TRANSFER ONE TO ALL: ANCHOR QUERY-SPECIFIC RUBRIC SET LEARNING WITH COARSE PREFERENCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work generates query-specific rubric sets to provide fine-grained criteria for evaluating responses in LLM post-training. When scores assigned under rubric sets are used as rewards, the policy learns to favor higher-scoring responses. Therefore, the contribution of rubric sets to policy improvement should be an explicit criterion for rewarding rubric-set generation. Existing methods learn rubric sets based on metrics such as diversity, non-redundancy, and response discrimination, but these measures do not directly assess whether the induced policy updates encourage the desired overall behavior. In this paper, we are the first to study how a coarse preference can serve as an anchor for generating fine-grained, query-specific rubric sets. Although coarse preferences and generated rubric sets evaluate responses using different criteria, they both provide rewards for training the same policy. Therefore, we derive a reference policy-update direction from the coarse preference and augment the rubric generator's existing training reward with an alignment term measuring how closely each rubric set's induced update matches this reference. For each query with a positively aligned candidate, the best-aligned rubric set supplies response scores for updating the policy. Extensive experiments on multiple benchmarks demonstrate improvements in both response discrimination using rubric sets and downstream policy performance. Our code is available on anonymous GitHub [1].

1 INTRODUCTION

Reward models based on rubric sets use explicit multi-dimensional natural language criteria to evaluate responses and provide rewards for large language model (LLM) post-training (Gunjal et al., 2026; Viswanathan et al., 2025). Query-specific rubric sets tailor these criteria to the requirements of individual queries, enabling more fine-grained response evaluation (Viswanathan et al., 2025; Gunjal et al., 2026; Shen et al., 2026).

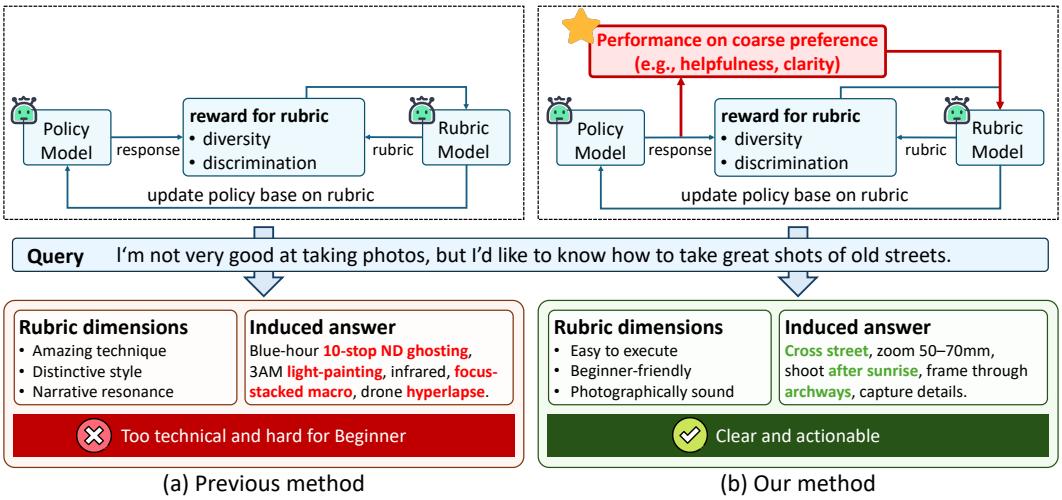
Existing work improves rubric-set quality by developing more comprehensive criteria and improving their ability to distinguish preferred responses. To improve the criteria, methods use reference answers to identify what a good response should contain, or select criteria that cover different aspects of response quality while reducing redundancy (Gunjal et al., 2026; Xie et al., 2025). To improve response discrimination, methods retain rubric sets that help a judge recover preference labels or assign larger score margins to preferred responses (Liu et al., 2026; Xu et al., 2026; Li et al., 2026).

However, as illustrated in Figure 1(a), these rubric-quality measures do not directly assess whether the induced policy updates encourage the desired overall behavior. The contribution to policy improvement should therefore be an explicit criterion for rewarding rubric-set generation. Specifying detailed behavioral targets for every query is impractical, whereas coarse preferences, such as helpfulness and clarity, can provide a shared target across queries. Therefore, we study how such a preference can anchor query-specific rubric-set learning, allowing rubric sets to adapt to individual queries while encouraging updates consistent with the coarse preference.

