RUBRICLEARNER: LEARNING EVALUATION RUBRICS THROUGH DIAGNOSIS-GUIDED REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

As language models take on open-ended work such as deep research and office assistance, response quality hinges on many purpose-dependent requirements rather than a single correct answer, so evaluation depends on rubrics that state what a response must satisfy and how shortcomings affect its assessment. A rubric is nevertheless an empirical artifact: drafted criteria can miss the distinctions a task actually requires, and piecemeal score-driven edits risk discarding still-valid guidance while overfitting to the errors they fix. We introduce RubricLearner, a framework that improves evaluation rubrics through a diagnosis-guided refinement loop. Each iteration contrasts erroneous scoring justifications with reference judgments and distills conditional rules specifying when a lesson applies, what to attend to, and which mistake to avoid. Accepted rules accumulate, and the complete rubric is regenerated from them; a regenerated rubric is retained only when it improves validation performance. Because each rule states when it applies and accepted rules are retained rather than overwritten, the accumulated guidance stays specific to its context and grows across iterations. With Qwen3.7-Max as the judge, RubricLearner outperforms a state-of-the-art baseline across eight public benchmarks, an average relative gain of 3.51%. Beyond these scores, RubricLearner produces higher-quality rubrics with fewer defective criteria, and the improvement extends to unseen examples.

1 INTRODUCTION

Large language models are increasingly deployed on open-ended tasks such as deep research (Du et al., 2026), office assistance (Zhou et al., 2026), and decision support, where a single response must satisfy multiple requirements, respect constraints, and balance competing objectives. Such tasks rarely admit a single correct answer: whether a response is good depends on the user's purpose — brevity may be valuable for a recommendation but insufficient when a decision requires an explanation of competing options. Evaluation therefore depends on explicit standards as well as the judge's capabilities (Liu et al., 2023; Kim et al., 2024). Rubrics make these standards concrete, specifying what an answer should satisfy and how shortcomings should affect its assessment (Gunjal et al., 2025; Viswanathan et al., 2025).

Explicit criteria still need an empirical basis. Even a carefully written rubric can encode distinctions that the task's actual judgments do not support (Qi et al., 2026). Feedback from reference judgments can expose missing requirements or misinterpreted priorities, but correcting one score is not the same as learning a useful evaluation principle. A local correction can be overgeneralized, and later edits may overwrite guidance that remains useful. For instance, correcting an omitted explanation should not become an unconditional preference for longer answers. The challenge is to learn lessons with clear applicability boundaries and accumulate them across revisions.

Existing methods already generate, diagnose, learn, and revise evaluation guidance. Dynamic Rubrics generates and refines criteria (Wang & Blanco, 2026), while Learnable Assessment Skills learns reusable rubric-construction guidance through error-driven revision and validation-based selection (Wang et al., 2026). Our focus is therefore not the existence of a feedback loop, but how it converts errors into reusable experience. Specifically, what should a diagnostic lesson preserve, and how should new lessons be incorporated without overwriting accepted guidance?

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.

We introduce RubricLearner, which improves evaluation rubrics through a diagnosis-guided refinement loop. By contrasting reference judgments with erroneous scoring justifications, the method distills feedback into rules that specify applicability conditions, positive evaluation criteria, and lessons about mistakes to avoid. It retains accepted experience, uses the accumulated rules to regenerate a complete rubric, and selects updates based on validation performance. Together, conditional applicability and append-only retention keep each lesson scoped to the situations it was learned from while letting useful guidance compound across iterations.

Across eight public benchmarks under Qwen3.7-Max, RubricLearner exceeds a state-of-the-art baseline by an average relative gain of 3.51%. Its rubrics are also of higher quality, showing an approximately 21% lower average defect rate (ADR) than the same baseline across two tasks. Its accuracy also holds up on unseen examples: on a shared HelpSteer3 held-out set, its development-to-test drop is 8.09 percentage points smaller than the comparator's. Our contributions are as follows.

1. **Diagnosis-guided rubric evolution.** We formulate a refinement loop that turns scoring errors into conditional evaluation rules, retains accepted experience, and selects complete regenerated rubrics through validation.
2. **State-of-the-art evaluation accuracy.** With Qwen3.7-Max as the judge, we report a 3.51% average relative score improvement over a state-of-the-art baseline across eight public benchmarks, computed as the equal-weight mean of task-wise relative gains in accuracy or Pearson correlation.
3. **Rubric quality and generalization.** We report an approximately 21% lower average defect rate than a state-of-the-art baseline across two tasks (13.15% for RubricLearner versus 16.65% for the baseline, each averaged over the two tasks), and an 8.09-percentage-point difference between the HelpSteer3 development-minus-test accuracy gaps for RubricLearner and the comparator, complemented by inspection of learning artifacts and diagnosis-data requirements.

2 Related Work

Learning evaluation criteria. Explicit criteria provide a basis for model judgments: G-Eval organizes evaluation around task-specific standards, while Rubrics as Rewards uses fine-grained criteria to compute rewards (Liu et al., 2023; Gunjal et al., 2025). The content of these standards still requires scrutiny; RIFT characterizes defects such as subjective, overly broad, and unsupported criteria (Qi et al., 2026). Methods for acquiring criteria draw on different sources of feedback. Dynamic Rubrics uses model feedback to generate and refine rubrics (Wang & Blanco, 2026); Auto-Rubric induces criteria from preference pairs and compresses them into a shared set (Xie et al., 2026); Feedback-to-Rubrics revises expert criteria through mismatches between predicted and reference comments (Yoshida et al., 2026). Another line learns guidance for constructing rubrics or performing evaluation: Learnable Assessment Skills revises reusable rubric-construction skills from scoring errors and selects candidates through validation, while Eval-Skill uses exploration to develop evaluation skills for reward modeling (Wang et al., 2026; Yue et al., 2026). These approaches establish rubric construction as a data-driven learning process, with different choices of feedback and learned representation. RubricLearner builds on this perspective, focusing on how experience acquired across revisions accumulates and informs subsequent criteria.

