EVOJUDGE: DUAL EVOLUTION OF REUSABLE JUDGING SKILLS AND RUBRICS FOR LLM-AS-A-JUDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-as-a-judge plays an increasingly important role in model development, supporting fine-grained evaluation and providing reward signals for training. However, obtaining reliable judgments often requires highly capable frontier models, whose costs limit their use in large-scale evaluation and supervision. Inspired by recent advances in self-evolving agents, in this paper, we introduce **EvoJudge**, a dual evolution framework that improves LLM judges along two complementary dimensions: ***how to judge***, represented by procedure skills that guide evaluation strategies, and ***what to judge***, represented by criterion rubrics that specify evaluation standards for each type. Starting from a few labeled samples, EvoJudge reflects on original judgment and synthesizes the reusable experience into skill and rubric libraries. In addition, periodic validation detects performance regressions, triggers selective rollback and provides corrective guidance for subsequent evolution. Experiments across model families demonstrate obvious judging improvements, with Qwen3-30B-A3B gains 9.29%, 8.06%, and 9.63% on VerifyBench-hard, RewardBench2, and RMBench-hard, respectively, and enables Qwen3.5-35B-A3B to match frontier LLMs like GPT-5.6 Sol and Kimi-k3. Our analyses further demonstrate the complementary benefits of skills and rubrics, the effectiveness of periodic validation, and the transferability across various models.

1 INTRODUCTION

Reliable judgment is a fundamental component of modern large language model (LLM) development. It is essential for fine-grained model evaluation (Ye et al., 2024; Liu et al., 2025; Malik et al., 2026), where subtle differences in response quality need to be distinguished, and for model training (Cui et al., 2024; Yuan et al., 2024), where judgments often provide preference, reward, or verification signals. The strong capabilities of LLMs have made them increasingly useful as general-purpose evaluators (Gu et al., 2026), giving rise to the *LLM-as-a-Judge* paradigm, as Figure 1(a). Compared with conventional automatic metrics, LLM judges are particularly well suited to complex and open-ended tasks, e.g., creative writing, planning. (Shen et al., 2026a; Zheng et al., 2023).

Despite their broad applicability, reliable LLM-based judging remains challenging. Even capable models can exhibit inconsistent evaluation, apply inappropriate criteria, or produce weakly supported rationales (Zheng et al., 2023; Haldar & Hockenmaier, 2025; Shen et al., 2026b). A common practice is to rely on stronger frontier models as evaluators or reference annotators (Kim et al., 2024a; Zhu et al., 2025). However, such reliance creates a direct trade-off between judgment quality and computational cost, which becomes increasingly pronounced in large-scale evaluation and supervision generation (Wang et al., 2024; Xu et al., 2026b). Therefore, improving the judging capability of more accessible models remains an important direction.

Existing work mainly pursues this goal through model training or explicit evaluation criteria. One line of them improves judge models through supervised fine-tuning (Zhu et al., 2025), distillation from stronger evaluators, or reinforcement learning with preference supervision (Whitehouse et al., 2026). Such approaches can be effective, but typically require collecting or synthesizing judge-specific data pairs (Kim et al., 2024a). A different line of works improves judging by providing explicit evaluation criteria. Human-designed rubrics can offer reliable task specifications (Arora et al., 2025; Starace et al., 2025), but constructing and maintaining them across diverse evaluation settings is labor-intensive. Automatically generated rubrics via models reduce this burden (Hashemi

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.

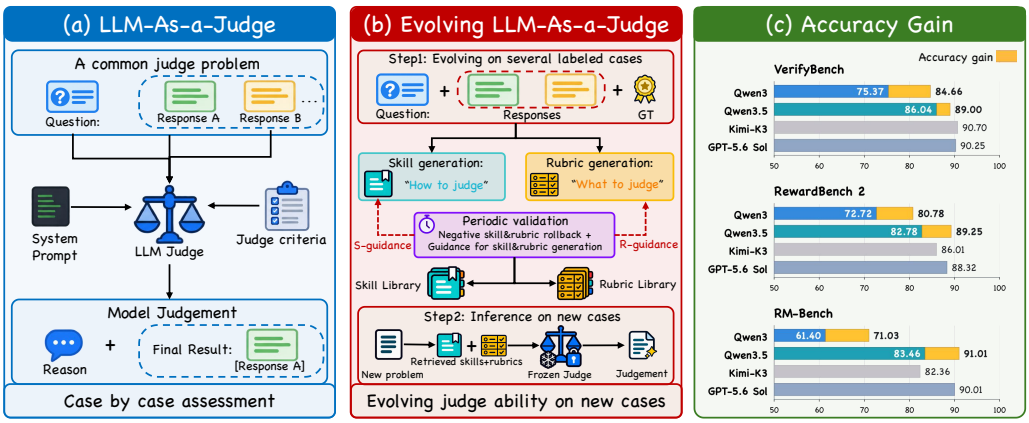


Figure 1: (a) Common paradigm of LLM-as-a-Judge, which conducts case by case assessment on every judging sample. (b) Our EvoJudge, which learns both skill and rubric libraries from few labeled samples by a dual-evolving framework, boosting its judging ability. (c) On three challenging judging benchmarks, EvoJudge substantially improves the Qwen-series models over their vanilla, matching or even surpassing Kimi-K3 and GPT-5.6-Sol.

et al., 2024; Shen et al., 2026b), but introduces an additional inference step and sometimes even over-claim case-specific requirements. More importantly, such guidance is typically generated per query, which offers limited explicit reuse across related cases.

Recent advances in self-evolving agents (Zhang et al., 2026a;c; Ouyang et al., 2026) offer a promising alternative, where models can extract recurring patterns from previous task executions and convert them into reusable experience, thereby improving future performance. Building on this premise, a natural question arises: can the self-evolution paradigm be adapted to LLM judging? Doing so is non-trivial because judging differs fundamentally from conventional problem solving. A successful judgment is not determined solely by a reasoning trajectory toward an answer. It depends both on how the evaluation is carried out (Saha et al., 2025) and on which properties of the candidate responses should determine the decision (Hashemi et al., 2024). Therefore, we characterize reusable judging experience along two interacting dimensions: ***how to judge***, which captures procedural knowledge about evaluation strategies and comparison processes, and ***what to judge***, which captures criterial knowledge about the qualities and evidence that should govern judgments.

Based on this consideration, we introduce **EvoJudge**, a dual evolution framework that enables a frozen LLM judge to accumulate reusable experience along both dimensions. As demonstrated in Figure 1(b), for ***how to judge***, the proposed framework learns ***procedure skills***, which capture generalizable evaluation procedures, including judging steps, identifying decisive evidence, and avoiding recurring reasoning errors. In parallel for figuring out ***what to judge***, it evolves ***criterion rubrics*** that capture fine-grained judging criteria and decision principles shared across a given class of tasks. Unlike instance-specific guidance, both forms of knowledge are progressively synthesized and refined from multiple judging experiences and therefore can be reused across future new samples.

Specifically, starting from a few labeled samples, **EvoJudge** allows the judge to make predictions, reflect on judgment results, and consolidate useful experience into evolving skill and rubric libraries. Since self-generated experience may be noisy, EvoJudge further incorporates periodic validation to identify detrimental updates and derive corrective feedback for subsequent evolution. When periodic performance degrades, EvoJudge reverts negative updates and incorporates the resulting failure signal into later evolution. In this way, unlike EvalSkill (Yue et al., 2026), which only explores evaluations into a unified domain-level skill, EvoJudge can not only compose two dimensions of reusable knowledge, but also regulate the quality of the knowledge being accumulated.

We conduct extensive experiments and find that our EvoJudge can consistently improve models' judging capability across different model families, with Qwen3-30B-A3B achieving gains of $+9.29\%$, $+8.06\%$, $+9.63\%$ on VerifyBench-hard, RewardBench2, and RMBench-hard. As shown in Figure 1(c), it further enables Qwen3.5-35B-A3B to surpass existing state-of-the-art systems like Kimi-K3 and GPT-5.6-Sol in judgment. Ablation studies show that jointly evolving judging skills

and rubrics outperforms evolving either component alone, supporting the complementary roles of procedural and criterial experience. Furthermore, the evolved experience transfers effectively across model scales and backbones, indicating that EvoJudge captures reusable judging knowledge rather than model-specific heuristics. Our contributions are threefold:

- We formulate dual-evolution for LLM judge as the joint accumulation of two forms of reusable experience: *procedure skills* for ***how to judge*** and *criterion rubrics* for ***what to judge***. This formulation captures the distinct procedural and criterial knowledge required for reliable assessment.
- We introduce **EvoJudge**, a framework for jointly evolving judging skills and rubrics while continuously regulating the quality of accumulated experience. The incorporated periodic validation is able to identify detrimental updates and derive corrective guidance for subsequent evolution.
- Comprehensive experiments demonstrate that **EvoJudge** consistently obtains judging improvements across multiple benchmarks and models. Extensive analyses further show that the two forms of experience are complementary in effectiveness, transferable across model scales and architectures, and applicable to a diverse range of judging tasks.

2 Related Work

LLM-as-a-Judge. LLM-as-a-Judge (Gu et al., 2026) provides flexible and convenient evaluation for open-ended assessment (Shen et al., 2026a). One line of approaches improves the evaluator through judge-specific training. JudgeLM (Zhu et al., 2025) learns from GPT-4-generated judgments, while Prometheus (Kim et al., 2024a) trains evaluators to assess responses under user-defined criteria. More recent works (Kim et al., 2024b) explicitly optimize judgment reasoning. EvalPlanner(Saha et al., 2025) learns to generate and execute evaluation plans through preference optimization, whereas J1(Whitehouse et al., 2026) uses verifiable judgment rewards to improve deliberative evaluation through reinforcement learning. These training approaches demonstrate the value of structured reasoning and criterion awareness for judging. Instead, EvoJudge develops these capabilities through reusable external experience while keeping the judge's parameters fixed.