To address this problem, we propose a framework that guides rubric-set learning through policy-update alignment (Figure 1(b)). Our key observation is that, although the coarse preference and

[1] https://anonymous.4open.science/r/Transfer_One_to_All

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.



(a) Previous method (b) Our method

Figure 1: Previous methods reward rubric sets based on their evaluation criteria and response discrimination. Our method anchors query-specific rubric-set learning on a coarse preference (e.g., helpfulness and clarity), steering the induced policy update toward desired overall behavior.

query-specific rubric sets evaluate responses using different criteria, both provide rewards for training the same policy. Comparing their induced update directions lets us assess consistency with the coarse preference without matching individual evaluation criteria. Specifically, we train an evaluator on coarse scores and use its predictions to construct a reference policy-update direction. For each query, we add the alignment between each candidate rubric set's induced update and this reference to the generator's existing reward. This augments its original objectives, such as response discrimination, and can be incorporated into different rubric-set learning methods. For each query with a positively aligned candidate, the best-aligned rubric set supplies response scores for updating the policy. Our main contributions are as follows:

- We are the first to study the important and practical problem: how coarse preferences can anchor query-specific rubric-set learning.
- We propose a model-agnostic alignment reward that augments existing rubric-set learning rewards with theoretical guarantees.
- Extensive experiments on multiple benchmarks demonstrate improvements in both rubric set quality and downstream policy performance.

2 Related Work

Reward models provide response-level rewards for policy optimization (Stiennon et al., 2020; Ouyang et al., 2022). Rubric-based evaluation makes assessment criteria explicit through natural-language criteria that guide judges in assessing responses (Hashemi et al., 2024; Kim et al., 2024). These criteria may be shared across queries within a task (Mu et al., 2024) or tailored to individual queries (Viswanathan et al., 2025; Gunjal et al., 2026). When rubric-based scores serve as training rewards, these criteria also influence policy optimization.

Rubric construction and refinement. A line of work focuses on constructing and refining evaluation criteria. RaR uses reference answers to formulate query-specific rubric sets (Gunjal et al., 2026). Checklists can use failure modes in candidate responses to construct instruction-specific requirements (Viswanathan et al., 2025). Other methods refine rubric sets through iterative search and selection. Auto-Rubric iteratively induces and refines rubrics, then compresses the candidate pool to retain diverse, non-redundant criteria (Xie et al., 2025). RRD applies recursive decomposition, filtering, and correlation-aware reweighting (Shen et al., 2026). Rubric sets can also evolve during policy training. OnlineRubrics adds criteria based on comparisons between current and control pol-

icy responses (Rezaei et al., 2026). DR Tulu updates criteria using search traces and contrasting responses (Shao et al., 2026).

Training rubric generators and judges. Complementary work trains the models that generate or apply rubric sets. OpenRubrics trains rubric generators and rubric-conditioned judges on curated synthetic data (Liu et al., 2026). Rubric-ARM alternates generator and judge updates, rewarding rubric sets that help recover labeled response preferences (Xu et al., 2026). EvoLM alternates generator and policy updates with a frozen judge, using score margins between paired responses to reward rubric generation. These pairs include responses from current and earlier policy checkpoints (Li et al., 2026). These approaches can improve both criterion content and response discrimination.

However, improvements in criterion content and response discrimination do not directly establish whether the induced policy updates are consistent with a coarse preference. Our method uses such preference as an anchor for query-specific rubric-set learning. Specifically, we derive a reference policy-update direction from the coarse preference and use gradient alignment to augment the generator's existing training reward and guide rubric-set selection for policy optimization.