Accumulating and revising evaluation knowledge. Learning over multiple revisions involves not only proposing new criteria, but also reusing prior experience and deciding which changes to retain. AMARIS retrieves historical analyses and update records to guide rubric improvement in rubric-based reinforcement learning (Wu et al., 2026). SHARP combines structured rule edits, validation, and human-readable version histories in trading policies, while SkillCoach combines candidate validation with accepted and rejected revision records for evaluating agentic skill use (Chen et al., 2026; Zhu et al., 2026). Historical context and revision selection thus already play a role in iterative improvement. RubricLearner differs in how it organizes learned rules and the rubrics generated from them: where Learnable Assessment Skills permits additions, removals, and revisions to existing skill rules (Wang et al., 2026), RubricLearner retains accepted conditional rules in an append-only history and regenerates the complete rubric from that history. Each proposed addition is assessed through the regenerated candidate and enters the active rule history only when the can-

didate's validation score exceeds that of the current rubric. Version records connect the diagnosis, proposed rule, and selection evidence. This organization allows the scoring text to be reorganized while preserving the rules and evidence underlying successive revisions.

3 RUBRICLEARNER

RubricLearner learns evaluation guidance from scoring errors while keeping the judge fixed. Its design separates the generation rules that accumulate diagnostic experience from the rubric used for scoring. In each iteration (Figure 1), diagnosis turns disagreements with reference judgments into conditional rules; refinement combines them with accepted rules to regenerate a complete rubric; and validation determines whether to adopt the update. Algorithm 1 specifies the complete learning procedure.

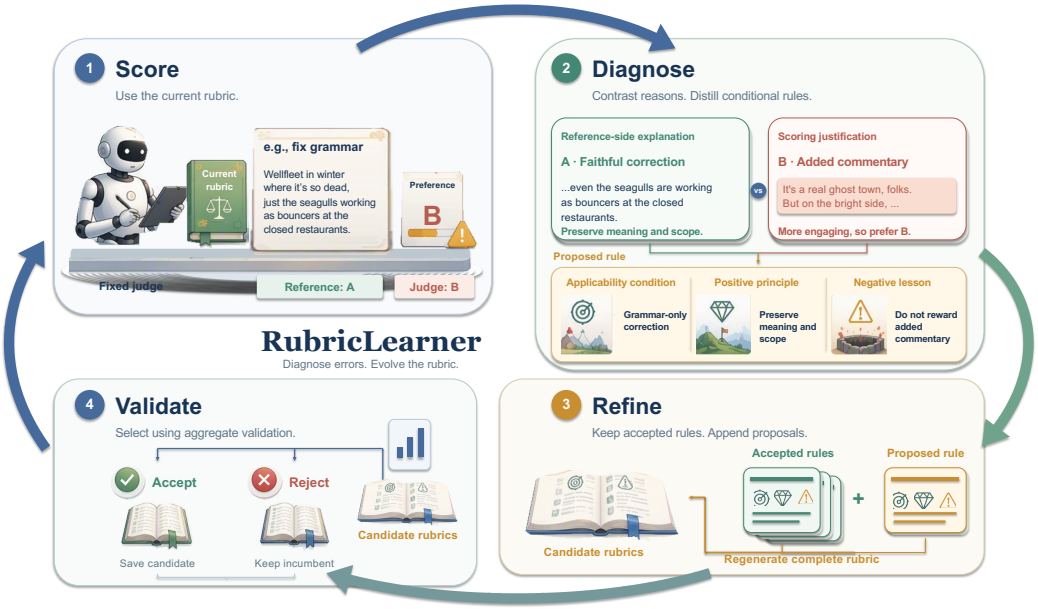


Figure 1: Overview of RubricLearner. A fixed judge scores responses; diagnosis contrasts scoring justifications with reference-side explanations it constructs from reference judgments to derive conditional generation rules. Refinement appends proposed rules to accepted history and regenerates a complete rubric; validation determines whether to retain the update. The grammar-correction example illustrates a rule for preserving the requested scope.

3.1 REPRESENTING EVALUATION KNOWLEDGE

Generation rules retain diagnostic experience; the rubric expresses it as criteria for the fixed judge. Its three fields are `key_elements` for independently checkable requirements, `score_levels` for behavioral anchors and score meanings, and `scoring_notes` for conditional guidance and edge cases. Rules guide all three fields; their condition–principle–lesson structure describes experience, not a one-to-one mapping onto these rubric fields.

Each learning epoch e separates labeled diagnosis data $\mathcal{D}^e_{\text{train}}$ from candidate-selection data $\mathcal{D}^e_{\text{val}}$. Initialization uses task instructions and representative training contents, without reference labels. The same five-step scaffold generates initial and refined rubrics (Appendix C).

3.2 LEARNING CONDITIONAL RULES FROM SCORING ERRORS

Diagnosis converts observed scoring errors into conditional lessons that can apply beyond the examples exposing them. Each round generates a working rubric from accepted rules and scores the current training batch. Diagnosis examines the judge's recorded scoring justifications alongside the

reference judgments. For each error, diagnosis constructs a reference-side explanation from the input, response contents, evaluation criteria, and reference label; no additional human-annotated rationale is required. Contrasting this explanation with the judge's justification helps identify omitted requirements, misplaced weights, or misunderstood criteria. Related failures are grouped and their task conditions examined to identify when a lesson should apply.

Each proposed generation rule has three parts: an *applicability condition* stating when the lesson applies, a *positive principle* specifying what to attend to, and a *negative lesson* identifying the erroneous reasoning to avoid.

Figure 1 illustrates grammar-only correction: preferring B for entertaining additions would reward an unrequested change. The rule calls for preserving meaning and scope and not rewarding added commentary, with grammar-only correction as its applicability condition. This limits the lesson to the relevant task, not creativity in general. In learning, negative lessons must be grounded in observed training errors; insufficient evidence permits no addition.

3.3 Refinement through Accumulated Experience

Refinement preserves accepted experience while allowing scoring criteria to be reorganized. New rules guide criterion construction, not sample answers or dataset-specific content; they are appended without modifying accepted rules, and the complete candidate history guides regeneration.

At revision round t of epoch e, let $\mathcal{H}_{e,t}$ be the ordered history of accepted rules and $\rho_{e,t}$ a proposed rule with the three parts described above. Tildes mark candidate objects that have not yet been accepted. Writing $[\rho_{e,t}]$ for a single-rule list and $\|$ for ordered append, the candidate history and complete candidate rubric are

$$\widetilde{\mathcal{H}}_{e,t} = \mathcal{H}_{e,t} \,\|\, [\rho_{e,t}], \qquad \widetilde{\mathcal{R}}_{e,t} = G_e\Big(\widetilde{\mathcal{H}}_{e,t}\Big).$$

The first expression leaves accepted history unchanged; the second uses G_e, the fixed scaffold with epoch-specific task and representative-example context, to regenerate all three rubric fields. The single-rule case extends to an ordered list, including an empty list when no addition is proposed. The candidate history is retained only on acceptance. Regeneration reorganizes the rubric text while keeping the accepted source rules intact.