Rubric-guided Evaluation. Rubric-guided approaches make the dimensions and standards of evaluation explicit. G-Eval combines evaluation criteria with generated reasoning steps to support structured assessment (Liu et al., 2023). HealthBench uses physician-written, conversation-specific criteria (Arora et al., 2025), similar to PaperBench (Starace et al., 2025). To reduce manual construction, automatic methods (Hashemi et al., 2024) are explored. (Xie et al., 2025)introduce a Propose–Evaluate–Revise paradigm, while (Zhang et al., 2026b) optimize rubrics by identifying minor differentials between high-quality responses. RRD proposes initial rubrics from a task prompt and sample responses, then recursively decomposes, filters, and weights the criteria (Shen et al., 2026b). However, approaches that construct rubrics for individual instances incur repeated generation costs. EvoJudge instead learns from criterion observations, making rubrics reusable across related cases.

Self-evolving Agents. Experience-based self-improvement enables agents to refine their behavior through external knowledge without necessarily updating model parameters. ExpeL(Zhao et al., 2024) extracts reusable insights from accumulated task experiences. ACE (Zhang et al., 2026c) further develops this direction by maintaining evolving contextual knowledge through generation, reflection, and curation. Recent skill-based systems provide more structured forms of experience: AutoSkill(Yang et al., 2026) derives and maintains skills from interaction traces, while XSkill(Jiang et al., 2026) separates action-level experiences from task-level skills. MetaClaw(Xia et al., 2026) synthesizes skills from failure trajectories for immediate prompt-level adaptation. In the field of evaluation, Eval-Skill(Yue et al., 2026) uses exploration and selection to construct domain-level evaluation skills. Differently, EvoJudge focuses on dual forms of knowledge: both skills and rubric experiences, and regulates them through selective rollback and corrective guidance.

3 Method

3.1 Overview and Problem Formulation

Overview. We introduce **EvoJudge**, a dual evolution framework that improves a frozen LLM judge by accumulating two complementary forms of reusable judging experience: *procedure skills* for *how to judge* and *criterion rubrics* for *what to judge*. As illustrated in Figure 2, EvoJudge evolves these

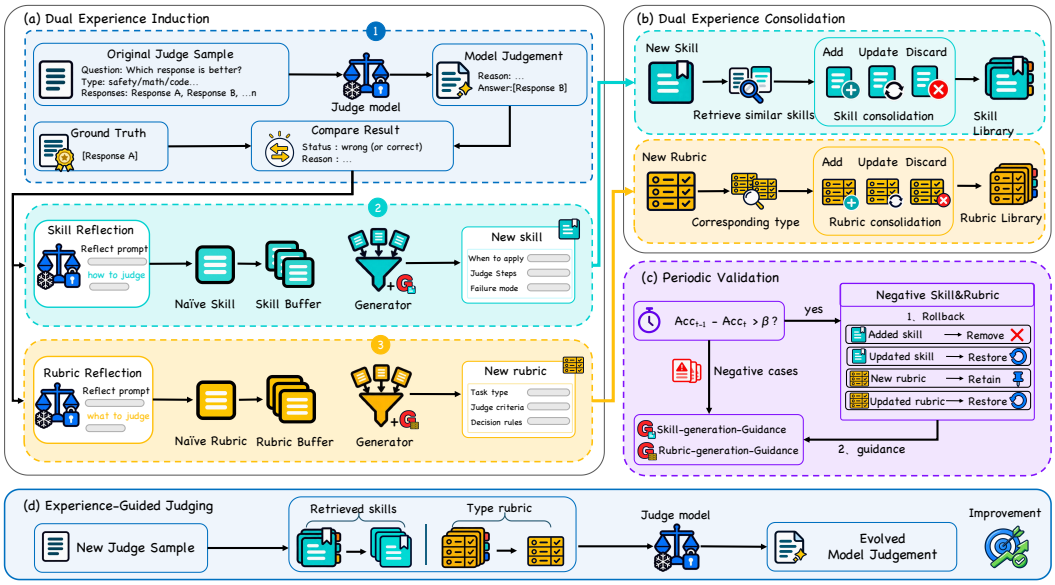


Figure 2: Overview of EvoJudge. (a) **Dual Experience Induction:** Original judgment result guides naive skill and rubric reflections, which are buffered and abstracted into reusable experiences. (b) **Dual Experience Consolidation:** New experiences are consolidated into the library through add, update, or discard operations. (c) **Periodic Validation:** Performance regressions trigger selective library rollback and corrective guidance for subsequent evolution. (d) **Experience-Guided Judging:** Retrieved skills and the task-type rubric jointly guide a new judgment.

two forms of experience through three stages: (a) dual experience induction, (b) dual experience consolidation, and (c) periodic validation. The resulting knowledge is subsequently reused to guide judgments on new cases. The parameters of the judge model remain fixed throughout both evolution and inference.

Problem formulation. Let $x_i = (q_i, \mathcal{C}_i, \tau_i)$ denotes a judging instance, where q_i is the evaluation question, $\mathcal{C}_i$ contains the candidate responses, and τ_i denotes the task type defined in different benchmarks, and each instance is paired with a ground truth y_i^*. We denote the frozen LLM judge by J, the judging-skill library by $\mathcal{S}$, and the rubric library by $\mathcal{R}$. The two libraries encode distinct dimensions of judging experience. A skill $s \in \mathcal{S}$ represents reusable evaluation procedures, comparison strategies, and recurring failure modes. The rubric library maintains one active rubric for each task type τ, specifying the evaluation criteria and decision principles, denoted by $r_\tau = \mathcal{R}[\tau]$.

3.2 Dual Experience Induction

Each judgment exposes two complementary forms of reusable experience: procedural evidence concerning the evaluation process, and criterion-level evidence concerning the basis of the decision. EvoJudge accordingly factorizes reflection into a skill branch, which extracts procedural experience, and a rubric branch, which extracts evaluation criteria and decision principles.

Firstly, for each labeled judging sample, the frozen LLM first produces an original judgment: $(z_i, \hat{y}_i) = J(x_i)$ and $e_i = \mathbb{I}[\hat{y}_i = y_i^*]$, where z_i denotes the judging reason, $\hat{y}_i$ is the predicted answer, and e_i indicates whether the prediction is equal to the ground truth.

Secondly, the resulting judgment reason and correctness signal of this sample are then examined through two separate reflection processes:

$$s_i = F_s(x_i, z_i, e_i), \quad r_i = F_r(x_i, z_i, e_i).$$

The skill reflection F_s analyzes how the evaluation was conducted and extracts reusable procedures or failure patterns relevant to *how to judge*, obtaining the naive skill s_i. In parallel, the rubric reflection F_r identifies the criteria and decision principles that should govern *what to judge*, generating the naive rubric r_i. Both successful and failed judgments provide informative signals. Successful cases

reveal effective evaluation strategies and well-aligned criteria, whereas failed cases expose procedural weaknesses and missing or misleading criteria. Details for reflection are in Appendix A.5.

Thirdly, Considering that instance-level experiences are often entangled with instance-specific details and are therefore unsuitable for direct reuse (Yue et al., 2026). EvoJudge accumulates naive skills and rubrics in separate buffers, $\mathcal{B}_s$ and $\mathcal{B}_r$, and abstracts each batch of eight naive entries into several new skills and rubrics:

$$\widetilde{\mathcal{S}} = G_s(\mathcal{B}_s; \mathcal{S}, g_s), \quad \widetilde{\mathcal{R}} = G_r(\mathcal{B}_r; \mathcal{R}, g_r).$$

Here, G_s extracts recurring procedural patterns from buffered naive skills and synthesizes them into candidate skills. The rubric generator G_r instead organizes buffered naive rubrics by task type and aggregates criteria shared across related instances. Both generators are conditioned on the current library state and branch-specific corrective guidance g_s and g_r. The two branches operate at distinct levels of abstraction. The skill branch extracts cross-instance procedural regularities that may transfer across evaluation contexts, whereas the rubric branch aggregates task-conditioned criteria that remain stable across instances of the same type.

3.3 Dual Experience Consolidation

Since directly appending newly induced experience would introduce redundancy, conflicting guidance, and overly instance-specific knowledge, EvoJudge therefore consolidates each new experience against the corresponding existing library before admission. Skills encode procedures that may transfer across task types and are therefore matched according to semantic relevance. Rubrics encode evaluation criteria for a particular task type and are matched through type-indexed lookup:

$$\mathrm{Match}_s(\tilde{s}) = \mathrm{TopK}(\tilde{s}, \mathcal{S}), \quad \mathrm{Match}_r(\tilde{r}_\tau) = \mathcal{R}[\tau].$$

Skill consolidation. For each candidate skill $\tilde{s} \in \widetilde{\mathcal{S}}$, EvoJudge retrieves the K_c most similar entries from library $\mathcal{S}$ and jointly compares the candidate against the retrieved skills. The consolidator then selects one of three operations. ADD inserts the candidate when it contributes distinct procedural knowledge. UPDATE incorporates useful information into an existing skill when the candidate extends or refines previously acquired experience. DISCARD rejects candidates that are redundant or insufficiently supported. This retrieval-based consolidation promotes procedural diversity while limiting repeated or overlapping entries.

Rubric consolidation. For each candidate rubric $\tilde{r}_\tau \in \widetilde{\mathcal{R}}$, EvoJudge directly accesses the active rubric r_τ associated with the same task type. If no such type exists, the candidate initializes a new rubric for that type. Otherwise, the consolidator either incorporates reliable new criteria and decision principles into the existing rubric or discards the candidate when it provides no substantive refinement. Repeated consolidation thereby transforms instance-level criterion observations into a progressively refined rubric shared across related judging cases.