3 PRELIMINARIES

Rubric sets and policy optimization. A rubric set $r = \{c_j\}_{j=1}^{|r|}$ contains natural-language criteria, each with a scoring rule and weight. For a query q, a rubric generator ρ_ϕ samples $r \sim \rho_\phi(\cdot \mid q)$. An answer policy π_θ samples $G > 1$ responses $\mathbf{a} = (a_1, \ldots, a_G)$, with each $a_n \sim \pi_\theta(\cdot \mid q)$. An LLM judge J evaluates each response under r, assigning $s_n = J(q, r, a_n)$. These scores $s = (s_1, \ldots, s_G)$ serve as rewards for Group Relative Policy Optimization (GRPO) (Shao et al., 2024), with advantages below:

$$A_n = \frac{s_n - \mathrm{mean}(s)}{\mathrm{std}(s) + \epsilon}, \tag{1}$$

where $\epsilon > 0$ stabilizes normalization. The policy maximizes:

$$\mathcal{J}_{\mathrm{GRPO}}(\theta; q, \mathbf{a}, A) = \frac{1}{G}\sum_{n=1}^{G}\frac{1}{|a_n|}\sum_{h=1}^{|a_n|}\Big[\min\big\{v_{n,h}(\theta)A_n, \mathrm{clip}(v_{n,h}(\theta), 1-\delta, 1+\delta)A_n\big\} - \beta\big(\kappa_{n,h}(\theta) - \log\kappa_{n,h}(\theta) - 1\big)\Big], \tag{2}$$

Here δ is the clipping threshold and β weights the KL penalty. At token h of response a_n, $v_{n,h}$ is the ratio of the current policy's token probability to that of the sampling policy, while $\kappa_{n,h}$ is the ratio of the fixed KL reference policy's token probability to that of the current policy. Appendix A.11 gives the full token-level definitions.

Rubric generator optimization. Prior work also uses GRPO to train rubric generators (Xu et al., 2026; Li et al., 2026). Specifically, the rubric generator samples $K > 1$ sets $\mathbf{r} = (r_1, \ldots, r_K)$, with each $r_k \sim \rho_\phi(\cdot \mid q)$. Each rubric set receives a training reward R_k, for example based on response discrimination or format validity. Writing $R = (R_1, \ldots, R_K)$, the training objective is:

$$\widetilde{A}_k = \frac{R_k - \mathrm{mean}(R)}{\mathrm{std}(R) + \epsilon}, \qquad \max_{\phi}\ \mathcal{J}_{\mathrm{GRPO}}(\phi; q, \mathbf{r}, \widetilde{A}). \tag{3}$$

4 METHOD

4.1 MOTIVATION

A coarse preference, such as helpfulness, provides a shared behavioral target for policy training across queries. We use this preference to guide query-specific rubric-set learning by assessing whether the induced policy updates support this target. To directly assess a candidate rubric set's effect on policy, we need to update the policy using rewards computed under that set and then evaluate the updated policy under the coarse preference. Repeating this process for every candidate is costly. We therefore examine each candidate rubric set's induced policy gradient, which describes the local update direction encouraged by its rewards. This avoids the cost of separately updating and evaluating the policy for each candidate.

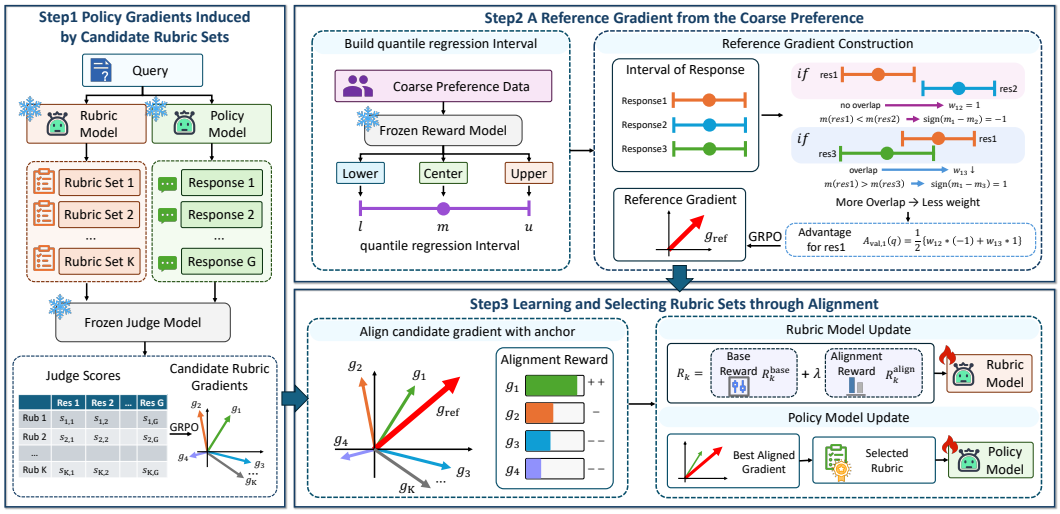


Figure 3: Overview of our method. Candidate rubric sets and the coarse preference induce gradients on the current policy. Their alignment supplements the rubric generator's base reward and selects the rubric set whose response scores are used to update the policy.

