CriticGen: Generation-Aware Evaluation as Actionable Feedback

Huifang Du[*1], **Zecheng Zuo**[*2], **Sen Wang**[2], **Chenghao Fan**[3], **Haofen Wang**[4], **and Yehui Yang**[†1]
[1]AI Lab, Qifu Technology, Beijing, China
[2]Beijing University of Posts and Telecommunications, Beijing, China
[3]University of Science and Technology Beijing, Beijing, China
[4]Tongji University, Shanghai, China

Abstract

Current evaluation methods for large language models are coarse-grained and decoupled from generation, producing generic explanations that fail to provide actionable feedback for model improvement. We propose CriticGen, a fine-grained, generation-aware evaluation framework that turns evaluation into actionable control for answer improvement. CriticGen first generates sample-specific evaluation dimensions and scoring criteria, then uses the induced rubric to jointly produce a score, a reason, an executable refinement suggestion, and a refined answer. Experiments show that fine-grained evaluation should be both instance-specific and actionable. CriticGen improves human-rated rubric relevance/coverage from 3.33/4.03 to 3.97/4.24. On the score-consensus-filtered REFINEDATA test set, the best CriticGen variant reaches 0.9556/0.9560 Pearson/Spearman agreement with the reference scores and improves reason/suggestion F1 to 0.7554/0.7900. Independent validation provides additional support: on 180 human-scored instances, CriticGen-Qwen2.5-7B obtains higher observed score correlations than GPT-5, with 0.8868/0.8969 Pearson/Spearman versus 0.8542/0.8615. Crucially, its feedback translates into substantial answer improvement, raising accuracy from 45.36% to 91.26% on 5,000 examples with reliable gold answers, with 47.30% of answers corrected from Wrong→Right and only 1.40% degraded from Right→Wrong. The code is available at https://github.com/mug2mag/chain_critic.

1 Introduction

Large language models (LLMs) are increasingly expected to produce not only correct final answers, but also faithful, consistent, and constraint-aware reasoning processes (Wei et al., 2022; Cobbe et al., 2021). For chain-of-thought (CoT) outputs (Wei et al., 2022), evaluation is therefore no longer only a leaderboard metric. It also serves as a diagnostic tool for identifying reasoning failures, curating training data, and guiding model improvement (Ouyang et al., 2022). However, current evaluation methods still leave a gap between *judging* an answer and *improving* it.

The first limitation is that many evaluation protocols are insufficiently adaptive to individual answers. Outcome-level metrics such as exact match or shallow text overlap only indicate whether the final answer matches a reference, but do not reveal which reasoning aspect is weak (Papineni et al., 2002). Recent LLM-as-a-judge and fine-grained evaluation methods improve this by comparing responses, assigning ratings, or scoring outputs along multiple skills or criteria (Zheng et al., 2023; Liu et al., 2023; Ye et al., 2024; Kim et al., 2024). Generative judges further enhance interpretability by producing natural-language critiques together with judgments (Li et al., 2024b). Nevertheless, their rubrics are often fixed at the task or scenario level. Such criteria may cover general quality dimensions, but they may fail to capture the specific omissions, contradictions, or constraint violations that appear in a particular question–answer pair.

The second limitation is that evaluation signals are often not directly executable for generation. Evaluator models such as Prometheus and LLaVA-Critic can provide fine-grained scores or feedback, and these signals can be used for reward-guided selection or later policy optimization (Kim et al., 2024; Xiong et al., 2025; Schulman et al., 2017). This paradigm is effective, but it typically separates evaluation from generation: the evaluator is trained first, and its signal affects generation only through an additional optimization or selection stage. Recent critic-based generation methods show that critique can improve model outputs, for example by refining erroneous reasoning steps or by training

[*]Equal contribution.
[†]Corresponding author: yangyehuisw@126.com. Also contactable via duhuifang123@gmail.com.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 3975–3990
November 6-10, 2026 ©2026 Association for Computational Linguistics

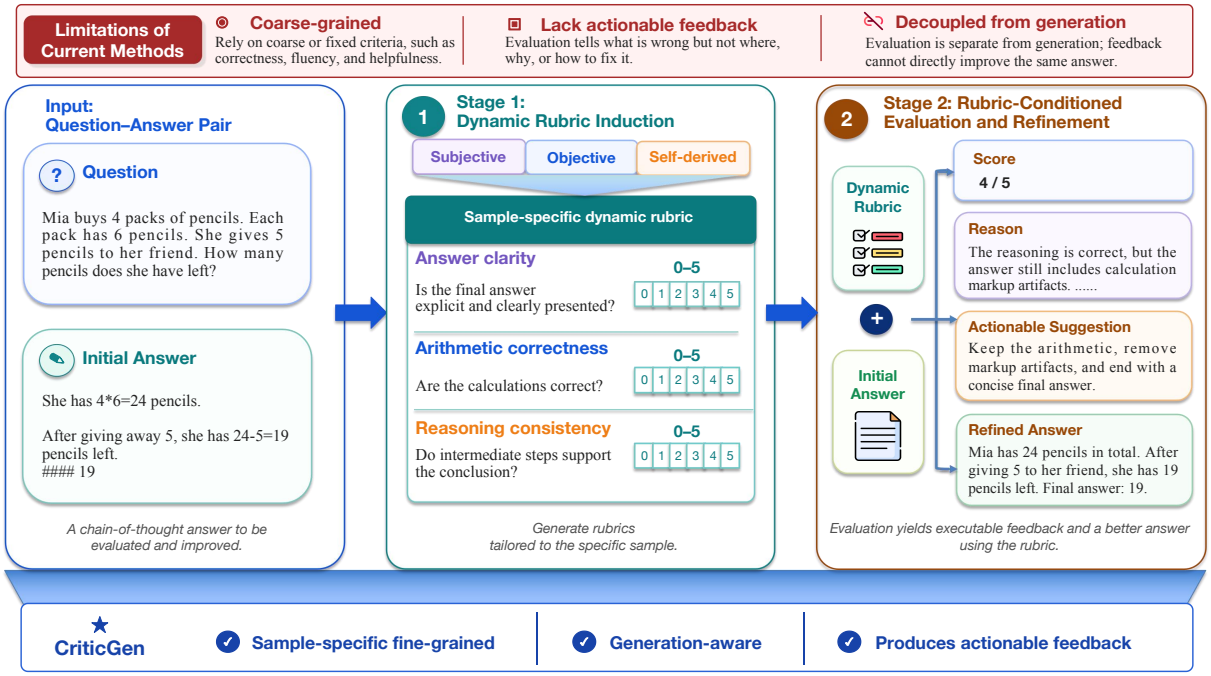


Figure 1: Illustration of CriticGen's core idea. Existing methods often provide fixed or generic feedback that is weakly connected to generation. CriticGen induces a sample-specific rubric and uses it to score, diagnose, plan, and refine the answer, turning evaluation into actionable control for answer improvement.

stronger policy behavior from critic data (Zheng et al., 2025; Wang et al., 2025). Yet the critique is still not necessarily grounded in the same fine-grained criteria used for scoring, nor is it explicitly converted into a concrete edit plan for revising the same answer.

We ask whether evaluation can be made both *sample-specific* and *actionable*. To this end, we propose **CriticGen**, a generation-aware evaluation framework that converts fine-grained evaluation into controllable answer refinement. CriticGen contains two separately trained SFT components coupled by a rubric interface. First, a rubric generator induces a sample-specific rubric $R(q, a) = \{(d_i, C_i)\}_{i=1}^{m}$, where each dimension d_i is paired with a graded scoring criterion C_i. The rubric is organized under high-level constraint families, including subjective quality, objective correctness, and self-derived reasoning constraints, while remaining specific to the current question and answer. Second, a rubric-conditioned evaluation-refinement model takes each (q, a, d_i, C_i) as input and generates a structured trajectory (s_i, r_i, u_i, a_i'): a score, a criterion-grounded reason, an executable refinement suggestion, and a refined answer. In this trajectory, the reason explains *why* the answer receives the score, while the suggestion specifies *what* should be changed before rewriting. Thus, CriticGen uses the same rubric to connect what to evaluate, why the answer is flawed, how it should be revised, and how the revised answer should be generated.