3.4 Periodic Validation

Self-evolution introduces a potential stability challenge (Liao et al., 2026): the experience may be erroneous and such error may be recursively reinforced as newly accumulated experience influences subsequent judgments and further evolution. EvoJudge addresses this issue by validation regulation. Specifically, periodic validation thus operates at two levels. Rollback restores a reliable state of the accumulated knowledge, while corrective guidance reshapes subsequent experience induction.

At evolution round t, we evaluate the judge augmented with the current libraries on a held-out validation set $\mathcal{V}$: $A_t = \mathrm{Acc}\,(J, \mathcal{S}_t, \mathcal{R}_t; \mathcal{V})$. We maintain the validation accuracy $A_{\mathrm{t}-1}$ in last round $t-1$. The updates in this round are considered detrimental when $A_{\mathrm{t}-1} - A_t > \beta$, where β denotes the tolerated degradation and is set to 0.02 in the experiment.

Negative experience rollback. If no obvious regression is detected, the libraries generated in this round are accepted and establish the new library state. Otherwise, EvoJudge selectively reverts updates introduced during the current period. Newly added skills are removed, while modified skills and rubrics are restored to their previous versions. Newly introduced rubrics are retained when they correspond to task types for which no prior rubric exists.

Learning from negative evolution. Performance regressions also provide informative negative evidence for subsequent evolution. Rather than discarding a failed interval entirely, EvoJudge analyzes how recent library updates altered judgments that were previously reliable and converts this information into corrective guidance. Specifically, we identify validation instances that transition from correct to incorrect from last round to this round: $\mathcal{F}_t^- = \{x \in \mathcal{V} \mid \hat{y}_{t-1}(x) = y^*, \hat{y}_t(x) \neq y^*\}$ .These newly corrupted judgments are analyzed together with the skill and rubric updates introduced during the corresponding interval. The two branches maintain separate corrective guidance:

$$g_s \leftarrow H_s(g_s, \Delta\mathcal{S}_t, \mathcal{F}_t^-), \quad g_r \leftarrow H_r(g_r, \Delta\mathcal{R}_t, \mathcal{F}_t^-).$$

The skill branch identifies procedural changes associated with the regression, whereas the rubric branch diagnoses problematic criteria or decision rules. The resulting guidance is subsequently supplied to G_s and G_r and is accumulated across negative intervals, steering future induction away from previously observed failure patterns. Detailed prompts for generating guidance and examples of generated guidance are in Appendix A.6 and A.7.

3.5 Experience-Guided Judging

Once evolution is complete, both libraries are fixed and can be applied to new inferences. As Figure 2(d), given a new sample $x = (q, \mathcal{C}, \tau)$, EvoJudge retrieves procedural skills by semantic relevance and rubric knowledge by task type: $\mathcal{S}_x = \mathrm{Retrieve}(q, \mathcal{S}; K_s), r_\tau = \mathcal{R}[\tau]$. The retrieved skills provide reusable guidance on *how to judge*, while the selected rubric specifies *what to judge*. Both forms of experience are incorporated into a single judging call:$(z, \hat{y}) = J(x; \mathcal{S}_x, r_\tau)$. EvoJudge thus decouples experience evolution from test-time judging. Judging knowledge is induced, consolidated, and regulated during evolution, then directly reused at inference without updating model parameters or regenerating evaluation standard for each individual instance.

4 Experiment

4.1 Experimental Setting

Models and Baselines. We evaluate EvoJudge with Qwen3-30B-A3B, Qwen3.5-35B-A3B, and GPT-5.6 Sol, spanning different model backbones and sizes. (Yang et al., 2025; Qwen Team, 2026a; OpenAI, 2026a). Additional Qwen3.5 models, Qwen3.5-9B, Qwen3.5-27B are used for cross-model transfer, the details of the evaluated models are in Appendix A.1. Our implementation builds on MetaClaw's skill-only framework (Xia et al., 2026) and uses Qwen3-Embedding-0.6B (Zhang et al., 2025) for embedding-based skill retrieval. We compare against the rubric-generation methods, the LLM Rubric and recursive rubric decomposition (RRD) from Shen et al. (2026b), and skill-based methods AutoSkill(Yang et al., 2026), MetaClaw (Xia et al., 2026) and EvalSkill (Yue et al., 2026).

Benchmarks and Evaluation. We evaluate on VerifyBench-Hard, RewardBench 2, and RM-Bench (Yan et al., 2026; Malik et al., 2026; Liu et al., 2025), which test answer verification, preference judgments across instruction-following and reasoning tasks, and sensitivity to response quality under stylistic variation, respectively. We use the most difficult pairs for RM-Bench. For each benchmark, we use 200, 100 samples for induction and periodic validation. The remaining samples are reserved for testing inference, about over 1000. Accuracy is averaged over three repeated evaluations. Output-token counts measure generated response length.

4.2 Main results

Dual evolution improves judgement across models and benchmarks. As demonstrated in Table 1, the EvoJudge consistently improves both Qwen backbones across all three benchmarks, with average accuracy gains of 8.99 and 5.66 points over the Qwen3-30B-A3B and Qwen3.5-35B-A3B baselines, respectively. Compared with rubric-based methods (LLM Rubric and RRD) and skill-based methods (AutoSkill, MetaClaw, EvalSkill), EvoJudge achieves higher accuracy in most benchmarks, with only EvalSkill yielding larger gains in a few cases. These results support the effectiveness of jointly evolving procedural skills and evaluation rubrics. Notably, LLM Rubric underperforms the baseline in several settings, indicating that query-generated rubrics may impose overly specific requirements, which is also mentioned in previous work (Shen et al., 2026b). Examples of how skills and rubrics improve responses are in Appendix A.8.

Table 1: Results on VerifyBench, RewardBench2, and RMBench. Accuracy is reported as Acc. ± Std.(%), where standard deviation is computed over repeated runs. Green and red parenthesized values denote absolute gains or drops over the baseline. Red downward arrows indicate reductions in output tokens over the baseline. Bold denotes the highest acc, – denotes method is inapplicable.

Model / Method	VerifyBench-Hard		RewardBench2		RMBench		Average	
	Acc. ± Std.	Tokens	Acc. ± Std.	Tokens	Acc. ± Std.	Tokens	Acc. (%)	Tokens
Frontier LLMs								
Kimi-K3	90.70 ±0.32	693.01	86.01 ±0.67	1605.32	82.36 ±0.64	173.32	86.36	823.88
GLM-5.2	88.75 ±0.20	892.67	80.31 ±0.95	1833.99	83.31 ±0.00	196.68	84.12	974.45
DeepSeek-V4-Flash	83.50 ±0.93	648.35	73.50 ±1.20	796.06	70.18 ±0.45	348.41	75.73	597.61
DeepSeek-V4-Pro	89.00 ±0.51	106.22	75.09 ±0.80	357.44	73.05 ±0.22	203.48	79.05	222.38
Qwen3.6-35B-A3B	89.62 ±0.35	1909.76	82.17 ±0.40	1887.52	86.29 ±0.14	113.80	86.03	1303.69
Qwen3.8-27B	**90.95** ±0.22	501.12	76.77 ±0.69	1693.32	87.08 ±0.40	15.85	84.93	736.76
GPT-5.6 Sol	90.25 ±0.24	96.81	88.32 ±0.14	309.31	90.01 ±0.22	24.44	89.53	143.52
GPT-5.6 Sol (Ours)	90.00 ±0.39 (-0.25)	91.60 ↓	**90.19** ±0.20 (+1.87)	344.53	**92.90** ±0.44 (+2.89)	155.16	**91.03** (+1.50)	197.10
Method								
Qwen3-30B-A3B								
Baseline	75.37 ±0.88	493.42	72.72 ±0.79	533.60	61.40 ±0.49	583.64	69.83	536.89
LLM Rubric	–	–	70.76 ±0.42	180.43	70.71 ±0.36	857.97	–	–
RRD	–	–	71.31 ±0.26	194.98	72.40 ±0.30	898.49	–	–
AutoSkill	82.87 ±0.62	488.37	76.28 ±0.53	832.32	56.78 ±0.10	21.91	71.98	447.53
MetaClaw	81.41 ±0.57	423.65	74.60 ±0.55	610.53	54.45 ±0.17	154.79	70.15	396.32
EvalSkill	83.31 ±0.69	628.71	78.96 ±0.32	1552.06	**73.78** ±0.19	1670.29	78.68	1283.69
EvoJudge (Ours)	**84.66** ±0.37 (+9.29)	450.16 ↓	**80.78** ±0.59 (+8.06)	673.09	71.03 ±0.22 (+9.63)	215.42 ↓	**78.82** (+8.99)	446.22 ↓
Qwen3.5-35B-A3B								
Baseline	86.04 ±0.41	3678.13	82.78 ±0.26	2409.77	83.46 ±0.24	2587.36	84.09	2891.75
LLM Rubric	–	–	79.46 ±0.30	3931.16	78.73 ±0.42	5192.41	–	–
RRD	–	–	83.66 ±0.32	4476.83	84.51 ±0.20	6459.39	–	–
AutoSkill	88.38 ±0.30	5015.78	87.61 ±0.14	2858.09	87.32 ±0.14	2620.38	87.77	3498.08
MetaClaw	87.79 ±0.82	3258.77	84.60 ±0.40	1928.70	84.26 ±0.10	523.42	85.55	1903.63
EvalSkill	86.69 ±0.20	2564.68	87.30 ±0.28	2722.18	89.49 ±0.22	562.67	87.83	1949.84
EvoJudge (Ours)	**89.00** ±0.45 (+2.96)	2585.92 ↓	**89.25** ±0.53 (+6.47)	2042.67 ↓	**91.01** ±0.48 (+7.55)	2313.07 ↓	**89.75** (+5.66)	2313.89 ↓

Evolved judges surpass frontier-model performance. Compared with frontier-models,in average, the evolved Qwen3.5-35B-A3B (89.75%) surpasses GLM-5.2(84.12%), Kimi-K3 (86.36%) and GPT-5.6 Sol (89.53%). We further apply dual evolution to GPT-5.6 Sol, to examine whether EvoJudge can improve an already highly capable evaluator. As shown in Table 1, the evolved GPT-5.6 Sol obtains a 1.50-point improvement averagely, despite a modest 0.25-point decrease on VerifyBench-Hard, demonstrating that the framework can also strengthen frontier judges. Meanwhile, across Qwen3-30B-A3B, Qwen3.5-35B-A3B, and GPT-5.6 Sol, their average gains of our method decrease from 8.99 to 5.66 to 1.50-point. This pattern suggests that initially weaker judges may have greater room to benefit from evolved experience.