3.4 Validating Proposed Refinements

Validation assesses proposed rules through the scoring performance of the complete rubric they generate, rather than the plausibility of the rules alone. Let $M_e(\mathcal{R}; \mathcal{D})$ measure agreement between the fixed judge's predictions under rubric $\mathcal{R}$ and the reference labels in labeled set $\mathcal{D}$; higher values indicate better agreement. We use accuracy for pairwise and listwise tasks, Cohen's κ for binary tasks, and quadratic weighted kappa (QWK), κ_w, for ordinal scale tasks. Only the aggregate score on $\mathcal{D}^e_{\text{val}}$ is used for selection; validation errors are not exposed for diagnosis.

The first candidate reaching validation after diagnosis and regeneration is accepted as the epoch's baseline. Thereafter, let $\mathcal{R}_{e,t}$ denote the best accepted rubric, or *incumbent*. A candidate is accepted only if

$$M_e(\widetilde{\mathcal{R}}_{e,t}; \mathcal{D}^e_{\text{val}}) > M_e(\mathcal{R}_{e,t}; \mathcal{D}^e_{\text{val}}). \tag{1}$$

The right-hand side is the incumbent's stored score, not a fresh reevaluation; the validation set, judge, and decoding configuration stay fixed within the epoch. Ties are rejected. Algorithm 1 specifies acceptance, same-batch retry, and stopping. Rules carry across epochs, but the validation baseline resets; cross-epoch improvement is not required. This is an instruction-driven protocol, not deterministic enforcement; implementation details and auxiliary interfaces appear in Appendix C.

For readability, the algorithm omits iteration subscripts from histories and candidates. $\mathcal{R}_{\text{work}}$ is regenerated for diagnosis, whereas $\mathcal{R}_{\text{best}}$ is the saved incumbent. Δ is the proposed rule list, m and m_{best} are candidate and stored validation scores, and f counts consecutive rejections.

Algorithm 1 RubricLearner

Require: Task instruction, labeled learning data $\mathcal{D}$, fixed judge, epoch seeds
Ensure: Best accepted rubric from the last epoch

$\mathcal{H} \leftarrow$ initial rule history (empty if unavailable)
for each epoch e **do**
  Split $\mathcal{D}$ into $\mathcal{D}^e_{\text{train}}$, $\mathcal{D}^e_{\text{val}}$; select the first training batch
  Set G_e's context from the task and training contents, excluding reference labels
  $\mathcal{R}_{\text{best}} \leftarrow \varnothing$; $m_{\text{best}} \leftarrow -\infty$; $f \leftarrow 0$
  while a training batch remains and $f < 3$ **do**
    $\mathcal{R}_{\text{work}} \leftarrow G_e(\mathcal{H})$; score the current batch with the fixed judge
    Contrast reference-side explanations with erroneous scoring justifications
    $\Delta \leftarrow$ conditional rules from diagnosis, ordered and possibly empty
    $\widetilde{\mathcal{H}} \leftarrow \mathcal{H} \parallel \Delta$; $\widetilde{\mathcal{R}} \leftarrow G_e(\widetilde{\mathcal{H}})$
    $m \leftarrow M_e(\widetilde{\mathcal{R}}; \mathcal{D}^e_{\text{val}})$ ▷ aggregate score only
    if $\mathcal{R}_{\text{best}} = \varnothing$ or $m > m_{\text{best}}$ **then**
      $\mathcal{H} \leftarrow \widetilde{\mathcal{H}}$; $\mathcal{R}_{\text{best}} \leftarrow \widetilde{\mathcal{R}}$; $m_{\text{best}} \leftarrow m$; $f \leftarrow 0$
      Advance to the next training batch
    else
      $f \leftarrow f + 1$; keep $\mathcal{H}$, $\mathcal{R}_{\text{best}}$, m_{best}, and the same batch
    end if
  end while
end for
return $\mathcal{R}_{\text{best}}$

4 Experimental Setup and Results

We evaluate whether diagnosis-guided rubric refinement improves evaluation quality on eight public benchmarks spanning four judgment formats, comparing against direct judging and prior rubric-learning methods under three fixed judges. We report aggregate performance first, then examine three complementary aspects of the learned rubrics: (1) criterion quality—does the resulting rubric express clear, appropriately scoped, and supported requirements? (2) development-to-held-out behavior—how do development records differ from performance on new examples? (3) diagnosis budget—how does the number of labeled diagnosis examples relate to evaluation accuracy?

4.1 Experimental Setup

Datasets and tasks. We use eight English-language benchmarks covering four judgment formats. HelpSteer3, PPE-IFEval, RewardBench, and JudgeBench require pairwise response comparisons (Wang et al., 2025; Frick et al., 2024; Lambert et al., 2024; Tan et al., 2025). RewardBench2 requires selecting the best of four responses (Malik et al., 2026); FollowBench requires binary judgments of instruction satisfaction (Jiang et al., 2024). BiGGen-Bench and Feedback-Bench require graded assessments (Kim et al., 2025; 2024). For each task, we split the available data into a development set, which each compared method may draw on during its own training or evolution, and a held-out set that no method sees during training; Table 1 evaluates every method on the full dataset, and Section 4.4 examines the held-out set. We report accuracy for pairwise, listwise, and binary tasks, and Pearson correlation r for scale tasks.

Methods and judges. Table 1 groups six methods under Qwen3-8B, Qwen3.7-Max, and DeepSeek-V4-Flash. Direct Judge scores responses without learned evaluation skills. Dynamic Rubrics generates instruction-specific criteria; we use its training-free variant (Wang & Blanco, 2026). Learnable Skills revises rubric-generation rules using training errors and validation feedback (Wang et al., 2026). Auto-Rubric induces and compresses criteria from preference pairs (Xie et al., 2026), while Eval-Skill learns evaluation skills through exploration (Yue et al., 2026). RubricLearner contrasts reference judgments with erroneous scoring justifications, appends supported conditional guidance to accepted generation rules, and regenerates complete rubrics.

Table 1: Evaluation scores by judge and judgment format. Avg. is the unweighted mean of four pairwise or two scale tasks, computed before rounding. Scores show three decimals; higher is better. Bold marks displayed column maxima within each judge, including ties.