We construct two supervision datasets to support this framework. RUBRICDATA trains the rubric generator to produce answer-specific dimensions and graded criteria. REFINEDATA trains the evaluation-refinement model to produce the complete score–reason–suggestion–rewrite trajectory under each rubric item. Unlike reward-modeling or preference-optimization pipelines, CriticGen does not rely on a separate reinforcement-learning stage. Instead, it studies whether structured supervised trajectories can directly teach models to transform evaluation into an executable refinement signal.

Experiments validate the central hypothesis of CriticGen: fine-grained evaluation should be both instance-specific and actionable. Human evaluation shows that CriticGen induces higher-quality rubrics than a strong nine-dimension static checklist, improving relevance/coverage from 3.33/4.03 to 3.97/4.24 despite using a comparable number of dimensions (8.4 on average). On the score-consensus-filtered REFINEDATA test set, CriticGen-Qwen3.5-9B reaches 0.9556/0.9560 Pearson/Spearman agreement with the reference scores and raises reason/suggestion

F1 from the best baseline values of 0.6369/0.5994 to 0.7554/0.7900. Beyond these teacher-derived references, CriticGen-Qwen2.5-7B obtains higher observed Pearson/Spearman correlations against independent human scores, reaching 0.8868/0.8969 compared with 0.8542/0.8615 for GPT-5. On 5,000 examples with reliable gold answers, CriticGen refinement increases accuracy from 45.36% to 91.26%, with corrections (47.30%) far exceeding regressions (1.40%). Out-of-domain results on Feedback Bench further show transfer of rubric-conditioned evaluation ability.

We summarize our contributions as follows:

- We propose sample-specific rubric-conditioned evaluation and refinement, which induces fine-grained rubrics for each question–answer pair and uses them to guide a score–reason–suggestion–rewrite trajectory, turning evaluation from descriptive judgment into actionable control for answer improvement.
- We construct RubricData and RefineData to train the two SFT components of CriticGen, enabling rubric induction and rubric-conditioned refinement without a separate reward-optimization stage.
- We validate CriticGen with rubric-quality judgments, reference-based evaluation, independent human scoring, gold-answer correctness, out-of-domain transfer, and supervision ablations, providing complementary evidence for both evaluation consistency and actual answer improvement.

2 Data Construction

CriticGen requires supervision for two related but distinct capabilities: generating instance-specific rubrics and refining answers under a given rubric. We therefore construct the training data in two layers, both starting from a diverse pool of candidate answers.

Diverse answer pool. We collect questions from five reasoning-oriented datasets: GSM8K (Cobbe et al., 2021), LIMO (Ye et al., 2025), NaturalReasoning (Yuan et al., 2025), NuminaMath-CoT (Li et al., 2024a), and QwQ-LongCoT-130K (Amphora, 2024). Together, these datasets cover arithmetic reasoning, symbolic/logical reasoning, open-domain natural-language reasoning, competition-level math, and long-form CoT generation, which helps the rubric and refinement models avoid overfitting to a single reasoning style. For each question, we prompt generator models of different scales and capabilities, ranging from weaker open-weight models to stronger teacher models, to produce multiple candidate answers. This multi-model generation is deliberate: using models with different error profiles leads to a more balanced distribution of data across the full 0–5 quality range.

Rubric supervision generation. For each question–answer pair (q, a), GPT-5 (OpenAI, 2025b) generates a sample-specific rubric $R^* = \{(d_i^*, C_i^*)\}_{i=1}^m$, where each item contains an evaluation dimension and its 0–5 scoring criterion. The dimensions are organized under the three high-level families used in our method: subjective quality, objective correctness, and self-derived constraints. We store each annotated instance as (q, a, R^*), which provides supervision for rubric induction.

Evaluation-refinement supervision generation. Given each rubric item (d_i^*, C_i^*), GPT-5 further acts as a judge and refiner. It annotates a score s_i^*, a criterion-grounded reason r_i^*, an executable refinement suggestion u_i^*, and a refined answer $a_i'^*$. Each resulting instance is stored as $(q, a, d_i^*, C_i^*, s_i^*, r_i^*, u_i^*, a_i'^*)$, forming the supervision for rubric-conditioned evaluation-to-refinement learning. The score and reason capture how the answer satisfies the criterion, while the suggestion and refined answer instantiate how the evaluation is converted into a concrete improvement.

Score coverage completion. To reduce score imbalance, we check whether the answers for each question cover the full 0–5 quality range. If some score regions are missing, we prompt GPT-5 to synthesize additional answers targeted at the missing quality levels. For these synthesized answers, we repeat rubric supervision generation and evaluation-to-refinement supervision generation. This step improves score coverage and increases answer diversity beyond what is obtained from the initial set of generator models. Appendix B provides the task-level prompt templates for rubric induction, score-coverage completion, evaluation–refinement annotation, and refinement verification.

Quality filtering. We first apply a consensus filter to acquire reliable supervision in the evaluation-to-refinement stage. Given the same question,

Table 1: Statistics of the constructed datasets after quality filtering.

Dataset	Train	Test	Total
RUBRICDATA	43,145	8,629	51,774
REFINEDATA	1,180,929	23,619	1,204,548

answer, evaluation dimension, and scoring criterion, we compare the scores produced by GPT-5, Claude Sonnet 4.5 (Anthropic, 2025), and Gemini 3.1 Pro Preview (Google, 2026; Google AI for Developers, 2026). We keep a sample only when the three models agree on the score, and discard samples with score disagreement. We further apply lexical-diversity analysis and near-duplicate filtering to avoid highly templated or repetitively phrased supervision. Specifically, we compute field-level n-gram statistics and sample-level 4-gram Jaccard overlap over the generated fields, and remove near-duplicate samples. Appendix C provides the filtering details. The resulting data consist of two datasets: RUBRICDATA for dynamic rubric induction and REFINEDATA for rubric-conditioned evaluation-to-refinement learning. We split RUBRICDATA and REFINEDATA into training and test sets with ratios of 5:1 and 50:1, respectively. Table 1 summarizes the final data statistics.

3 Method

CriticGen is designed around the two limitations identified in the introduction: evaluation should be fine-grained for each instance, and its output should directly guide generation. Accordingly, we separate the framework into two functions: *rubric induction*, which specifies what should be evaluated, and *rubric-conditioned refinement*, which uses the evaluation result to improve the same answer.

3.1 Problem Formulation

Let q denote a question and a a candidate reasoning answer. Instead of learning only a scalar evaluator $f(q, a) \rightarrow s$, CriticGen aims to produce a structured set of rubric-conditioned evaluation and refinement outputs:

$$F(q, a) \rightarrow \{(d_i, C_i, s_i, r_i, u_i, a_i')\}_{i=1}^{m}. \quad (1)$$

Here, d_i is an evaluation dimension, C_i is its graded scoring criterion, s_i is the predicted score, r_i is the score explanation, u_i is an executable refinement suggestion, and a_i' is the refined answer. This formulation makes evaluation not only descriptive, through (s_i, r_i), but also operational, through (u_i, a_i').

3.2 Sample-Specific Rubric Induction

The first component induces a dynamic rubric for each question–answer pair. Given (q, a), a rubric generator $\mathcal{G}$ produces a set of dimensions and criteria:

$$R(q, a) = \{(d_i, C_i)\}_{i=1}^{m}. \quad (2)$$

To ensure broad coverage while avoiding a fixed rubric, we organize dimensions under three high-level families:

- **Subjective quality**: clarity, coherence, fluency, and readability;
- **Objective correctness**: factual accuracy, numerical correctness, and task-constraint satisfaction;
- **Self-derived constraints**: consistency among reasoning steps, absence of contradictions, and whether the conclusion follows from previous steps.

For each generated dimension d_i, the model also produces a 0–5 criterion C_i that describes what different score levels mean for the current question and answer. This differs from task-level or scenario-level rubrics: the generated criteria are tied to the concrete failure modes that may appear in the current answer.