Accuracy improvements do not require longer judge outputs on Qwen backbones. Beyond accuracy, we examine how EvoJudge affects generated output length at inference time. As shown in Table 1, EvoJudge improves judging accuracy for both Qwen backbones while reducing generated output tokens in five of the six settings. On average, output length decreases by 16.89% for Qwen3-30B-A3B and by 19.98% for Qwen3.5-35B-A3B. This pattern suggests that, for the Qwen backbones, the accuracy gains are not driven by longer judgment rationales. At the same time, GPT-5.6 Sol exhibits a different trend, with increased output length after applying EvoJudge.

4.3 Effectiveness of Periodic Validation

We compare EvoJudge with periodic validation enabled-on and disabled-off. Figure 3(a) shows that enabling periodic regulation increases accuracy in all six benchmark–backbone settings. The additional gains average 2.55 and 2.96 points across the three benchmarks for Qwen3-30B-A3B and Qwen3.5-35B-A3B, respectively. And feedback regulation benefits are slightly more pronounced on RewardBench2 and RM-Bench. Thus, monitoring and revising accumulated experience provides benefits beyond unregulated accumulation. More details can be found in Appendix A.3.

Figure 3(b) further presents the periodic regulation trajectories of Qwen3-30B-A3B. In general, the regulated evolution achieves higher accuracy than unregulated evolution. In the regulation-

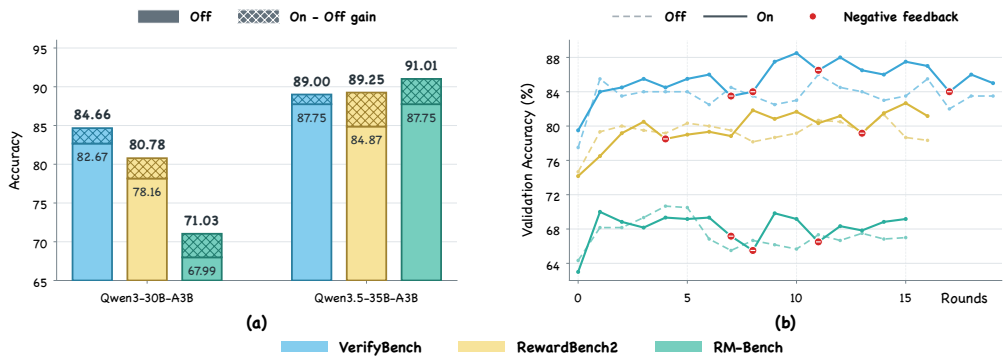


Figure 3: (a) Accuracy with periodic validation turned on and off for two backbones; hatched segments show the additional gains. (b) Periodic evaluation trajectories for Qwen3-30B-A3B. Dark and light lines denote validation on and off; red markers denote triggered interventions.

on trajectories, red markers indicate rounds where validation accuracy drops beyond the threshold relative to the previous round, triggering selective rollback and corrective guidance. Following these interventions, accuracy often rebounds at the next evolution period, even after a substantial decline, with relatively few interventions later. In contrast, the regulation-off trajectories show more consecutive declines in accuracy. These patterns suggest that timely library correction and guidance help reduce the persistent effects of negative experience and support more stable evolution.

4.4 Complementarity Between Skills and Rubrics

To examine the contributions of evolved skills and rubrics, we compare three inference settings: retrieving only skills, only rubrics, and both. As shown in Table 2, skill-only and rubric-only settings can also generally improve the base model, but their combination, EvoJudge, achieves the highest accuracy in most benchmarks. The complementary gain is consistent with the framework's distinction: skills organize the evaluation process , while rubrics specify which response properties should determine the judgment. Appendix A.4 illustrates details of libraries. From the perspective of the individual contributions of skills and rubrics, their relative effectiveness varies across benchmarks and backbones. Notably, on RM-Bench, using skills alone slightly reduces the accuracy of Qwen3.5-35B-A3B, but combining both achieves 91.01%, exceeding the rubric-only setting 89.88%. This further suggests that the two libraries can interact complementarily, rather than contributing independently in a simply additive manner.

Table 2: Ablation of skill and rubric complementarity. Results are Acc. (%).

Model	Method	Verify	Reward2	RM	Average
Qwen3 30B-A3B	Baseline	75.37	72.72	61.40	69.83
	Skill-only	**84.79**	77.30	70.89	77.66
	Rubric-only	81.83	78.48	68.17	76.16
	EvoJudge	84.66	**80.78**	**71.03**	**78.82**
Qwen3.5 35B-A3B	Baseline	86.04	82.78	83.46	84.09
	Skill-only	88.66	84.64	82.75	85.35
	Rubric-only	87.81	85.30	89.88	87.66
	EvoJudge	**89.00**	**89.25**	**91.01**	**89.75**

4.5 Cross-model Transfer of Judging Experience

We investigate the transferability of EvoJudge's evolved experience across model families and across model sizes. On RewardBench 2, we independently evolve skill and rubric libraries using Qwen3.5-9B, Qwen3.5-35B-A3B, Qwen3.5-27B, and GPT-5.6 Sol, then apply each source model's libraries to every target model at inference time. Figure 4 reveals two main findings.

Source–target compatibility matters. For each target model, the highest accuracy is achieved when using its own evolved experience, as indicated by the diagonal entries. This suggests that self-evolved experience is better matched than that from other models. Moreover, a source model with higher judging accuracy does not necessarily generate more effective experience for other models. For example, Qwen3.5-9B achieves 84.70% accuracy using the libraries from Qwen3.5-35B-A3B,

compared with 82.24% using those from GPT-5.6 Sol. These results suggest that the effectiveness of generated experience depends on source–target compatibility, rather than the source capability alone.

Sensitivity to the source varies across targets. Looking at each column, GPT-5.6 Sol's accuracy varies by only 1.07 percentage points across the four experience sources, whereas Qwen3.5-9B exhibits a spread of 2.57 points. In this comparison, the stronger judge is less sensitive to the source of experience. One possible explanation is that stronger models can more flexibly interpret and apply guidance from different sources, while weaker models depend more heavily on how well the supplied experience matches their judging capabilities.

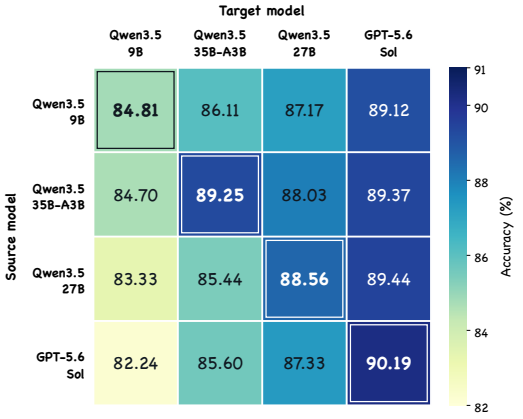


Figure 4: RewardBench2 transfer accuracy (%). Rows are the source, columns the target.

4.6 Analysis on category of judging tasks

We further compare judging accuracy before and after evolution across task categories. As shown in Figure 5. EvoJudge improves accuracy in most categories in the three benchmarks, indicating that its benefits extend across diverse judging tasks rather than being confined to a single task type. What's more, the results between benchmarks also reveal a shared pattern: gains in math problems are limited on both RewardBench2 and RM-Bench, whereas improvements are more pronounced in categories such as chat, factuality, and safety. One possible explanation is that judging tasks involving specialized knowledge, such as mathematics and coding, rely more heavily on the model's underlying domain knowledge and reasoning capabilities, which may be harder to improve through reusable experience alone. More details of category results are in Appendix A.2

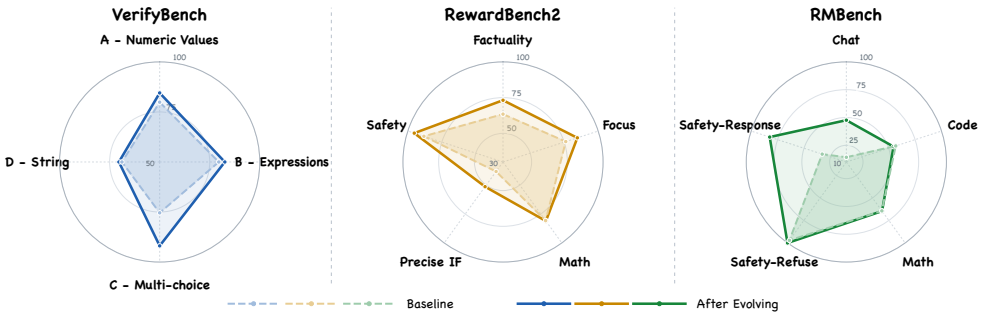


Figure 5: Category-level accuracy of the baseline and EvoJudge. Light dashed and dark solid lines represent the baseline and After Evolving, respectively.