	Pairwise Accuracy ↑					Listwise Accuracy ↑	Binary Accuracy ↑	Scale Pearson r ↑		
Method	**HelpSteer3**	**PPE-IFEval**	**Reward Bench**	**Judge Bench**	**Avg.**	**Reward Bench2**	**Follow Bench**	**BiGGen-Bench**	**Feedback-Bench**	**Avg.**
Qwen3-8B										
Direct Judge	0.677	0.691	0.824	0.712	0.726	0.570	0.638	0.366	0.589	0.478
Dynamic Rubrics	0.717	0.645	0.907	0.691	0.740	**0.818**	0.684	0.345	0.543	0.444
Learnable Skills	**0.725**	0.783	0.901	0.818	0.807	0.766	0.673	0.452	**0.764**	0.608
Auto-Rubric	0.724	0.777	**0.935**	0.757	0.798	0.809	**0.715**	0.463	0.735	0.599
Eval-Skill	0.704	0.794	0.870	0.797	0.791	0.705	0.712	0.496	0.729	0.613
RubricLearner	0.724	**0.799**	0.914	**0.823**	**0.815**	0.803	0.708	**0.524**	0.738	**0.631**
Qwen3.7-Max										
Direct Judge	0.747	0.794	0.911	0.825	0.819	0.745	0.742	0.464	0.696	0.580
Dynamic Rubrics	0.769	0.734	0.954	0.794	0.813	0.895	0.758	0.413	0.615	0.514
Learnable Skills	0.788	0.886	0.958	0.928	0.890	0.859	0.773	0.533	0.828	0.681
Auto-Rubric	0.780	0.867	0.952	0.896	0.874	**0.898**	0.793	0.562	0.790	0.676
Eval-Skill	0.786	0.885	0.956	0.928	0.889	0.880	0.805	0.580	**0.835**	0.708
RubricLearner	**0.791**	**0.903**	**0.960**	**0.940**	**0.899**	0.893	**0.810**	**0.614**	0.831	**0.723**
DeepSeek-V4-Flash										
Direct Judge	0.749	0.781	0.897	0.832	0.815	0.683	0.723	0.493	0.653	0.573
Dynamic Rubrics	0.785	0.734	0.971	0.816	0.827	0.880	0.735	0.428	0.605	0.517
Learnable Skills	0.787	0.863	0.965	0.929	0.886	0.832	0.753	0.555	0.822	0.689
Auto-Rubric	0.797	0.855	**0.975**	0.930	0.889	**0.896**	0.769	0.586	0.765	0.676
Eval-Skill	**0.799**	0.886	0.938	0.939	0.890	0.868	0.786	0.602	**0.827**	0.715
RubricLearner	0.793	**0.888**	0.964	**0.957**	**0.901**	0.867	**0.794**	**0.612**	0.822	**0.717**

Implementation details. RubricLearner keeps judge parameters fixed and separates diagnosis from validation within each epoch; candidates enter only on strict improvement over the epoch's validated baseline (Equation 1). Each run uses 100 labeled diagnosis examples per task under the same two-epoch pipeline (seeds 42 then 43, two diagnosis batches); the acceptance gate validates candidates on a separate 100-example selection set. Evaluation subsets and method-specific inference settings are in Appendix B.1.

4.2 Overall Evaluation Performance

Across the eight public benchmarks in Table 1, the Qwen3.7-Max comparison favors learning evaluation guidance over judging directly or generating criteria without learning. Learnable Skills, Auto-Rubric, Eval-Skill, and RubricLearner generally score above Direct Judge and training-free Dynamic Rubrics, without every learning method winning every comparison. Their common distinction is learning from examples or feedback, not generation alone: Eval-Skill learns evaluation skills, while Auto-Rubric induces criteria from preference pairs. The compared methods follow their respective evaluation protocols (Appendix B.1).

With Qwen3.7-Max as the judge, RubricLearner improves over Learnable Skills on all eight public benchmarks, achieving an average relative score improvement of 3.51%. We average each task's relative gain in accuracy or Pearson correlation equally. On BiGGen-Bench, Pearson correlation is 0.614 versus 0.533 for Learnable Skills; on FollowBench, accuracy is 0.810 versus 0.773. The former requires clear, actionable scoring criteria, while the latter requires interpreting instruction requirements under task-specific conditions. These improvements are consistent with contrasting reference judgments and erroneous scoring justifications to derive generation rules with explicit applicability conditions. Learnable Skills also uses diagnosis and iterative refinement, and the two methods differ in how accepted experience is organized and retained.

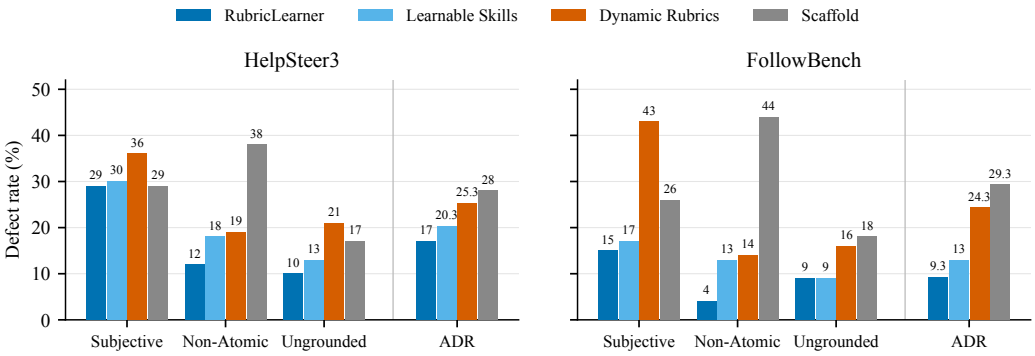


Figure 2: Textual defects in rubrics on HelpSteer3 (left) and FollowBench (right), measured using RIFT's Subjective, Non-Atomic, and Ungrounded categories (Qi et al., 2026). ADR is the arithmetic mean of the three defect rates; all values are percentages, and lower is better. Across the two tasks, RubricLearner has an approximately 21% lower average defect rate than Learnable Skills.