3.3 Rubric-Conditioned Evaluation and Refinement

The second component takes a single rubric item (d_i, C_i) and evaluates the candidate answer under that criterion. For each (q, a, d_i, C_i), a rubric-conditioned model $\mathcal{S}$ generates four outputs:

$$\mathcal{S}(q, a, d_i, C_i) \rightarrow (s_i, r_i, u_i, a_i'). \quad (3)$$

The score $s_i \in \{0, \ldots, 5\}$ measures how well the answer satisfies the criterion. The reason r_i explains the score by identifying the relevant strength or weakness. The refinement suggestion u_i turns this explanation into an executable instruction, such as adding a missing reasoning step, correcting a calculation, enforcing a neglected constraint, or clarifying a justification. Finally, the refined answer a_i' is generated under the same criterion.

The key design choice is that u_i is not merely another explanation: r_i diagnoses the problem, while

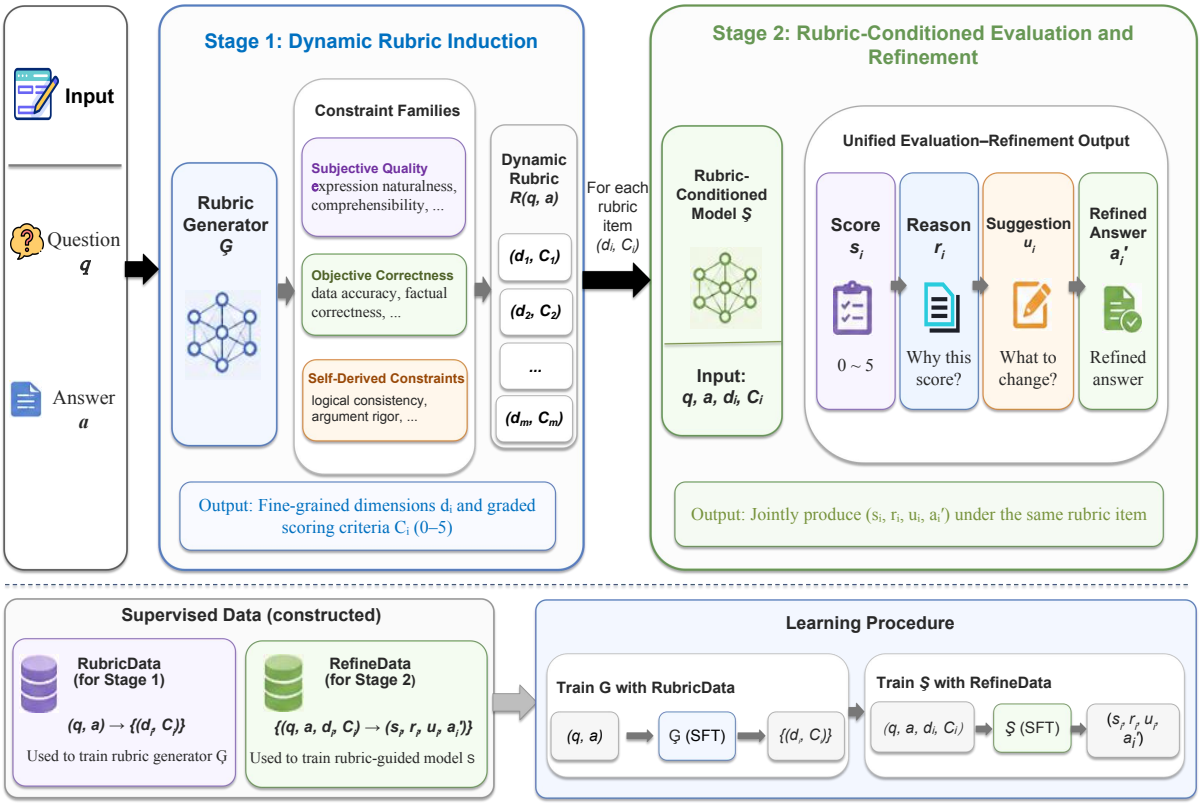


Figure 2: Overview of the CriticGen framework. CriticGen first induces a sample-specific dynamic rubric for each question–answer pair, and then uses each rubric item to condition a unified model that jointly produces a score, a reason, an executable refinement suggestion, and a refined answer.

u_i specifies how the refiner should act. We call u_i *executable* because it gives a concrete edit operation and target content, e.g., "insert a step before the final conclusion to apply the non-negativity constraint." Thus, the model learns to move from "why this score" to "what should be changed" and then to an improved answer.

3.4 Training Objectives

We train the two components with supervised fine-tuning on their corresponding constructed supervision. This design makes the source of the gains explicit: CriticGen is not optimized by a separate reward or preference stage, but learns from two forms of structured supervision—instance-specific rubrics and rubric-conditioned evaluation-to-refinement trajectories.

SFT for rubric induction. Let $\mathcal{D}_R = \{(q^{(n)}, a^{(n)}, R^{*(n)})\}_{n=1}^{N_R}$ denote RUBRICDATA, where $R^* = \{(d_i^*, C_i^*)\}_{i=1}^{m}$ is the constructed ground-truth rubric for a question–answer pair. The rubric generator $\mathcal{G}_\theta$ parameterizes a conditional distribution over variable-length rubric items and is optimized with the empirical negative log-likelihood:

$$L_{\text{rubric}}(\theta) = -\mathbb{E}_{\mathcal{D}_R} \log p_{\mathcal{G}_\theta}(R^* \mid q, a). \quad (4)$$

In practice, this objective is implemented as a token-level cross-entropy over the serialized dimensions and their 0–5 scoring criteria.

SFT for rubric-conditioned evaluation and refinement. Given REFINEDATA, each training example contains an input $x_i = (q, a, d_i, C_i)$ and a consensus-filtered teacher output $y_i^* = (s_i^*, r_i^*, u_i^*, a_i'^*)$, where the score s_i^* is retained only when multiple teacher models agree. We train the rubric-conditioned model $\mathcal{S}_\phi$ with a sequence likelihood objective over the serialized teacher output:

$$L_{\text{refine}}(\phi) = -\mathbb{E}_{\text{REFINEDATA}} \log p_{\mathcal{S}_\phi}(y_i^* \mid x_i). \quad (5)$$

This single objective teaches the model to produce the complete score–reason–revision-suggestion–rewrite trajectory under the same rubric item. The fixed field order also provides lightweight structural supervision, so the model learns not only the content of each field but also the interface through which evaluation is converted into an executable refinement signal.

Although the two modules are trained independently, they are coupled at inference time through the induced rubric, which serves as the interface between evaluation target generation and actionable refinement.

4 Experiments

We structure our experiments around the two components of CriticGen: sample-specific rubric induction and rubric-conditioned evaluation and refinement. For rubric induction, we examine whether the induced rubric $R(q, a) = \{(d_i, C_i)\}_{i=1}^{m}$ serves as a useful assessment plan for a specific question–answer pair, using human judgments of rubric quality. For rubric-conditioned evaluation and refinement, we test whether conditioning on a rubric item (d_i, C_i) supports reliable scoring, criterion-grounded reasoning, actionable revision suggestions, and improved answer refinement.

4.1 Setup

Held-out data. For rubric induction quality, we sample 240 (q, a) pairs, 48 from each of GSM8K, LIMO, NaturalReasoning, NuminaMath-CoT, and QwQ-LongCoT. These pairs are from the rubric-induction test set of RUBRICDATA. For **Ours**, we generate answer-specific rubrics $R(q, a) = \{(d_i, C_i)\}$ (mean 8.4 dimensions per pair; Table 2). **Static** uses the same 240 (q, a) pairs with nine fixed, instance-agnostic dimensions (three per family: subjective quality, objective correctness, self-derived constraints), keeping checklist length comparable to **Ours**. This setup tests whether answer-specific rubrics improve relevance and coverage over static ones with matched dimension counts.

Human evaluation protocol. For each of the 240 held-out (q, a) pairs, raters compare two rubrics under the same question and answer. Rubric identities are blinded as Rubric A/B. Two independent raters with prior experience in LLM-as-a-judge evaluation and rubric design assign five-point Likert scores (Likert, 1932) for two attributes: *relevance*, measuring how well the rubric dimensions and criteria fit the current (q, a), and *coverage*, measuring whether salient failure modes for judging the answer are represented. If the two raters differ by at least one point on either rubric for either dimension, a senior adjudicator re-scores both rubrics; otherwise we average the two ratings. Raters are not authors of this work.