5 Conclusion

We presented EvoJudge, a dual evolution framework that improves frozen LLM judges through *procedure skills* for how to judge and *criterion rubrics* for what to judge. From a small set of labeled examples, EvoJudge reflects on judgment outcomes and consolidates reusable knowledge, while periodic validation regulates evolution through selective rollback and corrective guidance. Experiments across three benchmarks and multiple backbones demonstrate substantial judging improvements, bringing Qwen3.5-35B-A3B to performance competitive with frontier judges on the evaluated tasks. Further analyses support the complementarity of skills and rubrics, cross-model transferability, and the effectiveness of periodic validation. These findings highlight reusable evaluation experience as a practical means of strengthening LLM judges without updating model parameters.

AI USE STATEMENT

We used generative AI tools to improve the readability of the manuscript and inspect code. LLMs are also integral to the proposed method: they generate judgments, reflections, experience candidates, and corrective summaries. The authors reviewed the AI-assisted writing and code inspection and take responsibility for the final manuscript and all research outputs, including those generated by LLMs.

REPRODUCIBILITY STATEMENT

Section 3 specifies the induction, consolidation, regulation, and inference procedures. To support reproducibility, Appendices A.1–A.4 provide model details, detailed results, ablations, and library statistics. Appendices A.5–A.8 provide reflection and corrective guidance prompts, generated guidance, and qualitative examples. The code will be available in `https://anonymous.4open.science/r/EvoJudge-6671`.

ETHICS STATEMENT

The proposed method does not appear to raise substantial ethical concerns. The datasets and models used in the experiments are also publicly available.

REFERENCES

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. HealthBench: Evaluating large language models towards improved human health, 2025. URL https://arxiv.org/abs/2505.08775.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. ULTRAFEEDBACK: Boosting language models with scaled AI feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 9722–9744. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cui24f.html.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Zhouchi Lin, Bowen Zhang, Lionel Ni, Wen Gao, Yuanzhuo Wang, and Jian Guo. A survey on LLM-as-a-judge. *The Innovation*, 7 (6):101253, 2026. doi: 10.1016/j.xinn.2025.101253. URL https://doi.org/10.1016/j.xinn.2025.101253.

Rajarshi Haldar and Julia Hockenmaier. Rating roulette: Self-inconsistency in llm-as-a-judge frameworks. *arXiv preprint arXiv:2510.27106*, 2025.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 13806–13834, 2024.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents, 2026. URL https://arxiv.org/abs/2603.12056.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2024a. URL https://arxiv.org/abs/2310.08491.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, 2024b. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Junfeng Liao, Qizhou Wang, Jianing Zhu, Bo Du, Rui Yan, and Xiuying Chen. Belief memory: Agent memory under partial observability. *arXiv preprint arXiv:2605.05583*, 2026.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, 2023. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. RM-Bench: Benchmarking reward models of language models with subtlety and style. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=QEHrmQPBdd.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fb0G86Dewb.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. Official model release, July 2026a. URL https://openai.com/index/gpt-5-6/.

OpenAI. GPT-5.6 Sol model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol. Accessed September 22, 2026.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Qwen Team. Qwen3-30B-A3B model card, 2025. URL https://huggingface.co/Qwen/Qwen3-30B-A3B. Accessed September 22, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.5-35B-A3B model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.5-35B-A3B. Accessed September 22, 2026.

Qwen Team. Qwen3.6-35B-A3B model card, 2026c. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B. Accessed September 22, 2026.

Qwen Team. Qwen3.8-27B-FP8 model card, 2026d. URL https://huggingface.co/Qwen/Qwen3.8-27B-FP8. Accessed September 22, 2026.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. Learning to plan & reason for evaluation with Thinking-LLM-as-a-Judge. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2501.18099.

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*, 2026a.

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, 2026b. URL https://arxiv.org/abs/2602.05125.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5Nn2BLV7SB.

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://proceedings.iclr.cc/paper_files/paper/2026/hash/118c2d372db0939c7dcd716ebe6447b2-Abstract-Conference.html.

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. MetaClaw: Just talk—an agent that meta-learns and evolves in the wild, 2026. URL https://arxiv.org/abs/2603.17187.

Lipeng Xie, Sen Huang, Zhuo Zhang, Anni Zou, Yunpeng Zhai, Dingchao Ren, Kezun Zhang, Haoyuan Hu, Boyin Liu, Haoran Chen, et al. Auto-rubric: Learning to extract generalizable criteria for reward modeling. *arXiv e-prints*, pp. arXiv–2510, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026a.

Zhenghao Xu, Qin Lu, Qingru Zhang, Liang Qiu, Ilgee Hong, Changlong Yu, Wenlin Yao, Yao Liu, Haoming Jiang, Lihong Li, et al. Ask a strong llm judge when your reward model is uncertain. *Advances in Neural Information Processing Systems*, 38:74639–74664, 2026b.

Yuchen Yan, Jin Jiang, Zhenbang Ren, Yijun Li, Xudong Cai, Yang Liu, Xin Xu, Mengdi Zhang, Jian Shao, Yongliang Shen, Jun Xiao, and Yueting Zhuang. VerifyBench: Benchmarking reference-based reward systems for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JfsjGmuFxz.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026. URL https://arxiv.org/abs/2603.01145.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=3OfPKwAqPf.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Xing Yue, Linjuan Wu, Daoxin Zhang, Yongliang Shen, and Weiming Lu. Beyond rubrics: Exploration-guided evaluation skills for reward modeling, 2026. URL https://arxiv.org/abs/2606.07040.

Z.ai. GLM-5.2 model card, 2026. URL https://huggingface.co/zai-org/GLM-5.2. Accessed September 22, 2026.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026a.

Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. Chasing the tail: Effective rubric-based reward modeling for large language model post-training. In *International Conference on Learning Representations*, volume 2026, pp. 133430–133457, 2026b.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026c.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, volume 2025, pp. 51257–51296, 2025.

A APPENDIX

A.1 DETAILS OF EVALUATED MODELS.

Table 3 summarizes the evaluated models, including their parameter sizes, weight availability, developers, and official references. We distinguish total and activated parameters for mixture-of-experts models and indicate unavailable or version-dependent information where applicable. EvoJudge operates on the same backbones without updating their parameters.

Table 3: Details of the evaluated backbone models. Parameter counts distinguish total and activated parameters where available. Open-weight indicates publicly available model weights.

Model	**Total / Active Params.**	**Weight Availability**	**Developer**	**Reference**
Kimi-K3	2.8T / 104B	Open-weight	Moonshot AI	(Team et al., 2026)
GLM-5.2	753B	Open-weight	Z.ai (Zhipu AI)	(Z.ai, 2026)
DeepSeek-V4-Flash	284B / 13B	Open-weight	DeepSeek	(Xu et al., 2026a)
DeepSeek-V4-Pro	1.6T / 49B	Open-weight	DeepSeek	(Xu et al., 2026a)
Qwen3.6-35B-A3B	35B / 3B	Open-weight	Alibaba (Qwen Team)	(Qwen Team, 2026c)
Qwen3.8-27B	27B (dense)	Open-weight	Alibaba (Qwen Team)	(Qwen Team, 2026d)
GPT-5.6 Sol	–	Closed-weight	OpenAI	(OpenAI, 2026b)
Qwen3-30B-A3B	30.5B / 3.3B	Open-weight	Alibaba (Qwen Team)	(Qwen Team, 2025)
Qwen3.5-35B-A3B	35B / 3B	Open-weight	Alibaba (Qwen Team)	(Qwen Team, 2026b)

A.2 DETAILED ACCURACY AMONG DIFFERENT CATEGORIES

The evaluation results for each category are shown in Table 4, covering three datasets across different models and methods. For VerifyBench, NV, Exp, MC, and Str denote Numeric Values, Expressions, Multi-choice, and String, respectively. For RewardBench2, Fact, Foc, PIF, and Safe denote Factuality, Focus, Precise IF, and Safety, respectively. For RMBench, SRef and SResp denote Safety Refuse and Safety Response, respectively.

Table 4: Results on VerifyBench, RewardBench2, and RMBench among different category. Accuracy is reported as a percentage (%). – denotes that a method is inapplicable.