4.3 Examining the Learned Evaluation Criteria

We examine whether diagnosis-guided refinement improves criterion quality by comparing the reported defect rates of four rubric sources on HelpSteer3 and FollowBench: RubricLearner, Learnable Skills, training-free Dynamic Rubrics, and a scaffold ablation. In Figure 2, the *Scaffold* series is the one-shot rubric that the generator of Section 3 produces before any rules are learned, i.e., RubricLearner without the refinement and validation loop. We measure three RIFT defects: Subjective, Non-Atomic, and Ungrounded (Qi et al., 2026). Defect rates are percentages of criteria exhibiting each defect; Average Defect Rate (ADR) is their arithmetic mean, and lower is better.

In the reported results, RubricLearner's ADR averaged across the two tasks is approximately 21% lower than Learnable Skills' (Figure 2). The two non-learning references are higher still, with Dynamic Rubrics and the scaffold ablation about 47% and 54% above RubricLearner, so the gain reflects learned revision rather than merely having a rubric. Non-Atomic defects show the largest difference among the three categories, while differences in the other categories are smaller. This pattern suggests that the improvement lies chiefly in more focused criteria and clearer separation of independent requirements rather than wording alone, a direction consistent with diagnosis-guided revision.

4.4 Evaluation on Held-Out Examples

Does development-stage performance persist on unseen examples? Figure 3 reports development and held-out accuracy for RubricLearner, Learnable Skills, and Eval-Skill (Yue et al., 2026) on HelpSteer3, FollowBench, and PPE-IFEval; the controlled gap comparison focuses on RubricLearner and Eval-Skill on the shared HelpSteer3 held-out set under Qwen3.7-Max, the two systems whose development records and inference protocols are documented side by side (Appendix B.3). We define the gap as development score minus held-out accuracy. These two systems record development scores differently and use different inference protocols.

RubricLearner's gap is 8.09 percentage points smaller than Eval-Skill's: its held-out accuracy actually exceeds its development record by 2.29 percentage points, whereas Eval-Skill drops by 5.79 (Figure 3). On the shared 384 examples the two systems are statistically indistinguishable, with 316 versus 315 correct (McNemar $p = 1.0$). The higher development record for Eval-Skill therefore does not translate into a clear advantage on unseen examples. This contrast highlights why learning evaluation criteria should not pursue development scores alone: held-out evaluation is needed to assess whether the learned criteria remain useful beyond development.

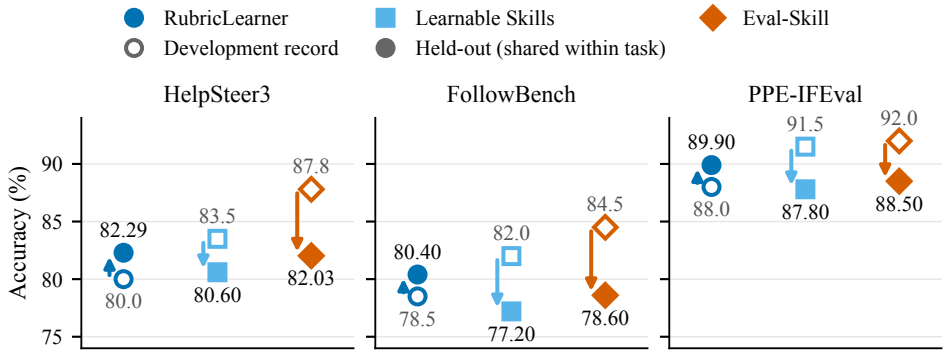


Figure 3: Development and held-out accuracy (%) on HelpSteer3, FollowBench, and PPE-IFEval. Marker shape and color identify the method; open and filled markers denote development and held-out values, respectively, and arrows show the transition. On HelpSteer3, RubricLearner maintains its development-level performance on unseen examples.

4.5 Diagnosis-Data Requirements

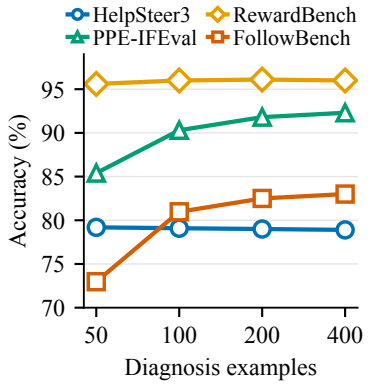
How does the diagnosis budget relate to evaluation accuracy? We vary the labeled diagnosis examples from 50 to 400, excluding validation labels, and measure evaluation accuracy on four tasks (Figure 4a).

The four tasks fall into two groups (Figure 4a). HelpSteer3 and RewardBench are essentially flat from 50 to 400 examples (about 79% and 96%), because their prompts share most criteria and a small diagnosis set already covers them. FollowBench and PPE-IFEval instead rise with diminishing returns, from 72.97% to 83.00% and from 85.40% to 92.30%, with most of each gain realized by 100 examples, the budget used in the main table. Panel 4b suggests why: two HelpSteer3 prompts as different as defining a word and writing a school report are judged by the same shared criteria, whereas each FollowBench prompt imposes a distinct structural constraint, such as three sections in a fixed order or exactly one code block with no extra text. New diagnosis examples therefore keep surfacing new prompt-specific criteria exactly where criteria are prompt-specific, and because revisions enter only on validated improvement, a larger budget can extend the rubric without merely fitting the development set.

5 Discussion and Limitations

Discussion. RubricLearner connects reference-versus-error diagnosis, append-only generation rules, complete-rubric regeneration, and validation selection in a diagnosis-guided learning process, with revision records supporting inspection; revision candidates enter only on strict improvement over the epoch's validated baseline. The main table and three analyses address different aspects of learned criteria: the audit scores only the criteria a rubric states and covers the three RIFT defect dimensions on HelpSteer3 and FollowBench, and the held-out comparison shows that differently defined development records can diverge from unseen-example performance, since RubricLearner's and Eval-Skill's development-minus-held-out gaps differ by 8.09 percentage points while their held-out endpoints stay close.

Limitations and future work. Several directions extend the present evidence beyond the studied tasks and protocols. Broader task and domain coverage would test how far diagnosis-guided rubric learning transfers, and deploying the lifecycle interfaces (versioning, rollback, and audit) in real evaluation workflows would measure their practical utility. A full accounting of annotation efficiency across the rubric lifecycle, from diagnosis labels to rubric maintenance, is another natural next step. Deployed tasks also evolve, as new policies and user needs gradually change what a good response must satisfy; extending the diagnosis-and-revision loop to track such requirement drift, with revision records keeping each update traceable, would carry the lifecycle view from offline learning to sustained operation. Portability across judges is a further open direction: the cross-judge



(a) Accuracy vs. diagnosis budget.