Independent validation. To validate second-stage scoring independently of synthetic references, we sample 180 rubric-conditioned instances balanced across the six 0–5 score levels. Two independent annotators assign scores blindly, with substantial disagreements re-evaluated by a senior annotator. We compare model predictions against the resulting human scores using Pearson and Spearman correlations; Appendix E provides the complete model comparison.

For objective refinement validation, we sample 5,000 unique question–answer pairs from the REFINEDATA test split, restricted to GSM8K, NuminaMath, and the applicable subset of NaturalReasoning with reliable gold answers. The sample covers all six reference-score levels. We extract the last explicit final-answer span, normalize its surface form, and manually resolve parser failures or ambiguous extractions before computing exact-answer accuracy; Appendix F provides further details.

Baseline models. We select baselines to cover backbone, scale, family, and proprietary-model comparisons. Qwen2.5-7B-Instruct, Qwen3.5-4B, and Qwen3.5-9B are the backbones of our CriticGen variants, enabling direct task-specific fine-tuning. Qwen2.5-14B-Instruct and Qwen2.5-32B-Instruct test whether gains come from scale alone, while Mistral-7B-Instruct-v0.3 and Ministral-3-14B-Instruct-2512 provide non-Qwen open-weight comparisons. GPT-5 serves as a high-capacity proprietary reference evaluator. Details and citations are provided in Appendix G.

Evaluation and Refinement Metric Protocol. For rubric-conditioned evaluation and refinement, we assess the outputs (s_i, r_i, u_i, a_i') under each rubric item (d_i, C_i) along four strands:

Scores s_i: we report Pearson and Spearman correlations with the reference scores in REFINEDATA.

Reasons r_i and **executable suggestions** u_i: since both reasons and suggestions are free-form texts, we evaluate them by reference-based semantic-unit alignment rather than exact matching (Zhang et al., 2020; Sellam et al., 2020; Li et al., 2025). For a generated text x_i and its ground-truth reference x_i^*, we split them into semantic units $X_i = \{x_{i,j}\}_{j=1}^{m_i}$ and $G_i = \{g_{i,k}\}_{k=1}^{n_i}$, and compute the pairwise similarity matrix

$$A_{k,j}^{(i)} = \mathrm{Sim}(g_{i,k}, x_{i,j}). \tag{6}$$

We then define recall-style coverage, precision-

style consistency, and their harmonic mean as:

$$R_i = \frac{1}{n_i} \sum_{k=1}^{n_i} \max_j A_{k,j}^{(i)},$$
$$P_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \max_k A_{k,j}^{(i)}, \quad (7)$$

with $F_i = 2P_iR_i/(P_i + R_i)$. Here, R_i measures how well the generated text covers the ground-truth semantic units, while P_i measures whether the generated units are supported by the ground-truth reference. We report macro-averaged Precision, Recall, and F1 over all examples. This protocol is applied to reasons by setting $(x_i, x_i^*) = (r_i, r_i^*)$, and to executable suggestions by setting $(x_i, x_i^*) = (u_i, u_i^*)$.

Refined answers a_i': a GPT-5 verifier scores both the original answer a and the refined answer a_i' under the same rubric item (d_i, C_i). We report the improvement rate, unchanged rate, degradation rate, and the non-degradation rate, where non-degradation means that the refined answer is rated no worse than the original answer.

4.2 Rubric Induction Quality

Rubric quality. Table 2 reports human judgments of rubric quality on 240 held-out (q, a) pairs. The adjudication rate is 17.1%, and the pre-adjudication Cohen's κ (Cohen, 1968) is 0.61 for relevance and 0.57 for coverage. Under the same (q, a), raters assign higher scores to *Ours* than to *Static* on both relevance and coverage. The improvement is larger on relevance, increasing from 3.33 to 3.97, while coverage increases from 4.03 to 4.24. This result suggests that induced rubrics mainly improve instance-level targeting rather than simply covering more generic evaluation aspects. Qualitatively, Static dimensions behave as templates aligned with broad failure families, so raters still see many salient checks covered and assign mid-to-high coverage. Answer-conditioned induction adds criteria that explicitly mark omissions the fixed nine dimensions never single out, which is why raters score *coverage* higher for *Ours*.

Appendix D further supports that the induced rubrics are answer-sensitive, not only higher-quality under human judgment.

4.3 Rubric-Conditioned Evaluation Capability

Tables 3, 4, and 5 evaluate rubric-conditioned evaluation in terms of score prediction, reason generation, and executable suggestion generation. Overall, CriticGen variants obtain higher reference-based metrics than their corresponding backbones and the prompt-only instruction-following baselines. Because these baselines receive the same rubric-conditioned inputs but are not trained on CriticGen's structured supervision, this comparison measures the effect of learning the evaluation-to-refinement trajectory rather than isolating rubric induction alone.

For score prediction, Table 3 shows that CriticGen achieves much stronger agreement with the constructed reference scores. Within this reference-based setting, CriticGen-Qwen3.5-9B obtains the highest Pearson/Spearman correlations of 0.9556/0.9560, exceeding the strongest open-weight baseline Qwen3.5-9B (0.8342/0.8328) by 0.1214/0.1232 and the prompted GPT-5 baseline (0.9212/0.9143) by 0.0344/0.0417. This comparison measures consistency with the score-consensus-filtered references under our evaluation protocol and should not be interpreted as universal evaluator superiority over GPT-5. The gains are also evident for smaller models: CriticGen-Qwen2.5-7B improves Qwen2.5-7B-Instruct from 0.6355/0.6300 to 0.9518/0.9521, suggesting that the benefit comes from rubric-conditioned evaluation rather than

Table 2: Rubric induction: human judgments of rubric quality on 240 held-out (q, a) pairs. Both rubrics are evaluated under the same question and answer; only the rubric text differs. Scores are post-adjudication means on a five-point Likert scale.

Rubric source	Rel.	Cov.	#Dim.
Static	3.33	4.03	9.0
Ours	**3.97**	**4.24**	8.4

Table 3: Rubric-conditioned score evaluation quality. P. and S. denote Pearson and Spearman correlations with reference scores in REFINEDATA.

Model	P.	S.
Qwen2.5-7B-Instruct	0.6355	0.6300
Qwen2.5-14B-Instruct	0.8144	0.8133
Qwen2.5-32B-Instruct	0.8099	0.8210
Qwen3.5-4B	0.7568	0.7504
Qwen3.5-9B	0.8342	0.8328
Mistral-7B-Instruct-v0.3	0.6293	0.6236
Ministral-3-14B-Instruct-2512	0.8052	0.8108
GPT-5	0.9212	0.9143
CriticGen-Qwen2.5-7B	0.9518	0.9521
CriticGen-Qwen3.5-4B	0.9407	0.9413
CriticGen-Qwen3.5-9B	**0.9556**	**0.9560**

Table 4: Rubric-conditioned reason generation quality. P., R., and F1 denote semantic-unit precision, recall, and F1 against reference reasons.

Model	P.	R.	F1
Qwen2.5-7B-Instruct	0.6674	0.5190	0.5796
Qwen2.5-14B-Instruct	0.6894	0.5098	0.5820
Qwen2.5-32B-Instruct	0.6919	0.5106	0.5836
Qwen3.5-4B	0.6636	0.5932	0.6218
Qwen3.5-9B	0.6768	0.6092	0.6369
Mistral-7B-Instruct-v0.3	0.6638	0.5053	0.5701
Ministral-3-14B-Instruct-2512	0.6472	0.5865	0.6075
CriticGen-Qwen2.5-7B	0.7150	0.7188	0.7150
CriticGen-Qwen3.5-4B	0.7478	0.7560	0.7500
CriticGen-Qwen3.5-9B	**0.7549**	**0.7594**	**0.7554**

Table 5: Rubric-conditioned executable suggestion generation quality. P., R., and F1 denote semantic-unit precision, recall, and F1 against reference suggestions.