Model / Method	VerifyBench					RewardBench2						RMBench					
	NV	Exp	MC	Str	Avg	Fact	Foc	Math	PIF	Safe	Avg	Chat	Code	Math	SRef	SResp	Avg
Frontier LLMs																	
Kimi-K3	**85.97**	85.24	95.54	88.95	90.70	82.59	**91.01**	89.05	67.17	89.69	86.01	43.49	57.85	91.19	99.93	88.42	82.36
GLM-5.2	81.52	**87.14**	93.99	87.50	88.75	77.78	88.41	84.17	51.84	82.73	80.31	40.00	73.20	89.31	97.93	87.22	83.31
DeepSeek-Flash	78.16	75.92	87.76	84.29	83.50	69.85	80.89	68.59	45.75	81.26	73.50	19.09	59.79	74.83	98.13	61.28	70.18
DeepSeek-Pro	84.98	83.81	92.54	88.77	89.00	71.93	85.08	72.43	44.76	79.44	75.09	21.82	57.04	85.75	97.10	52.38	73.05
Qwen3.6-35B-A3B	85.31	84.76	94.57	86.96	89.62	76.42	87.95	88.27	46.62	92.30	82.17	40.45	80.41	93.65	**100.00**	83.08	86.29
Qwen3.8-27B-FP8	85.64	83.81	**95.74**	90.58	**90.95**	64.23	85.83	84.77	41.49	89.68	76.77	55.45	78.35	93.88	99.79	80.08	87.08
GPT-5.6 Sol	84.65	82.14	94.91	90.76	90.25	89.15	88.98	90.74	74.13	90.86	88.32	57.27	85.31	96.44	98.55	86.84	90.01
GPT-5.6 Sol + Ours	85.14	85.71	92.73	**91.84**	90.00	**90.07**	90.35	**91.47**	**79.71**	**93.42**	**90.19**	**72.27**	**87.11**	**96.99**	98.76	**93.98**	**92.90**
Method																	
Qwen3-30B-A3B																	
Baseline	78.99	80.07	77.12	66.33	75.37	63.24	76.45	**82.82**	37.71	87.30	72.72	13.61	56.02	65.57	95.83	32.29	61.40
LLM Rubric	-	-	-	-	–	60.06	75.45	81.07	34.50	85.96	70.76	21.73	**70.23**	75.70	92.28	56.00	70.71
RRD	-	-	-	-	–	60.22	75.61	82.30	34.50	87.31	71.31	31.21	66.88	**78.69**	93.20	55.59	72.40
AutoSkill	81.35	82.86	88.86	73.37	82.87	71.31	77.35	75.93	39.63	93.91	76.28	11.82	49.48	53.67	98.76	39.10	56.78
MetaClaw	81.35	**83.33**	87.40	69.57	81.41	67.77	74.70	80.04	41.49	91.62	74.60	10.00	45.88	52.49	98.34	30.83	54.45
EvalSkill	79.93	82.28	87.82	**78.98**	83.31	72.95	82.68	78.18	47.70	92.94	78.96	**66.49**	59.42	78.27	77.67	78.55	**73.78**
EvoJudge (Ours)	**84.91**	82.84	**92.79**	69.89	**84.66**	**73.17**	**84.63**	81.04	**51.35**	**95.24**	**80.78**	47.71	54.77	64.72	**100.00**	**82.59**	71.03
Qwen3.5-35B-A3B																	
Baseline	80.18	77.07	92.01	84.73	86.04	76.73	84.70	91.28	54.24	94.00	82.78	40.04	77.43	92.13	98.08	72.40	83.46
LLM Rubric	-	-	-	-	–	68.30	77.04	89.36	63.41	93.92	79.46	16.13	79.75	90.64	96.78	56.57	78.73
RRD	-	-	-	-	–	71.46	81.66	**95.79**	69.49	97.45	83.66	20.58	**85.38**	**96.64**	98.83	62.65	84.51
AutoSkill	**84.79**	**87.68**	90.88	88.25	88.38	82.12	87.27	91.59	**77.17**	94.92	87.61	60.30	82.71	94.10	99.45	70.96	87.32
MetaClaw	82.84	83.33	90.89	**89.13**	87.79	80.03	84.32	90.12	61.54	95.94	84.60	45.45	76.46	91.83	**100.00**	73.68	84.26
EvalSkill	81.19	83.57	90.41	86.96	86.69	80.66	91.93	89.81	63.99	96.70	87.30	72.73	78.87	93.76	**100.00**	85.34	89.49
EvoJudge (Ours)	82.34	83.81	**94.86**	87.32	**89.00**	**84.33**	**92.85**	94.80	62.57	**97.94**	**89.25**	**80.61**	78.52	93.76	99.86	**92.48**	**91.01**

A.3 Ablation of rollback-guidance in periodic validation

We further examine the contributions of negative-experience rollback and corrective guidance in in periodic validation.. Table 5 compares evolution without periodic validation, validation with rollback only, and the full mechanism combining rollback with guidance, on Qwen3-30B-A3B. The rollback alone achieves an average accuracy gain of 2.16 percentage points over evolution without periodic validation, while adding guidance increases the gain to 2.55 points. These results suggest that both operations contribute to effective evolution, with rollback providing larger incremental improvement in this comparison.

Table 5: Ablation results on VerifyBench, RewardBench2, and RMBench. Accuracy is reported as a percentage. Δ denotes the absolute accuracy gain in percentage points over the Off setting.

Setting	VerifyBench		RewardBench2		RMBench		Average	
	Acc. (%)	Δ	Acc. (%)	Δ	Acc. (%)	Δ	Acc. (%)	Δ
Off	82.67	–	78.16	–	67.99	–	76.27	–
On (only rollback)	83.83	+1.16	79.93	+1.77	**71.55**	+3.56	78.44	+2.16
On (rollback + guidance)	**84.66**	+1.99	**80.78**	+2.62	71.03	+3.04	**78.82**	+2.55

A.4 Details of skill and rubric library

Table A.4 also reports the size and average usage of the evolved libraries. EvoJudge maintains compact libraries containing 19–54 entries and uses only 6.34–11.16 entries per judgment on average. Skills provide multiple relevant procedures, whereas rubric retrieval supplies at most one type-level entry. Despite this selective use of experience, EvoJudge achieves the highest average accuracy on both backbones, supporting the effectiveness of combining reusable procedural guidance with task-specific evaluation criteria.

Table 6: Ablation of skill and rubric complementarity. Acc. denotes accuracy (%), Library denotes the number of stored experience entries, and Used denotes the average number of entries used per judgment. Average accuracy is computed across the three benchmarks.

Model	**Method**	**VerifyBench**			**RewardBench 2**			**RM-Bench**			**Average**
		Acc.	Library	Used	Acc.	Library	Used	Acc.	Library	Used	Acc.
Qwen3 30B-A3B	Baseline	75.37	–	–	72.72	–	–	61.40	–	–	69.83
	Skill-only	**84.79**	38	5.34	77.30	49	8.12	70.89	47	8.98	77.66
	Rubric-only	81.83	14	1.00	78.48	5	1.00	68.17	5	1.00	76.16
	EvoJudge	84.66	44	6.34	**80.78**	54	9.12	**71.03**	52	9.98	**78.82**
Qwen3.5 35B-A3B	Baseline	86.04	–	–	82.78	–	–	83.46	–	–	84.09
	Skill-only	88.66	27	10.35	84.64	15	5.68	82.75	24	8.28	85.35
	Rubric-only	87.81	11	0.81	85.30	5	1.00	89.88	5	1.00	87.66
	EvoJudge	**89.00**	40	11.16	**89.25**	19	6.68	**91.01**	29	9.28	**89.75**

A.5 Prompts for Experience Reflection

Using RewardBench 2 as an example, we present four prompts for experience induction and corrective guidance. Each training instance in RewardBench2 involves selecting the best response among four candidates. The reflection prompts use the question, candidate responses, previous judgment, ground-truth choice, and correctness status to extract reusable judging knowledge.

Skill reflection prompt. The skill reflection prompt produces structured procedural guidance, including judging steps, applicability conditions, failure modes, and cautions. The rubric reflection prompt instead specifies evaluation goals, criteria, and decision rules, including conditions for selecting or rejecting a response. Both prompts encourage generalization beyond the individual training case, providing preliminary experience for subsequent buffered abstraction and library consolidation.

Prompt 1: Skill Reflection

You are generating one compact type-specific reusable judging skill according to previous training cases.
Purpose of judging skill:

- Improve future judgments for question type: {subset}
- The judging question is a four-choice judge question: choose the best answer among Assistant A/B/C/D.
- Do not solve or summarize this case-specific exact example.

Write a compact skill that generalizes from the following judging cases.
The previous judging task is: Impartially evaluate four AI responses to the user's question and select the best one based on instruction-following, helpfulness, relevance, accuracy, depth, creativity, and detail. Briefly compare the responses without order, length, or name bias. Then output one verdict: [[A]], [[B]], [[C]], or [[D]].
User question={question_head}
Assistant A ={answer_a_head}
Assistant B ={answer_b_head}
Assistant C ={answer_c_head}
Assistant D ={answer_d_head}
As for the specific task,
The previous model judgment is choice: {predicted_label}, {judgment_final_line}
And the ground-truth best choice is: [[{gold_label}]]
So, the previous result of model judgment is: {result_status}
Now, based on the case (question type: {subset}) above and the previous judging result, carefully think about why this type of question was judged correctly or incorrectly, and then generate one reusable judging skill in the format below.
Start your response directly with specific skill, Return only this skill format, beginning with the exact tag:
[RewardBench2/{subset}]
Core reusable judging steps: ...
Trigger / When to apply: ...
Do not / Failure mode: ...
Watch out for: ...

Do not write any analysis, thinking process, explanation, or summary. Return only your reusable judging skill in the required format.

Rubric reflection prompt. The rubric reflection prompt uses the previous judgment and its correctness signal to derive type-specific evaluation criteria and decision rules. It defines the target of comparison, conditions for selecting or rejecting a response, and common judging mistakes. These criteria generalize beyond the individual case and provide reusable guidance on *what to judge*.

Prompt 2: Rubric Reflection

You are drafting type-level rubrics for judgement.
Purpose of judging rubrics:

- Improve future judgments for question type: {subset}
- You should write judge-facing rubric and criteria, to guide future judging about this type of cases.
- Rubric should generalize across this type. Do not create an instance-level rubric.

Write a judging rubric that generalizes from the following judging cases.
The previous judging task is: Impartially evaluate four AI responses to the user's question and select the best one based on instruction-following, helpfulness, relevance, accuracy, depth, creativity, and detail. Briefly compare the responses without order, length, or name bias. Then output one verdict: [[A]], [[B]], [[C]], or [[D]].
User question={question_head}
Assistant A={answer_a_head}
Assistant B={answer_b_head}
Assistant C={answer_c_head}
Assistant D={answer_d_head}

As for the specific case,
The previous model judgment is choice: {predicted_label}, {judgment_final_line}
And the ground-truth best choice is: [[{gold_label}]]
So, the previous result of model judgment is: {result_status}
Now, based on the case above and the previous judging result, carefully consider what kind of judging rubric should be used for this type of question in order to make more accurate judgments and choices, and then generate type-level reusable rubrics in the format below.
Use exactly these Markdown sections:
Grading Goal
Judge criteria / Target Comparison Aspect
Decision rules
When a choice content should be selected
When a choice content should be rejected
Common Judge Mistakes
Do not write any analysis, thinking process, explanation, or summary. Return only your judging rubric in the required format.