HelpSteer3 · criteria shared across prompts

Prompt: What does "ace" mean?
Criterion: Are the meanings accurate?

Prompt: Write Sky's school report from notes.
Criterion: Notes preserved, no invented details?

FollowBench · criteria unique to each prompt

Prompt: Explain film photography using an introduction, main argument, and conclusion.
Criterion: Three sections present, in order?

Prompt: Return only terminal output in one code block.
Criterion: Exactly one code block, with no extra text.

(b) Example prompts and their criteria.

Figure 4: (a) Evaluation accuracy (%) versus the number of diagnosis examples (50–400) on four tasks; higher is better: HelpSteer3 and RewardBench stay flat, while FollowBench and PPE-IFEval rise with diminishing returns. (b) For example prompts, criteria shared across prompts contrasted with criteria unique to each prompt.

results suggest that learned rubrics travel between judges, and testing them across broader judge families, including open-weight judges of differing strength, would clarify how much of a rubric's value is judge-specific.

6 Conclusion

RubricLearner learns evaluation rubrics through diagnosis-guided refinement while keeping the judge fixed. It contrasts reference judgments with erroneous scoring justifications, appends conditional guidance to accepted generation rules, and regenerates complete rubrics. The first validated candidate establishes each epoch's baseline; subsequent candidates require strict improvement in the recorded validation metric. Versioning, rollback, and audit are supporting interface specifications rather than additional learning stages. The main benchmark comparison and three complementary analyses characterize cross-task performance, criterion quality, development-to-held-out behavior, and diagnosis-data requirements. The results characterize the complete system across the reported tasks and protocols. Lifecycle-interface utility, total annotation efficiency, and broader generalization remain for direct future evaluation.

AI use statement

In this work, we used generative AI tools (large-language-model assistants) for language editing and polishing of the manuscript and for assisting with boilerplate analysis and plotting scripts. We did not use generative AI tools to produce experimental results, quantitative claims, or scientific conclusions: every number in the paper is computed by our evaluation pipelines and checked by the authors. The language models that act as judges and rubric generators are experimental subjects of this study, not writing assistants. We have reviewed all AI-assisted text and code and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

We see no major ethical concerns. All experiments use publicly released benchmarks and pretrained language models accessed through their public interfaces; no human subjects, personal, or private data are involved. RubricLearner improves the quality of automated evaluation; like other evaluation tools it could be misapplied outside its validated scope, and we discuss its scope and limitations in Section 5.

Reproducibility statement

The method is fully specified in Section 3 and Algorithm 1, and the evaluation configurations, baseline protocols, and gate metrics are given in Section 4 and Appendix B. Per-task values underlying every aggregate are listed in the appendix audit tables so that reported numbers can be recomputed and checked. We will release code and evaluation scripts upon acceptance.

References

Xiwen Chen, Wenhui Zhu, Songzhu Zheng, Kashif Rasul, Yueyue Deng, and Huayu Li. Sharp: A self-evolving human-auditable rubric policy for financial trading agents, 2026. URL https://arxiv.org/abs/2605.06822.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents, 2026. URL https://arxiv.org/abs/2506.11763.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N. Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for rlhf, 2024. URL https://arxiv.org/abs/2410.14872.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains, 2025. URL https://arxiv.org/abs/2507.17746.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models, 2024. URL https://arxiv.org/abs/2310.20410.

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, 2024. URL https://arxiv.org/abs/2310.08491.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models, 2025. URL https://arxiv.org/abs/2406.05761.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling, 2024. URL https://arxiv.org/abs/2403.13787.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. URL https://arxiv.org/abs/2303.16634.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation, 2026. URL https://arxiv.org/abs/2506.01937.

Zhengyang Qi, Charles Dickens, Derek Pham, Amanda Dsouza, Armin Parchami, Frederic Sala, and Paroma Varma. Rift: A rubric failure mode taxonomy and automated diagnostics, 2026. URL https://arxiv.org/abs/2604.01375.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges, 2025. URL https://arxiv.org/abs/2410.12784.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models, 2025. URL https://arxiv.org/abs/2507.18624.

Yun Wang, Xin Xia, Xuansheng Wu, Xiaoming Zhai, and Ninghao Liu. Learnable assessment skills for llm-based automated scoring: Rubric construction via iterative optimization, 2026. URL https://arxiv.org/abs/2605.29274.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages, 2025. URL https://arxiv.org/abs/2505.11475.

Zijie Wang and Eduardo Blanco. Generating and refining dynamic evaluation rubrics for llm-as-a-judge, 2026. URL https://arxiv.org/abs/2605.30568.

Peilin Wu, Xinlu Zhang, Kun Wan, Wentian Zhao, Gang Wu, Xinya Du, and Zhiyu Chen. Amaris: A memory-augmented rubric improvement system for rubric-based reinforcement learning, 2026. URL https://arxiv.org/abs/2605.18592.

Lipeng Xie, Sen Huang, Zhuo Zhang, Anni Zou, Yunpeng Zhai, Dingchao Ren, Kezun Zhang, Haoyuan Hu, Boyin Liu, Haoran Chen, Zhaoyang Liu, and Bolin Ding. Auto-rubric: Learning from implicit weights to explicit rubrics for reward modeling, 2026. URL https://arxiv.org/abs/2510.17314.

Kotaro Yoshida, So Kuroki, Yuki Imajuku, Taishi Nakamura, Ryunosuke Iwai, Haruki Goda, and Takuya Akiba. Feedback-to-rubrics: Can we learn expert criteria from inline comments?, 2026. URL https://arxiv.org/abs/2605.29857.

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.

Jingbo Zhou, Yusai Zhao, Qi Bao, Jingjia Cao, Zhenghai Chen, Chang Gao, Kaiqi Guo, Muxin Guo, Mingxuan Li, Xinjiang Lu, Yanru Ma, Yixiong Xiao, Zenghui Zhang, Le Zhang, and Hua Wu. Omegause-officeval: Benchmarking llm agents on long-horizon office-suite tasks with economic grounding, 2026. URL https://arxiv.org/abs/2607.27155.

Jiayin Zhu, Kelong Mao, Yudong Guo, Dengbo He, Sulong Xu, Simiu Gu, and Yutao Yue. Skillcoach: Self-evolving rubrics for evaluating and enhancing agentic skill-use, 2026. URL https://arxiv.org/abs/2607.01874.