Model	P.	R.	F1
Qwen2.5-7B-Instruct	0.5988	0.4119	0.4844
Qwen2.5-14B-Instruct	0.6540	0.4459	0.5260
Qwen2.5-32B-Instruct	0.6921	0.4669	0.5533
Qwen3.5-4B	0.6451	0.4740	0.5414
Qwen3.5-9B	0.6976	0.5342	0.5994
Mistral-7B-Instruct-v0.3	0.6696	0.4553	0.5373
Ministral-3-14B-Instruct-2512	0.6372	0.5122	0.5612
CriticGen-Qwen2.5-7B	0.7393	0.7357	0.7340
CriticGen-Qwen3.5-4B	0.7783	0.7780	0.7754
CriticGen-Qwen3.5-9B	**0.7930**	**0.7921**	**0.7900**

model scale alone.

Independent human validation. To complement the consensus-filtered reference evaluation, we compare model predictions against independent human scores on 180 rubric-conditioned instances. CriticGen-Qwen2.5-7B achieves Pearson/Spearman correlations of 0.8868/0.8969, compared with 0.8542/0.8615 for GPT-5 and 0.8205/0.8244 for Qwen3.5-9B. The complete comparison is reported in Appendix E. These results provide an independent human anchor showing that CriticGen's strong scoring performance is not limited to agreement with the teacher-derived references.

Tables 4 and 5 further show that CriticGen improves the generative side of evaluation. For reasons, CriticGen-Qwen3.5-9B increases the best baseline F1 from 0.6369 to 0.7554, with recall rising from 0.6092 to 0.7594. For executable suggestions, the gain is larger, improving F1 from 0.5994 to 0.7900 and recall from 0.5342 to 0.7921. Similar improvements appear on smaller backbones. These results support our claim that evaluation should go beyond scalar scoring: by jointly modeling scores, reasons, and executable suggestions under dynamic rubrics, CriticGen turns evaluation into actionable control for answer refinement.

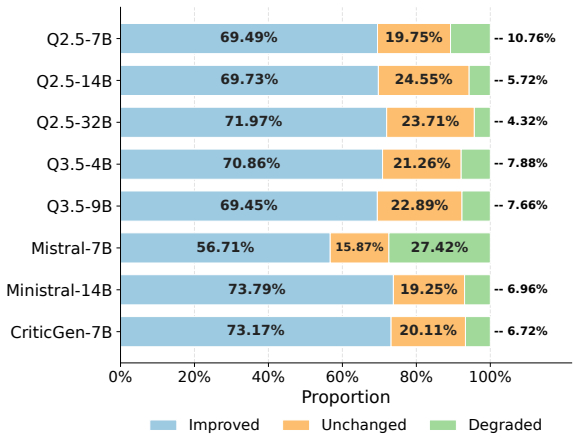


Figure 3: Rubric-conditioned refinement outcomes. Each bar shows the proportions of improved, unchanged, and degraded cases after refinement under the same rubric item.

4.4 Refinement Capability

Figure 3 compares each rubric-conditioned refined answer a_i' with the original answer a under the same rubric item, and categorizes the outcome as improved, unchanged, or degraded. The non-degradation rate is the sum of improved and unchanged cases.

Overall, most models improve many answers, but their reliability varies. Mistral-7B-Instruct-v0.3 performs worst, with 56.71% improved cases and 27.42% degraded cases. Stronger baselines are more stable: Qwen2.5-32B-Instruct achieves the highest non-degradation rate of 95.68%, while Ministral-3-14B-Instruct-2512 obtains the highest baseline improvement rate of 73.79%. CriticGen achieves a strong balance, improving 73.17% of cases, leaving 20.11% unchanged, and degrading only 6.72%, for a non-degradation rate of 93.28%. Compared with its Qwen2.5-7B backbone, CriticGen improves both the improvement rate, from 69.49% to 73.17%, and the non-degradation rate, from 89.24% to 93.28%.

These results show that CriticGen converts fine-grained evaluation into actionable refinement signals, producing revisions that improve answers while keeping degradation risk low.

Gold-answer validation. We further evaluate refinement with an objective final-answer metric on

Table 6: Ablation study on the second-stage evaluation-to-refinement model. All variants use the same rubric condition (d_i, C_i). Score P. denotes Pearson correlation for score prediction; reason and suggestion quality are measured by semantic-unit F1. Improved denotes the proportion of refined answers rated better than the original answers.

Variant	Sc. P.	R. F1	Sugg. F1	Improved
Full CriticGen	**0.9407**	**0.7500**	**0.7754**	**73.54**
w/o reason	0.9323	–	0.7663	60.48
w/o suggestion	0.9304	0.7484	–	61.57
rewrite only	–	–	–	62.35

5,000 unique question–answer pairs from the REFINEDATA test split with reliable gold answers. The original answers achieve 45.36% accuracy, while CriticGen-Qwen2.5-7B reaches 91.26%. The Wrong→Right rate is 47.30%, compared with only 1.40% Right→Wrong. GPT-5 achieves the highest overall accuracy at 96.84%, while CriticGen obtains the best result among the evaluated open-weight models and improves its identical Qwen2.5-7B backbone from 85.48% to 91.26%. The complete comparison is reported in Appendix F. This objective evaluation provides direct evidence that the refinement gains include improved final-answer correctness, rather than only higher scores from the GPT-5 verifier.

4.5 Ablation Study on Evaluation-to-Refinement

We ablate the structured score–reason–suggestion–rewrite trajectory to study its role in rubric-conditioned refinement. Since rubric induction is evaluated separately in Section 4.2, all variants use the same rubric condition (d_i, C_i) and differ only in the second-stage supervision fields. All experiments are based on CriticGen-Qwen3.5-4B-SFT and evaluated on the held-out split of REFINEDATA.

Table 6 shows that Full CriticGen performs best, achieving an improved rate of 73.54. Removing reason or suggestion supervision only mildly affects the remaining evaluation metrics, but reduces refinement success to 60.48 and 61.57, respectively. The rewrite-only variant also underperforms Full CriticGen (62.35) and loses interpretable score, reason, and suggestion outputs. These results indicate that criterion-grounded diagnosis and executable edit planning are crucial for converting evaluation into effective and controllable refinement.

5 Conclusion

We presented CriticGen, a generation-aware evaluation framework that turns fine-grained evaluation into actionable control for answer refinement. CriticGen first induces sample-specific rubrics for each question-answer pair and then uses the induced criteria to jointly produce a score, a criterion-grounded reason, an executable refinement suggestion, and a refined answer. Experiments show that CriticGen improves rubric quality, score correlation, reason and suggestion generation, and answer refinement reliability. These results suggest that evaluation can serve not only as a diagnostic signal but also as a direct mechanism for controllable generation improvement.

Limitations

This work has several limitations. First, CriticGen is trained with supervised fine-tuning, which depends on the quality and coverage of the constructed supervision data. Although our results show strong performance, the framework may still inherit biases or blind spots from the generated rubrics and refinement trajectories. Second, our current experiments focus on text reasoning tasks, and further validation is needed on broader domains such as multimodal reasoning, long-form generation, and interactive decision-making. Third, while CriticGen directly converts evaluation into refinement through a structured SFT trajectory, it does not yet optimize the model with reinforcement learning. Future work can further use reinforcement learning or preference optimization to strengthen the alignment between rubric-conditioned feedback and actual answer improvement.

References

Amphora. 2024. QwQ-LongCoT-130K. Hugging Face dataset.

Anthropic. 2025. Introducing Claude Sonnet 4.5. Anthropic News. Accessed 2026-05-26.

Yinzhu Chen, Abdine Maiga, Hossein A. Rahmani, and Emine Yilmaz. 2026. Retrieval-augmented agentic rubric generation for reliable medical response evaluation. *arXiv preprint arXiv:2601.15161*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman.

2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220.

Google. 2026. Gemini 3.1 Pro: A smarter model for your most complex tasks. Google Blog. Accessed 2026-05-26.