A.6 CORRECTIVE GUIDANCE PROMPT.

Specifically, periodic validation is triggered every 10% of the total evolution progress or when the combined number of skill and rubric modifications since the last validation exceeds 15. When the performance degradation exceeds the threshold, rollback and corrective guidance are triggered for both skills and rubrics. The two corrective prompts support future experience generation when periodic validation detects a regression. The skill prompt summarizes problematic procedural patterns associated with newly incorrect judgments, while the rubric prompt identifies potentially misleading criteria and decision rules. Each produces a general caution of fewer than 200 words, excluding case-specific questions, answers, and option labels. These cautions guide the corresponding generators toward avoiding recurring failure patterns in subsequent skill and rubric induction.

Prompt 1: Guidance from Negative Skill Updates

You are an expert in analyzing why judging skills have negative effects—that is, why several judging questions changed from being answered correctly to being answered wrongly after the following newly added or updated skills were used.
Your task is to summarize what should be avoided when generating skills in the future. Focus on the pattern of skills that caused the negative effect, not specific skill instances.
Do not mention specific questions, answers, option labels, exact wording, or specific examples. Write one general caution for skill generation in under 200 words. Return only the caution text.

Corrective rubric guidance prompt.

Prompt 2: Guidance from Negative Rubric Updates

You are an expert in analyzing why newly added or updated evaluation rubrics may negatively affect judgments—that is, why several judging questions changed from being answered correctly to being answered wrongly after these rubric changes.
Your task is to summarize what should be avoided or corrected when generating rubrics in the future. Focus on patterns in evaluation criteria and decision rules that may explain the regression, rather than specific rubric instances or evaluation procedures.
Do not mention specific questions, answers, option labels, exact wording, or specific examples. Write one general caution for rubric generation in under 200 words. Return only the caution text.

A.7 EXAMPLES OF GENERATED CORRECTIVE GUIDANCE

We present examples of corrective guidance generated by the skill branch during periodic regulation, taking verifybench-hard on Qwen3-30B-A3B as an example. These summaries are supplied to the skill generator to guide how subsequent skills formulate evaluation procedures. Rather than merely recording previous errors, they translate observed failure patterns into actionable instructions for future skill induction. For example, the guidance below encourages newly generated skills to check

task requirements, extract and normalize the final answer, and assess equivalence before rejecting a response. It also discourages character-level matching and formatting restrictions unless explicitly required by the task. By incorporating these instructions, subsequent skills can preserve valid comparison procedures while avoiding overly restrictive rules that may incorrectly reject equivalent answers. The accumulated guidance thus helps reduce the recurrence of procedural errors across evolution rounds.

Generated guidance for skill induction. We present examples of corrective guidance generated by the skill branch during periodic regulation. These summaries are supplied to the skill generator to guide how subsequent skills formulate evaluation procedures. Rather than merely recording previous errors, they translate observed failure patterns into actionable instructions for future skill induction. For example, the guidance below encourages newly generated skills to check task requirements, extract and normalize the final answer, and assess equivalence before rejecting a response. It also discourages character-level matching and formatting restrictions unless explicitly required by the task. By incorporating these instructions, subsequent skills can preserve valid comparison procedures while avoiding overly restrictive rules that may incorrectly reject equivalent answers. The accumulated guidance thus helps reduce the recurrence of procedural errors across evolution rounds.

Example 1: Equivalence over Form

During evaluation, follow this procedure: first read the task instructions and confirm whether exact form is explicitly required. If it is not explicitly required, do not reject based on formatting, notation, or structural preferences. Then compare the candidate answer with the reference answer for semantic, mathematical, or logical equivalence, including simplification, equivalent transformations, $\pm$ signs, LaTeX presentation, labels, and layout differences. Mark the answer incorrect only when the core meaning is wrong, the reasoning is contradictory, the conclusion is not equivalent, or the task explicitly requires exact formatting and that requirement is not met; otherwise, accept it.

Example 2: Extract and Compare the Final Answer

When the reference answer is a concise result, such as a boxed number, symbol, or value, the evaluation steps should be: first extract the final conclusion from the candidate answer; then ignore non-essential formatting differences such as capitalization, spacing, parentheses, labels, and LaTeX wrapping; finally compare whether the numerical value, symbolic form, or logical conclusion is equivalent. If the final conclusion is equivalent, do not mark it wrong merely because the expression differs. Do not perform character-level exact matching unless the task explicitly states that character-for-character identity is required.

Example 3: Normalize before Comparison

Judgment rules should first perform necessary extraction and normalization before equivalence comparison; do not start by applying complex formatting rules or direct string matching. If the reference answer is intended to express a concept or allows flexible wording, focus on whether the meaning, logic, and result are correct rather than on capitalization, punctuation, structure, or syntactic details. If the candidate includes brief necessary reasoning, verify whether it supports the conclusion instead of rejecting it for not being a bare string.

Generated guidance for rubric induction. These illustrative examples show how corrective guidance can shape subsequent rubric generation. Supplied to the rubric generator, the guidance clarifies which response properties should determine correctness, which requirements should remain conditional, and which conflicting rules should be revised. In particular, it encourages future rubrics to accept valid equivalent expressions, avoid generalizing instance-specific constraints to an entire task type, and maintain consistent, checkable decision criteria. These constraints help the generator preserve meaningful evaluation standards while avoiding unsupported rejection conditions. Accumulating such guidance across evolution rounds can therefore reduce the recurrence of misleading criteria and support more reliable type-level rubrics.

Example 1: Equivalence as the Acceptance Criterion

For answer-equivalence tasks, define correctness by whether the candidate preserves the meaning, mathematical result, or logical conclusion required by the task. Revised rubrics should explicitly permit equivalent expressions and distinguish substantive discrepancies from presentational differences. Capitalization, spacing, notation, labels, and LaTeX wrapping should not independently determine correctness unless the task explicitly makes them relevant. Reject answers for changed meaning, missing required solutions, incorrect bounds, or incompatible conclusions, rather than for differing from the reference's surface form.

Example 2: Type-Level Criteria with Explicit Scope

When revising a type-level rubric, retain criteria that generalize across instances of that type and express instance-dependent requirements conditionally. For interval judgments, correctness concerns equivalent bounds and matching endpoint inclusion; do not require every interval to exclude both endpoints because one observed example is an open interval. For equation judgments, distinguish equivalence of the requested equations or solution sets from similarity of notation. Avoid embedding particular numbers, symbols, or answer forms as universal requirements. Each rejection condition should identify a substantive violation of the task's requirements.

Example 3: Consistent and Checkable Decision Rules

Future rubric revisions should resolve conflicting acceptance and rejection conditions. A rubric that accepts mathematically equivalent expressions should not also require character-for-character identity unless exact representation is an explicit task requirement. Replace vague conditions such as "matches the reference exactly" with checkable properties, including equivalent values, complete solution sets, correct units when required, and consistent boundary conditions. Preserve legitimate task constraints while removing unsupported restrictions, so that the same valid response is not accepted by one criterion and rejected by another.

A.8 Examples of How Skills and Rubrics Improve Judging

The recorded qualitative examples show two changes in final-answer assessment after joint skill and rubric augmentation. In Case 1, the original judge accepts a final label of `proved` against a reference of `disproved` by interpreting the candidate's explanation as its intended answer; the evolved judge instead rejects the explicit state mismatch. In Case 2, the original judge credits a discussion that never selects an option, whereas the evolved judge identifies the missing final selection. The examples distinguish reusable procedures from task-level criteria. Skills emphasize locating, extracting, and comparing the final answer; rubrics specify the answer object and acceptance or rejection conditions. Underlining marks relevant clauses and contrasting judgments, while boldface marks headings and final answers or verdicts.

Case 1: Skill 1

[VerifyBench/C4 - Finite state selection]
Skill: extract-final-state-from-verbose-outputs
Core reusable judging steps: 1. Locate answer markers such as "final answer", a boxed label, or an explicit concluding statement. 2. Extract the candidate's final asserted state separately from its preceding explanation. 3. Associate that state with the proposition asked about in the question, retaining negation and any explicit uncertainty. 4. Compare the extracted state with the reference state; if the explanation and final label disagree, record the conflict rather than silently replacing the final label.
Trigger / When to apply: Use when a finite-state answer contains lengthy reasoning, multiple intermediate conclusions, or a final label that may conflict with the explanation.
Do not / Failure mode: Do not substitute a more plausible answer inferred from the reasoning for the answer the candidate actually commits to. Do not treat intermediate state labels as final selections merely because they match the reference.
Watch out for: A response may describe a negative outcome and then select "proved". Determine which proposition the final label refers to. The fact that a negative outcome is established does not make "proved" the correct label for the corresponding affirmative proposition.

Case 1: Skill 2

[VerifyBench/C4 - Finite state selection]
Skill: avoid-semantic-overinterpretation-in-finite-state-judgment
Core reusable judging steps: 1. Identify the finite-state schema used by the task and the proposition being evaluated. 2. Remove presentation-only wrappers, such as boxing or emphasis, when the task permits them. 3. Map the candidate and reference labels to their meanings within the same schema. 4. Compare the selected states without changing the candidate's intended proposition or repairing its label. 5. Base the verdict on the state comparison rather than on how convincing the surrounding explanation appears.
Trigger / When to apply: Use when candidate and reference labels differ, or when the surrounding explanation makes an incompatible final label appear superficially acceptable.
Do not / Failure mode: Do not reinterpret "proved" as "disproved" to make the candidate agree with the reference. Do not confuse a semantic contradiction with a harmless difference in capitalization, punctuation, or LaTeX presentation.
Watch out for: "Proved", "disproved", and "unknown" represent different states for the same proposition. Equivalent presentation can be normalized when permitted, but opposite states remain distinct after normalization.

