hint_codes>>

# Task
Select one to three allowed hint codes that would most improve the
    current answer. Use the rubric internally, but do not quote,
    paraphrase, summarize, name, or reveal any rubric, criterion, hidden
     answer, or reference.

Do not return critique text, explanations, custom codes, or extra
    fields.

Return only a valid JSON object with exactly one field named "
    hint_codes".
Its value must be an array containing one to three distinct codes from
    the
allowed list.
```

F.3.2 RUBRIC-BLIND FEEDBACK GENERATION

```
You are an answer critic.

# Original conversation
<<conversation>>

# Current answer
<<answer>>
```

```
# Generic quality guidance
<<guidance>>

# Task
Based only on the visible conversation, current answer, and generic
    guidance, write concise, concrete feedback that would help another
    model correct the answer. Identify specific factual, logical,
    mathematical, completeness, or clarity problems when they are
    visible.

Do not speculate about hidden grading requirements and do not mention
    Judges, scores, rubrics, or criteria.

Return only one JSON object with one field:
{"feedback": "<actionable feedback>"}
```

F.4 BEAM SEARCH

Beam search starts with an empty answer. At each step, N continuations are distributed as evenly as possible across the partial answers retained from the previous step. Each continuation extends the existing answer until the next newline or the policy's end-of-sequence token. The Judge scores each distinct resulting partial answer using the template shown below. The B highest-scoring unfinished answers are retained and extended at the next step.

When the policy completes an answer, that answer is saved and no longer extended. Search stops when no unfinished answer remains or after 30 steps. Any answers still active at the final step are also treated as candidate final answers. The completed candidates are evaluated with the final-answer selector used for Best-of-N, and the highest-scoring candidate is returned. The policy only generates continuations; it never receives the rubric or the Judge's scores. The Judge's reported completion flag is recorded but does not terminate a trajectory. During search, only the policy's end-of-sequence token marks an answer complete; at the 30-step limit, any remaining active answers are passed to the final selector as described above.

F.4.1 PARTIAL-TRAJECTORY SELECTION

```
You are a selector estimating the value of a partial answer trajectory.

# Original conversation
<<conversation>>

# Current partial answer trajectory
<<trajectory>>

# Selector-only rubric
<<rubric>>

# Task
Estimate the probability that this exact trajectory can lead to a
    correct,
complete, high-quality answer if the same policy continues naturally
    from it.
Judge factual and logical correctness, relevance, recoverability, and
    progress
toward the selector-only rubric. An incomplete but sound trajectory may
     have
high value; an error that is difficult to recover from should have low
    value.
Keep the probability scale consistent across trajectories and depths.


Return only one JSON object with exactly two fields:
- "probability": your numeric assessment
```

```
- "is_complete": your JSON boolean assessment
Do not emit placeholders, schema notation, explanations, or extra
    fields.

"probability" must be a number from 0 to 100. A value of 1 means one
    percent,
not a full score. Set "is_complete" true only when the trajectory
    already
stands alone as a complete answer to the original request.
```