Google AI for Developers. 2026. Gemini API models. Documentation. Accessed 2026-05-26.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lelio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothee Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024a. NuminaMath. Hugging Face dataset.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2024b. Generative judge for evaluating alignment. In *International Conference on Learning Representations*.

Shusheng Li, Jiale Li, Yifei Qu, Xinwei Shi, Yanliang Guo, Ziyi He, Yubo Wang, and Wenjun Tan. 2025. Semantic-Eval: A semantic comprehension evaluation framework for large language models generation without training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9675–9690.

Rensis Likert. 1932. *A Technique for the Measurement of Attitudes*. Number 140 in Archives of Psychology. The Science Press, New York.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Mistral AI. 2024. Mistral-7B-Instruct-v0.3. Hugging Face model card. Accessed 2026-05-26.

Mistral AI. 2025. Ministral 3 14B. Official model card. Accessed 2026-05-26.

OpenAI. 2025a. GPT-5 system card. OpenAI. Accessed 2026-05-26.

OpenAI. 2025b. Introducing GPT-5. OpenAI. Accessed 2026-05-26.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Qwen Team. 2026a. Qwen3.5-4B. Hugging Face model card. Accessed 2026-05-26.

Qwen Team. 2026b. Qwen3.5-9B. Hugging Face model card. Accessed 2026-05-26.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

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

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.

Xiyao Wang, Chunyuan Li, Jianwei Yang, Kai Zhang, Bo Liu, Tianyi Xiong, and Furong Huang. 2025. LLaVA-Critic-R1: Your critic model is secretly a strong policy model. *arXiv preprint arXiv:2509.00676*.

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. 2024. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Tianyi Xiong, Xiyao Wang, Dong Guo, Qinghao Ye, Haoqi Fan, Quanquan Gu, Heng Huang, and Chunyuan Li. 2025. LLaVA-Critic: Learning to evaluate multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*. Version 2 revised January 2025.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2024. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *International Conference on Learning Representations*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. LIMO: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Dong Wang, Ilia Kulikov, Kyunghyun Cho, Yuandong Tian, Jason E. Weston, and Xian Li. 2025. NaturalReasoning: Reasoning in the wild with 2.8m challenging questions. *arXiv preprint arXiv:2502.13124*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

Xin Zheng, Jie Lou, Boxi Cao, Xueru Wen, Yuqiu Ji, Hongyu Lin, Yaojie Lu, Xianpei Han, Debing Zhang, and Le Sun. 2025. Critic-CoT: Boosting the reasoning abilities of large language model via chain-of-thought critic. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1731–1751.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Related Work

Reasoning Evaluation Beyond Final Answers. Chain-of-thought prompting improves mathematical and multi-step reasoning by eliciting intermediate steps (Wei et al., 2022; Cobbe et al., 2021), but also complicates evaluation: correct answers may still contain incomplete, unclear, or constraint-inconsistent reasoning. Our work evaluates and revises reasoning under explicit criteria, rather than merely encouraging longer chains.

LLM Judges and Fine-Grained Criteria. LLM-as-a-judge methods scale automatic evaluation by prompting or training models to compare responses and assign ratings (Zheng et al., 2023; Liu et al., 2023; Wang et al., 2024). Fine-grained evaluators decompose quality into skills or criteria (Ye et al., 2024; Kim et al., 2024), and generative judges further produce critiques or feedback (Li et al., 2024b; Wang et al., 2023). However, their criteria are often task- or scenario-level, and their outputs usually remain scores, critiques, or rewards. CriticGen instead induces a sample-specific rubric for each question–answer pair and conditions both evaluation and refinement on it.

Recent work by Chen et al. (Chen et al., 2026) also constructs instance-specific rubrics, using a retrieval-augmented multi-agent framework to ground medical-response criteria in authoritative evidence and user-interaction constraints. Their framework emphasizes knowledge-grounded rubric construction for reliable clinical evaluation and uses the resulting criteria to guide response refinement. CriticGen is complementary but differs in both conditioning and learning setup: it induces rubrics directly from each question–answer pair and trains reusable rubric-induction and rubric-conditioned evaluator–refiner capabilities, with the score, reason, executable suggestion, and rewrite produced under the same rubric item rather than

assembled by an inference-time retrieval and agent pipeline.

Evaluation as Reward for Model Improvement. Evaluators are also used as reward models or selection functions for improving separate policy models. Preference modeling and RLHF-style methods learn reward signals from human or model feedback (Ziegler et al., 2019; Ouyang et al., 2022; Schulman et al., 2017), while Prometheus and LLaVA-Critic provide fine-grained feedback for reward-guided selection or reinforcement learning (Kim et al., 2024; Xiong et al., 2025). In these methods, evaluation and generation are typically separated. CriticGen studies a more direct path: the model scores the current answer, diagnoses weaknesses, generates an executable refinement suggestion, and rewrites the answer under the same rubric.

Self-Critique and Critic-Based Generation. Self-refinement and Reflexion show that model-generated feedback can improve later outputs through critique and revision (Madaan et al., 2023; Shinn et al., 2023). Critic-CoT trains step-wise critiques for reasoning errors (Zheng et al., 2025), and LLaVA-Critic-R1 shows that critic training can yield strong policy behavior after reinforcement learning on critic data (Wang et al., 2025). These works validate critic signals, while CriticGen focuses on converting dynamically induced, fine-grained rubrics into directly executable refinement signals by grounding the score, reason, suggestion, and rewrite in the same instance-specific criteria.

B Prompt Templates and Output Formats

For reproducibility, we provide the task-level prompt templates used to instantiate the data-construction and evaluation pipeline. Square-bracketed fields are replaced with the corresponding instance content. Provider-specific message wrappers are omitted because they do not alter the task instructions or required output schema.

Rubric induction.

```
You are an expert evaluator of
reasoning answers. Given a question and
a candidate answer, construct a
sample-specific evaluation rubric. The
rubric must contain only dimensions
that are useful for evaluating this
particular answer. Cover relevant
aspects from three families: subjective
quality, objective correctness, and
self-derived reasoning constraints.
Avoid generic, redundant, or
overlapping dimensions. For every
dimension, provide a 0–5 criterion
whose six score anchors are specific,
mutually distinguishable, and grounded
in the supplied question and answer. A
score of 0 denotes complete failure on
that dimension and 5 denotes full
satisfaction. Do not score or rewrite
the answer.
Question: [QUESTION]
Candidate answer: [ANSWER]
Return only a JSON array. Each item
must contain “family”, “dimension”, and
“criterion”, where “criterion” maps the
keys “0” through “5” to their score
descriptions.
```

Evaluation–refinement annotation.

```
You are an expert judge and editor.
Evaluate the candidate answer only
under the supplied dimension and
scoring criterion. First assign an
integer score from 0 to 5. Then explain
the score using concrete evidence from
the answer. Next provide one executable
refinement suggestion that states both
what should change and where or how to
change it. Finally, rewrite the answer
so that it better satisfies the
criterion while preserving correct and
relevant content. The refined answer
must be self-contained and must not
mention the evaluation process.
Question: [QUESTION]
Candidate answer: [ANSWER]
Dimension: [DIMENSION]
Scoring criterion: [CRITERION]
Return only one JSON object with the
keys “score”, “reason”, “suggestion”,
and “refined_answer”, in that order.
```

Score-coverage completion.

```
Generate one plausible candidate answer
to the question at the requested
overall quality level on a 0–5 scale.
Match the target level through the
correctness and completeness of the
reasoning, not by stating or referring
to the target score. Use realistic
mistakes for low or intermediate levels
and avoid artificial corruption.
Preserve a natural answer style and
vary the error pattern relative to the
existing answers. Level 0 is irrelevant
or wholly incorrect; level 1 has severe
errors; level 2 shows limited progress
with major errors; level 3 is partially
correct but has an important gap; level
4 is mostly correct with a minor issue;
and level 5 is fully correct, complete,
and well justified.
Question: [QUESTION]
Target quality level: [TARGET SCORE]
Existing answers to avoid duplicating:
[EXISTING ANSWERS]
Return only the candidate answer.
```

Rubric-grounded refinement verification.

```
Compare the original and refined
answers under exactly the same
evaluation dimension and scoring
criterion. Consider both the reasoning
process and the final answer. Label the
refined answer "improved" only if it
better satisfies the criterion without
introducing a new material error; label
it "degraded" if it satisfies the
criterion less well or introduces a new
material error; otherwise label it
"unchanged". Judge content rather than
wording or length.
Question: [QUESTION]
Dimension: [DIMENSION]
Scoring criterion: [CRITERION]
Original answer: [ORIGINAL ANSWER]
Refined answer: [REFINED ANSWER]
Return only one JSON object with the
keys "label" and "rationale".
```

C Data Filtering

To ensure that the constructed data are not dominated by templated or repetitive generations, we conduct lexical-diversity analysis and near-duplicate filtering during data construction. This process is applied after score-consensus filtering and before the final train/test split.