A SYMBOL REFERENCE

Symbol	Meaning	Domain or unit
e, t	Learning epoch and within-epoch revision round	Non-negative integers
$\mathcal{R}_{e,t}$	Incumbent complete rubric after the epoch's validation baseline	Structured text object
$\widetilde{\mathcal{R}}_{e,t}$	Regenerated complete candidate rubric	Structured text object
$\mathcal{H}_{e,t}, \widetilde{\mathcal{H}}_{e,t}$	Accepted and candidate rule histories	Ordered rule sequences
$\rho_{e,t}$	Proposed rule: condition, principle, and lesson	Structured text rule
$\|$	Ordered append	Sequence operator
G_e	Complete-rubric generator for epoch e	Rule history $\rightarrow$ rubric
$\mathcal{D}^e_{\text{train}}, \mathcal{D}^e_{\text{val}}$	Diagnosis and candidate-selection data	Sets of examples
M_e	Internal candidate-selection metric	Accuracy, κ, or κ_w
$\mathcal{R}_{\text{work}}, \mathcal{R}_{\text{best}}$	Working rubric and saved incumbent in Algorithm 1	Structured text objects
Δ	Ordered list of proposed conditional rules	Possibly empty sequence
m, m_{best}	Candidate score and stored best score	Agreement scores
f	Consecutive rejection count	Non-negative integer
Acc	Accuracy against gold labels	$[0, 1]$
r	Pearson correlation for external scale evaluation	$[-1, 1]$
κ, κ_w	Cohen's kappa and quadratic weighted kappa	$[-1, 1]$
ADR	Mean rate over retained defect dimensions	$[0, 1]$
ρ_s	Spearman length–defect correlation	$[-1, 1]$
$N_{\max}$	Maximum labeled examples exposed for diagnosis	Positive integer
T	Judge sampling temperature	Non-negative real
b, c, p	McNemar discordant counts and p value	Counts; $p \in [0, 1]$
n	Number of evaluated examples or artifacts	Positive integer

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 BENCHMARK AND BASELINE CONFIGURATIONS

Evaluation configurations. The eight-task comparison uses Qwen3.7-Max for RubricLearner, Dynamic Rubrics, and Learnable Skills. The evaluated sample counts are 1,918 for HelpSteer3, 2,486 for PPE-IFEval, 500 for RewardBench, 350 for JudgeBench, 1,756 for RewardBench2, 925 for FollowBench, 2,776 for BiGGen-Bench, and 1,000 for Feedback-Bench. RewardBench is a sampled subset, labeled RewardBench500 in the result files. RewardBench2 uses four-candidate listwise selection; its standardized source contains 1,763 items, of which 1,756 have recorded evaluations here. These counts describe the evaluated configurations. The main-table evaluation uses each method's own learning and evaluation protocol rather than a single matched held-out protocol across methods. The separate 384-item HelpSteer3 evaluation in Appendix B.3 addresses new-example performance and is reported separately from the main-table column.

Baseline identity. Learnable Skills denotes the baseline in the historical run records, not Learnable Assessment Skills or Eval-Skill. Its implementation combines a fixed five-step rubric-generation procedure with editable generation rules. Training-batch errors guide additions, edits, and deletions to those rules; the resulting rubric is accepted only if its validation metric improves. The selected rubric supplies the judge's scoring criteria. Unlike RubricLearner's instruction-driven retention of accepted source rules with append-only additions, Learnable Skills permits revision of existing rules; both procedures regenerate complete rubrics and use validation feedback. For RubricLearner, the first candidate reaching validation after diagnosis and regeneration establishes the epoch's baseline; subsequent candidates must strictly improve the best recorded validation metric within that epoch. RewardBench accuracy is computed from 479 correct predictions among 500 pairs. Historical files use AutoEval for the system presented here as RubricLearner.

Published protocols. Auto-Rubric's Qwen3-8B scores come from its HelpSteer3-induced configuration (Xie et al., 2026, version 2, Table 10). The reported evaluation uses Voting@10 for RewardBench2 and Voting@5 for RewardBench and JudgeBench, with Qwen3-8B in thinking mode at $T = 0.7$. HelpSteer3 is the rubric-induction source, not an additional reported evaluation score.

Eval-Skill's scores and the Direct Judge Vanilla controls follow its original protocol (Yue et al., 2026, version 2, Table 2 and Appendix C.1). After excluding the Tie subset of RewardBench2, the source reserves 100 items in each of five domains for evolution, leaving $1,763 - 500 = 1,263$ test items. It reports the mean of five domain accuracies under average@3, not accuracy on our 1,756-item evaluation. For RewardBench, it starts with 2,984 pairs across four domains and reserves 100 pairs per domain, leaving $2,984 - 400 = 2,584$ test pairs and reporting four-domain macro accuracy under average@3. The Direct Judge entries use this same source protocol, not Auto-Rubric's zero-shot control.

Aggregation. Each Avg. equally weights four pairwise benchmarks or two scale benchmarks, using values before rounding to three decimal places. Accuracy percentages from publications are divided by 100 without otherwise changing precision. The group summaries are arithmetic means, not pooled accuracy or pooled correlation; there is no cross-metric total. Bold identifies displayed column maxima, including ties.

B.2 Criteria Audit and Instrument Checks

The recorded audit covers RubricLearner, Learnable Skills, Dynamic Rubrics, and the scaffold ablation, with 20 prompts per method and task. Each audit item contains the original prompt and the complete rubric. The judges decompose the rubric into atomic criteria and assess Subjective, Non-Atomic, and Ungrounded defects, with reasons and quoted evidence. The judges are Qwen3.7-Max under its recorded default configuration and Qwen3.8-Max with chain-of-thought disabled. Each performs two passes. For each method and task, each reported defect rate averages the proportion of decomposed criteria flagged for that category across valid audit passes within each prompt–rubric pair, then takes an equally weighted mean across prompts. Table 2 reports the retained dimension rates, their arithmetic mean (ADR), and the mean number of atomic criteria. Criterion count is not character length or a measure of requirement coverage.

Table 2: Detailed audit results for 20 prompts per method and task. Criteria is the mean atomic-criterion count; all defect columns are percentages. These values underlie Figure 2.