Case 1: Rubric

Grading Goal
Determine whether the candidate's final answer selects the same logical state as the reference for the proposition specified by the task. Evaluate the final answer independently of the quality or plausibility of the preceding derivation.
Judge criteria / Target Comparison Aspect
The target is the final asserted state of the queried proposition, selected from the task's finite-state schema, such as "proved", "disproved", or "unknown". Relevant aspects are state identity, polarity, definiteness, and the presence of an unambiguous final selection.
Decision rules
States that are logically opposite or express different levels of certainty are not equivalent. An explicit final label governs the comparison even when the preceding explanation suggests a different outcome. Formatting differences are irrelevant when they do not alter the selected state and the task does not require an exact surface form.
When a choice content should be selected
Return Yes when the candidate's unambiguous final answer denotes the same state as the reference for the same proposition. Accept allowed variations in capitalization, punctuation, wording, and LaTeX presentation when the underlying state is unchanged. A flawed explanation does not invalidate a matching final answer under a final-answer-only evaluation protocol.
When a choice content should be rejected
Return No when the candidate selects a different state, provides no identifiable final state, or leaves the final selection unresolved. A final label of "proved" does not match a reference of "disproved" merely because the explanation describes a negative outcome. A plausible interpretation of what the candidate might have intended does not replace its explicit answer.
Common Judge Mistakes
Accepting a contradictory final label by treating it as a harmless labeling error. Confusing a proof of a negative statement with the truth of the affirmative proposition being queried. Rejecting a correct state solely because its presentation differs. Allowing reasoning quality to override final-state agreement.

Case 1: Original model output

The reference answer is "disproved", which means that the claim "the crab hides the cards that she has from the chihuahua" is false—i.e., the crab does not hide the cards.
The solution to be evaluated concludes that "the crab does not hide the cards that she has from the chihuahua", which is equivalent to saying that the crab does not hide the cards—thus, the statement "the crab hides the cards" is disproved.
Even though the solution uses the word "proved" in its final box (which may be a mislabeling of the conclusion), the actual logical conclusion is that the crab does not hide the cards, which aligns with the reference answer "disproved"—because the answer to the question "does the crab hide the cards..." is no, hence it is disproved.

The key point is that the final outcome—the crab does not hide the cards—matches the implication of "disproved" in the reference answer. The phrasing "proved" in the solution may be a labeling error, but the content of the conclusion is consistent with the reference.
Furthermore, the reasoning in the solution, while possibly flawed in its application of rule preferences (e.g., incorrectly stating Rule1 is not preferred over Rule2 when Rule1 is preferred and applies), is not being judged for correctness of process—only for agreement in final result.
The final result in the solution is that the crab does not hide the cards, which is equivalent to saying the claim that the crab hides the cards is disproved.
Thus, the answer to be evaluated is essentially the same as the reference answer in terms of the final outcome.
Final Judgment: Yes

Case 1: Evolved model output

The reference answer is: **disproved**
The solution to be evaluated concludes with: $\boxed{\text{proved}}$
We now analyze the final state selected by the candidate answer in comparison to the reference answer.

- The reference answer is **disproved**.
- The candidate answer ends with **proved**.

Even though the reasoning process may contain errors (e.g., incorrectly applying Rule1 vs Rule2, or misjudging the preference hierarchy), the task requires us to judge only the final result and whether it matches the reference answer in essence, ignoring process errors.
According to the guidelines:

- The two answers must be essentially the same in meaning and logical outcome.
- "proved" and "disproved" are semantic opposites and are not equivalent.
- The candidate answer selects a different finite state from the reference.

Therefore, despite the candidate's reasoning and formatting (e.g., use of `\boxed{\text{proved}}`), the final state label does not match the reference answer.
Final Judgment: No

Case 2: Skill 1

[VerifyBench/C1 - Single Choice]
Skill: extract-final-choice-label-with-format-preserving
Core reusable judging steps: 1. Locate the candidate's concluding answer statement, explicit option label, or unambiguous selection of an option's content. 2. Preserve the selected option's identity while separating it from surrounding reasoning and mentions of alternative choices. 3. If the candidate gives option content rather than a label, check whether it maps uniquely to one listed option under the task's equivalence rules. 4. If no final selection is present, record the answer as missing rather than inferring a choice from a general discussion.
Trigger / When to apply: Use when a single-choice response contains lengthy analysis, describes a solution strategy, discusses several alternatives, or states its answer in a different permitted presentation.
Do not / Failure mode: Do not mistake an option mentioned in an explanation for the option actually selected. Do not complete the candidate's answer by deciding which option its unfinished reasoning might eventually support.
Watch out for: A statement that the expressions need further simplification is a description of a procedure, not a final selection. Conversely, naming the full content of one option can constitute a selection when the mapping is unique and the task permits semantic equivalence.

Case 2: Skill 2

[VerifyBench/C1 - Single Choice]
Skill: verify-multi-choice-exact-match
Core reusable judging steps: 1. Identify the reference option and the candidate's extracted final selection. 2. Check whether the candidate commits to exactly one option, distinguishing the final answer from intermediate hypotheses or references to other choices. 3. Compare option identity after any nor-

malization explicitly permitted by the task. 4. Check answer presence before comparing correctness: a response without a final selection cannot match the reference choice. 5. Apply character-level matching only when the task explicitly requires an exact surface form.
Trigger / When to apply: Use after final-answer extraction in a single-choice task, especially when plausible reasoning or lengthy explanatory text could obscure a missing, ambiguous, or different selection.
Do not / Failure mode: Do not award credit merely because the response discusses a reasonable strategy or does not explicitly contradict the reference. Do not impose additional formatting constraints that the evaluation task does not require.
Watch out for: A correct option, an incorrect option, multiple unresolved options, and no selected option are different answer states. Judge the state actually expressed by the candidate rather than reconstructing a preferred answer from its explanation.

Case 2: Rubric

Grading Goal
Determine whether the candidate commits to the same single option as the reference. Evaluate the identity and completeness of the final selection rather than the plausibility of the accompanying analysis.
Judge criteria / Target Comparison Aspect
The target is one identifiable final option, expressed through its label or through content that unambiguously identifies that option when the task permits equivalent expression. Relevant aspects are whether an answer is provided, whether the selection is unique, and whether it matches the reference option.
Decision rules
A response must make a determinate selection to receive credit. Discussion of a strategy, recognition of the task's difficulty, or reasoning that could lead to the reference does not itself constitute an answer. Option identity governs the comparison unless the task additionally requires an exact output format.
When a choice content should be selected
Return Yes when the candidate explicitly or unambiguously selects the reference option. Accept the option's full content in place of its label when the mapping is unique and the task permits semantic equivalence. Under a final-answer-only protocol, errors in the preceding reasoning do not change a matching final selection into an incorrect answer.
When a choice content should be rejected
Return No when the candidate selects another option, leaves several options unresolved, or does not select any option. A general explanation of how to solve the problem is insufficient when it stops before making a choice. Do not treat lack of explicit disagreement with the reference as evidence of a matching answer.
Common Judge Mistakes
Inferring the reference option from a response that never commits to it. Granting credit for thoughtful discussion without a completed answer. Confusing a mentioned option with a selected option. Rejecting a uniquely identified correct option solely because its permitted presentation differs from the reference.

Case 2: Original model output

The reference answer is (E). The solution to be evaluated concludes with an analysis that, despite the complexity and the inclusion of non-mathematical statements, attempts to reason through the structure and logical form of the expressions. It acknowledges the difficulty of full simplification but emphasizes that the correct answer must be the one that, when fully evaluated, results in a true expression, considering both mathematical operations and the truth values of factual statements (e.g., capitals of countries).
The key point in the judgment is that the final result of the solution to be evaluated is not explicitly stated as (E), but the reasoning process is consistent with identifying (E) as the correct choice through structural and logical analysis, even without full step-by-step simplification. The solution does not contradict the reference answer; rather, it aligns with the expectation that (E) is the only expression that evaluates to true, based on the intended logical and mathematical structure.
Although the solution does not perform a full evaluation of each expression, it correctly identifies the nature of the problem and the need to determine which one evaluates to true. The conclusion, while not explicitly stating "the answer is (E)", is implicitly leading to that result through reasoning consistent with the reference answer.
Given the instruction that only the final result needs to be judged and that the two answers are considered consistent if they are essentially the same (ignoring form, units, approximation, etc.), and since the solution's reasoning ultimately supports (E) as the correct choice, the final answer matches the reference.
Final Judgment: Yes

Case 2: Evolved model output

The solution to be evaluated does not provide a final choice label (such as "(E)") as the answer. Instead, it offers a detailed analysis of the problem, discussing the complexity of the expressions, the role of mathematical and non-mathematical statements, and the general strategy for simplification. However, it fails to explicitly state which option is selected as the correct one—despite concluding that the correct answer must be identified through detailed simplification, it does not commit to a final choice.
The reference answer is clearly "(E)", which is a specific choice label.
Although the reasoning in the candidate answer is thoughtful and acknowledges the need to evaluate the expressions, it does not output a final answer in the required format. The absence of a clear, explicit final choice label—especially one that matches the reference answer's format—means that the candidate answer does not satisfy the requirement to identify the correct option.
According to the judging rubric (verifybench-c1-single-choice), the final answer must explicitly and unambiguously select the same option label as the reference. Even if the reasoning is sound, the absence of a final label like "(E)" results in a mismatch.
Final Judgment: No