Text normalization. We normalize all textual fields before computing lexical statistics. Specifically, we lowercase English text, remove redundant whitespace, and tokenize each field into word-level tokens. For each instance, we consider both the input-side fields and the generated fields in the constructed data. The input-side fields include the question and the original answer. The generated fields include the induced rubric, the evaluation reason, the executable refinement suggestion, and the refined answer. We analyze these fields separately because they play different roles in the constructed evaluation-to-refinement trajectory.

Field-level lexical diversity. We first measure field-level lexical diversity using Distinct-n, defined as the ratio of unique n-grams to the total number of n-grams in a field collection:

$$\text{Distinct-}n = \frac{|\text{UniqueNgrams}_n(\mathcal{T})|}{|\text{AllNgrams}_n(\mathcal{T})|}, \quad (8)$$

where $\mathcal{T}$ denotes all texts from a specific field. Higher Distinct-n values indicate greater lexical diversity and lower template reuse. We compute Distinct-2, Distinct-3, and Distinct-4 for questions, answers, rubrics, reasons, executable suggestions, and refined answers. Table 7 reports the field-level diversity statistics.

Field	Distinct-2	Distinct-3	Distinct-4
Questions	0.132	0.232	0.410
Answers	0.182	0.431	0.547
Rubrics	0.258	0.362	0.569
Reasons	0.243	0.432	0.651
Suggestions	0.213	0.396	0.678
Refined answers	0.199	0.480	0.693

Table 7: Field-level lexical diversity of the constructed data. The generated evaluation-to-refinement fields, especially reasons, executable suggestions, and refined answers, show substantial high-order n-gram diversity. Questions exhibit lower diversity because they are more task-driven and often follow similar query forms. Overall, the results suggest that the constructed data are not dominated by repeated lexical templates.

Sample-level near-duplicate filtering. Field-level diversity measures the overall lexical variation of the dataset, but does not directly identify near-duplicate instances. We therefore apply a sample-level overlap filter. For each candidate instance, we concatenate its generated fields, including the rubric, reason, executable suggestion, and refined answer, into a single text sequence. We then compute the 4-gram Jaccard overlap between the candidate and previously retained instances:

$$J_4(x_i, x_j) = \frac{|\mathcal{G}_4(x_i) \cap \mathcal{G}_4(x_j)|}{|\mathcal{G}_4(x_i) \cup \mathcal{G}_4(x_j)|}, \quad (9)$$

where $\mathcal{G}_4(x)$ denotes the set of 4-grams in instance x. A candidate instance is removed if its maximum 4-gram Jaccard overlap with any retained instance exceeds $\tau = 0.75$. In implementation, we use MinHash-based locality-sensitive hashing to retrieve high-overlap candidates efficiently, followed by exact Jaccard computation for final filtering.

Overall, Table 7 shows that the constructed data retain meaningful lexical variation across both input-side and generated fields. Although the questions are relatively more standardized due to the task-oriented data construction process, the generated reasons, executable suggestions, and refined answers exhibit high-order diversity, indicating that the evaluation-to-refinement trajectories are not merely repeated templates.

D Answer Sensitivity of Induced Rubrics

Data. We use 180 held-out questions with 3–5 candidate answers each. The questions are held out from training and are used only for the answer-sensitivity analysis.

Protocol. To measure whether rubrics are answer-sensitive, we compare rubrics generated for different answers to the same question. For **Ours**, we compare answer-specific rubrics $R(q, a)$ and $R(q, a')$. For **Q-only**, we use a shared question-only rubric $R(q)$ for both answers, which cannot vary with the answer by construction. For **Q-only×2**, we compare two independently generated question-only rubrics for the same question, providing a lexical-noise control for repeated rubric generation. Rubric change is measured by the lexical Jaccard shift over pooled `dimension_name` tokens.

Results. We evaluate whether induced rubrics change when the answer changes. As a lightweight, reproducible proxy, we measure *lexical* turnover in `dimension_name` strings only and exclude the 0–5 criterion text, which reduces sensitivity to long paraphrases in scoring anchors. For a rubric $R(q, a)$, we take every `dimension_name`, lowercase it, tokenize it into alphanumeric word *types*, and pool those types into a single set W_a (duplicate tokens across dimensions count once). Fixing the same question q, we induce $R(q, a)$ and $R(q, a')$ from distinct answers a and a', construct W_a and $W_{a'}$, and compute the Jaccard distance

$$D_{\text{Jaccard}} = 1 - \frac{|W_a \cap W_{a'}|}{|W_a \cup W_{a'}|}. \tag{10}$$

The distance is 0 if both token sets are empty. Larger D_{Jaccard} means less overlap in the word types appearing anywhere in dimension titles when the conditioning answer changes. Table 8 summarizes means on 180 held-out questions. **Q-only** uses the same question-only rubric for both answers, so its shift is zero by design. **Q-only×2** compares two independently generated question-only rubrics and shows a small shift (0.09), reflecting lexical variation from repeated generation. **Ours** compares $R(q, a)$ and $R(q, a')$ and achieves a much larger shift of 0.37. This observed separation supports a necessary property of sample-specific evaluation: the rubric changes with the answer, not just the question. Together with the human evaluation in Section 4.2, these results indicate that our rubrics are both more relevant to each (q, a) and more sensitive to answer-specific evaluation needs.

Table 8: Answer sensitivity of induced rubrics on 180 held-out questions. *Q-only* uses a shared question-level rubric, *Q-only×2* compares two independently generated question-level rubrics, and *Ours* compares answer-conditioned rubrics for different answers to the same question.

Comparison	n	Jaccard shift
Q-only	180	0.00
Q-only×2	180	0.09
Ours	180	**0.37**

Table 9: Correlation with independent human scores on 180 rubric-conditioned instances.

Model	Pearson	Spearman
Qwen2.5-14B-Instruct	0.7250	0.7300
Ministral-3-14B-Instruct-2512	0.7396	0.7567
Qwen3.5-9B	0.8205	0.8244
GPT-5	0.8542	0.8615
CriticGen-Qwen2.5-7B	**0.8868**	**0.8969**

E Independent Human Validation of Scoring

We conduct a blinded human evaluation on 180 rubric-conditioned instances, balanced across the six 0–5 score levels. Following the human-evaluation protocol in Section 4, two independent annotators assign scores, and cases with substantial disagreement are re-evaluated by a senior annotator. Table 9 reports correlations between model predictions and the resulting human scores.

CriticGen-Qwen2.5-7B achieves the highest correlation with independent human scores in this comparison. This result provides evidence that its scoring predictions align strongly with human judgments rather than only with GPT-5-derived references.

F Gold-Answer Refinement Evaluation

We evaluate refinement correctness on 5,000 unique question–answer pairs from the REFINE-DATA test split, restricted to GSM8K, Numina-Math, and the applicable subset of NaturalReasoning with reliable gold answers. The sample covers all six 0–5 reference-score levels. For each model output, we extract the last explicit answer span (e.g., content following "Final Answer:" or a dataset-specific answer delimiter), normalize whitespace, punctuation, thousands separators, and equivalent numeric surface forms, and manually resolve cases in which the deterministic parser fails or returns multiple candidates before accuracy computation.

Table 10: Gold-answer refinement results on 5,000 examples. W→R and R→W denote Wrong→Right and Right→Wrong flips. Accuracy and flip rates are percentages; Δ is the accuracy change in percentage points. Bold denotes the best overall result, and underlining denotes the best open-weight result in each column.