Method	Criteria	Subjective	Non-Atomic	Ungrounded	ADR
HelpSteer3					
RubricLearner	16.2	29	12	10	17.0
Learnable Skills	13.4	30	18	13	20.3
Dynamic Rubrics	5.5	36	19	21	25.3
Scaffold	39.6	29	38	17	28.0
FollowBench					
RubricLearner	18.1	15	4	9	9.3
Learnable Skills	15.2	17	13	9	13.0
Dynamic Rubrics	5.6	43	14	16	24.3
Scaffold	37.4	26	44	18	29.3

Checks on rubrics with injected defects reported recall of 100% for Subjective and Ungrounded defects and 75% for Non-Atomic defects. The missing-criterion category flagged the clean control and was excluded. Disconnected/rigid judgments had no corresponding injected validation variant and are not included in ADR or the main conclusions. For HelpSteer3/FollowBench, the reported inter-rater κ values are 0.92/0.91 for Subjective, 0.58/0.64 for Non-Atomic, and 0.57/0.86 for Ungrounded defects. The excluded missing-criterion category has $\kappa = 0.21/0.45$.

The analysis reports positive associations between rubric character count and its defect-density summary: $\rho_s = 0.694$ on HelpSteer3 and 0.712 on FollowBench, with 60 artifacts per task across RubricLearner, Dynamic Rubrics, and the scaffold ablation. These are pooled-across-method associations between rubric length and defect density.

B.3 Development Records and Shared Held-Out Evaluation

Table 3 distinguishes the development records from the common 384-example HelpSteer3 evaluation. RubricLearner's development score is obtained from validation predictions on the 100-item

candidate-selection set. Eval-Skill's development score is an evolution-stage cumulative trial accuracy, 404/460, not a statistic on an identically defined validation set. Those 460 trials are not 460 independent examples. The two development records therefore have different samples and statistical units.

Table 3: HelpSteer3 development records and held-out accuracies (%). The development-score column has method-specific meanings, shown explicitly. The held-out columns use the same 384 examples; 3.7/3.8 indicate judge versions.

Method	Development record	Score	Held-out (3.7)	Held-out (3.8)
RubricLearner	Validation (100 examples)	80.00	82.29 (316/384)	81.25
Eval-Skill	Evolution (460 trials)	87.83	82.03 (315/384)	80.99

Under Qwen3.7-Max, RubricLearner uses a single pass at $T = 0$; Eval-Skill uses three passes at $T = 0.7$, aggregating by majority vote per example. The paired counts are $b = 25$ for RubricLearner-only correctness and $c = 24$ for Eval-Skill-only correctness, yielding McNemar $p = 1.0$. We define each gap as development score minus held-out accuracy. RubricLearner's gap is $100(80/100 - 316/384) \approx -2.29$ percentage points; Eval-Skill's is $100(404/460 - 315/384) \approx +5.79$ percentage points. Calculated from the unrounded counts, Eval-Skill's gap minus RubricLearner's gap is 8.09 percentage points. With Qwen3.8-Max and chain-of-thought disabled, held-out accuracies are 81.25% and 80.99%, differing by 0.26 percentage points.

B.4 Diagnosis Budgets and Comparability

Table 4 reports the four budget settings on the four tasks.

Table 4: Diagnosis-budget records, in accuracy percentages. The 100-example entries are the main-table reference runs.

Diagnosis cap	HelpSteer3	PPE-IFEval	RewardBench	FollowBench
50	79.19	85.40	95.60	72.97
100	79.09	90.30	96.00	80.97
200	79.00	91.80	96.10	82.50
400	78.90	92.30	96.00	83.00

The design uses the same pipeline, two epochs, seed sequence 42 then 43, and two diagnosis batches, with the batch size varying with the budget. HelpSteer3 at the 50-example budget has 1518/1917 correct predictions, whereas the full main-table anchor has $n = 1918$; FollowBench's 50-example result is 675/925. The 50-example HelpSteer3 run also generated ten score levels, compared with two in the reference run.

$N_{\max}$ counts the maximum labeled examples available for diagnosis, not the validation labels used for candidate selection or an end-to-end annotation budget. The budget runs use a single seed trajectory.

C Implementation Details and Supporting Interfaces

Algorithm 1 in the main text specifies the learning loop; this appendix gives its generation and diagnosis procedures, configuration, and auxiliary interfaces.

Initialization and epoch scheduling. Initialization uses the task instruction and the contents of 10–15 representative training examples, excluding their reference labels from the generation context. The fixed five-step scaffold analyzes the task, identifies key elements, defines score levels, writes behavioral descriptors, and anticipates variation. Each round first generates a working rubric from accepted rules for training-batch diagnosis, then regenerates a candidate rubric after proposing additions. The saved incumbent is updated only on acceptance, not by working-rubric generation. The reported protocol uses two epochs with seeds 42 and 43 and a rejection limit of three. As shown

in Algorithm 1, rules carry forward between epochs, while the data split, batch position, rejection count, and best-metric tracker reset.

Diagnostic procedure. Diagnosis follows EXPLORE–CLUSTER–EXPLAIN–REVISE. EXPLORE summarizes confusion patterns, over- or under-scoring, and shared example structures; CLUSTER groups related failures; and EXPLAIN relates these groups to deficiencies in the rubric and its generation rules. Before proposing a revision, REVISE contrasts the explanation supporting each reference judgment with the judge's recorded erroneous scoring justification. This comparison examines omitted requirements, misplaced weights, misunderstood criteria, and misjudgments that depend on task conditions. Diagnosis uses labeled training examples, not validation errors or test examples; it does not replace reference judgments with unlabeled behavioral signals. Proposals guide rubric construction rather than encode sample answers, dataset names, or dataset-specific content, and may be omitted when too few errors or insufficient common structure support a reusable conditional rule.

Metric interpretation and scope. The internal selection metrics, Cohen's κ and QWK, differ from the external reporting metrics: accuracy for categorical tasks and Pearson correlation r for scale tasks. The first validated candidate establishes each epoch's baseline, and strict improvement is required for subsequent candidates within that epoch. Fixed decoding settings keep the judge's configuration constant across candidates. Rule preservation, generation, and state transitions are instruction-driven requirements of the protocol.

Supporting lifecycle interfaces. Versioning, rollback, and audit support inspection rather than add a learning stage. The interface contract distinguishes the incumbent from proposed candidates and associates each version with its parent, appended rules, diagnosis rationale, validation score, and acceptance decision, including rejected proposals. Rollback is specified as selecting a previously accepted complete rubric and recording the source, target, and reason without deleting later history; the audit view exposes these relations and decisions.