Model	Acc.	W→R	R→W	Δ
Original	45.36	–	–	–
Ministral-3-14B-Instruct-2512	90.72	46.66	1.30	+45.36
Mistral-7B-Instruct-v0.3	61.36	23.26	7.26	+16.00
Qwen2.5-32B-Instruct	89.50	44.88	<u>0.74</u>	+44.14
Qwen2.5-14B-Instruct	87.08	42.62	0.90	+41.72
Qwen3.5-4B	86.90	43.62	2.08	+41.54
Qwen3.5-9B	85.74	42.56	2.18	+40.38
Qwen2.5-7B-Instruct	85.48	41.84	1.72	+40.12
GPT-5	**96.84**	**51.92**	**0.44**	**+51.48**
CriticGen-Qwen2.5-7B	<u>91.26</u>	<u>47.30</u>	1.40	<u>+45.90</u>

The original answers achieve 45.36% accuracy. CriticGen-Qwen2.5-7B increases accuracy to 91.26%, with a 47.30% Wrong→Right rate and a 1.40% Right→Wrong rate. GPT-5 reaches the highest overall accuracy at 96.84%. Among the evaluated open-weight models, CriticGen achieves the best accuracy and improves its identical Qwen2.5-7B backbone from 85.48% to 91.26%. This provides an objective ground-truth anchor showing that refinement improves final-answer correctness rather than merely receiving higher scores from the GPT-5 verifier.

G Baseline Models and CriticGen Variants

We compare CriticGen with a diverse set of instruction-following LLM baselines. Unless otherwise specified, all baseline models are evaluated under the same rubric-conditioned prompts and output format, and are not fine-tuned on our constructed data.

- **Qwen2.5-7B-Instruct.** Qwen2.5-7B-Instruct serves as the primary open-weight backbone in our experiments (Yang et al., 2024). It provides a strong 7B-scale instruction-following baseline and is used to assess the zero-shot or prompt-only capability of a model with the same architecture scale as our main CriticGen variant.

- **Qwen2.5-14B-Instruct.** We include Qwen2.5-14B-Instruct as a medium-scale Qwen2.5 baseline (Yang et al., 2024). Comparing it with Qwen2.5-7B-Instruct and Qwen2.5-32B-Instruct allows us to examine whether a larger model capacity alone improves rubric-conditioned evaluation and refinement.

- **Qwen2.5-32B-Instruct.** Qwen2.5-32B-Instruct is used as a larger Qwen2.5-family baseline (Yang et al., 2024). It provides a scale-controlled comparison for evaluating whether CriticGen's gains can be attributed to task-specific supervision rather than simply to increasing the number of model parameters.

- **Qwen3.5-4B.** We include Qwen3.5-4B as a newer small-scale Qwen-series baseline (Qwen Team, 2026a). It allows us to test whether the recent base-model improvements can already support rubric-conditioned evaluation without task-specific fine-tuning.

- **Qwen3.5-9B.** Qwen3.5-9B is used as a newer medium-scale Qwen-series baseline (Qwen Team, 2026b). Together with Qwen3.5-4B, it helps evaluate the robustness of CriticGen's training strategy across different Qwen generations and parameter scales.

- **Mistral-7B-Instruct-v0.3.** We use this model as a non-Qwen open-weight instruction-following baseline (Jiang et al., 2023; Mistral AI, 2024). Its inclusion helps assess whether CriticGen's advantages are specific to the Qwen model family or remain competitive against independently developed open-weight instruction models.

- **Ministral-3-14B-Instruct-2512.** We include Ministral-3-14B-Instruct-2512 as a recent non-Qwen instruction-following baseline (Mistral AI, 2025). It provides an additional comparison point for evaluating CriticGen against contemporary open-weight instruction models.

- **GPT-5.** GPT-5 is included as a strong proprietary model baseline (OpenAI, 2025b,a). In addition to serving as a high-capability comparison model, it is also used as a reference judge in parts of our data construction and verification pipeline.

Our CriticGen variants are obtained by applying the same supervised fine-tuning procedure to select Qwen backbones using our constructed training data. Specifically, CriticGen-Qwen2.5-7B-SFT, CriticGen-Qwen3.5-4B-SFT,

and CriticGen-Qwen3.5-9B-SFT are initialized from Qwen2.5-7B-Instruct (Yang et al., 2024), Qwen3.5-4B (Qwen Team, 2026a), and Qwen3.5-9B (Qwen Team, 2026b), respectively, and trained on RUBRICDATA and REFINEDATA. This design isolates the effect of CriticGen's data and learning objective from the choice of backbone: the base instruction models measure zero-shot or prompt-only rubric-conditioned evaluation ability, while the CriticGen variants measure the effect of learning to generate sample-specific rubrics and to produce the coupled score–reason–revision-suggestion–rewrite trajectory from our supervision.

H Training Details

We train CriticGen variants with supervised fine-tuning (SFT). All training runs are initialized from the corresponding instruction-tuned backbone and use the same serialized output format as described in Section 3. For rubric induction, the model is trained to generate the complete set of induced rubric items, including dimension names and 0–5 scoring criteria. For rubric-conditioned evaluation and refinement, the model is trained to generate the complete target sequence, including the score, reason, executable revision suggestion, and refined answer. We optimize the standard next-token prediction loss with AdamW, using a learning rate of 2×10^{-5}, a cosine learning-rate schedule, a warmup ratio of 0.03, and weight decay of 0.01. Unless otherwise specified, we train for 3 epochs with bf16 precision, gradient checkpointing, and a maximum sequence length of 4096 tokens. The effective batch size is set to 128 through gradient accumulation.

We choose the SFT checkpoint as the main CriticGen model in all reported experiments. This choice keeps the empirical comparison focused on the proposed data construction and evaluation-to-refinement formulation. It also avoids introducing an additional preference-optimization stage whose gains may depend on preference-pair construction, verifier calibration, and reward-model noise rather than on the core CriticGen framework.

I Out-of-Domain Evaluation on Feedback Bench

To further examine the out-of-domain generalization ability of CriticGen, we evaluate it on Feedback Bench from Prometheus (Kim et al., 2024). Feedback Bench is designed for absolute grading with customized score rubrics. Each instance provides an instruction, a response to be evaluated, a reference answer, and a fine-grained scoring rubric, and the evaluator LM is required to assign a scalar score and feedback according to the given rubric.

Table 11: Out-of-domain evaluation on Feedback Bench. We report Pearson, Spearman, and Kendall's τ correlations between model-predicted scores and reference scores.

Model	Pearson	Spearman	Kendall τ
Qwen2.5-7B-Instruct	0.7520	0.7522	0.6606
Qwen3.5-4B	0.7373	0.7451	0.6532
Mistral-7B-Instruct-v0.3	0.6783	0.6801	0.5943
CriticGen-Qwen2.5-7B-SFT	**0.7556**	**0.7612**	**0.6755**

Different from our in-domain REFINEDATA, Feedback Bench is not used during training and follows an independently constructed evaluation format. We therefore use it as an out-of-domain benchmark to test whether the evaluation capability learned by CriticGen transfers to external rubric-conditioned scoring scenarios. To ensure a fair comparison, we compare CriticGen-Qwen2.5-7B-SFT with evaluator LMs of a similar scale, including Qwen2.5-7B-Instruct, Qwen3.5-4B, and Mistral-7B-Instruct-v0.3.

Following the evaluation protocol of Prometheus, we measure the correlation between model-predicted scores and reference scores using Pearson correlation, Spearman correlation, and Kendall's τ. The results are shown in Table 11.

CriticGen-Qwen2.5-7B-SFT achieves the highest observed values across all three correlation metrics among the compared models. Compared with its base model Qwen2.5-7B-Instruct, CriticGen improves Pearson correlation from 0.7520 to 0.7556, Spearman correlation from 0.7522 to 0.7612, and Kendall's τ from 0.6606 to 0.6755. Although the improvement is modest, the consistent observed gains across all metrics suggest potential transfer of CriticGen's rubric-conditioned evaluation capability to out-of-domain customized scoring settings.