To investigate whether a coarse preference can provide a useful reference, we compare policy gradients induced by physician-annotated rubric sets, EvoLM-generated rubric sets, and a coarse preference on HealthBench (Arora et al., 2025). As shown in Figure 2, the coarse preference captures some of the behavioral goals expressed by the annotated rubric sets, while leaving their query-specific requirements unspecified. Rubric sets learned by EvoLM with our coarse-preference guidance (EvoLM + Ours) induce gradients with higher cosine similarity to the annotated-rubric reference than those from EvoLM alone or the coarse preference. We therefore study how to use this coarse preference to guide the policy updates induced by generated rubric sets while retaining the detailed criteria needed for each query.

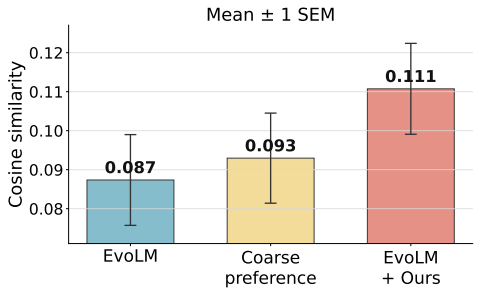


Figure 2: Cosine similarity to gradients induced by annotated rubric sets on HealthBench (higher is better).

4.2 Policy Gradients Induced by Candidate Rubric Sets

To consider the effect of each rubric set on policy training, we evaluate all candidates on the same policy responses. For a query q, a judge J scores each of the G sampled responses under each of the K candidate rubric sets:

$$S(q) \in \mathbb{R}^{K\times G}, \qquad S_{k,n}(q) = J(q, r_k, a_n). \tag{4}$$

Normalizing the response scores in row $S_{k,:}(q)$ as in Equation 1 gives advantages $A_{k,:}(q)$. Substituting these advantages into the GRPO objective yields the policy gradient induced by r_k at the current policy parameters θ_t:

$$g_k(q) = \nabla_\theta \mathcal{J}_{\text{GRPO}}\big(\theta; q, \mathbf{a}, A_{k,:}(q)\big)\big|_{\theta=\theta_t}. \tag{5}$$

We compute all candidate gradients before updating either model, holding the sampled responses, advantages, and KL reference fixed.

4.3 A Reference Gradient from the Coarse Preference

Learning coarse-score intervals. The reference gradient requires coarse scores for responses generated by the current policy. We therefore fit an evaluator $m(q, a)$ to existing coarse scores y using

squared-error regression and apply it to score new policy responses. To make reward construction more robust when applying this evaluator to new data, instead of only using point scores, we further supplement score intervals learned by quantile regression (QR) in reward construction (Koenker & Bassett, 1978). Specifically, we compute residuals $e_i = y_i - m(q_i, a_i)$ on N examples with known coarse scores that were not used to fit m. For $\alpha \in (0, 1)$, two heads $q_l(q, a)$ and $q_h(q, a)$ estimate the lower $\alpha/2$ and upper $1 - \alpha/2$ residual quantiles by minimizing:

$$\mathcal{L}_{\mathrm{QR}} = \frac{1}{N} \sum_{i=1}^{N} \Big[\ell_{\alpha/2}\big(e_i - q_l(q_i, a_i)\big) + \ell_{1-\alpha/2}\big(e_i - q_h(q_i, a_i)\big) \Big], \tag{6}$$

where $\ell_\tau(v) = \max\{\tau v, (\tau - 1)v\}$ is the quantile loss. The lower-quantile loss penalizes overestimation more heavily, whereas the upper-quantile loss penalizes underestimation more heavily. Adding the two predicted residual quantiles to m gives the score interval:

$$\begin{aligned} l(q, a) &= m(q, a) + q_l(q, a), \\ u(q, a) &= m(q, a) + q_h(q, a). \end{aligned} \tag{7}$$

If the quantiles are learned correctly, the interval contains the true coarse score with probability at least $1 - \alpha$. Using this interval reduces reliance on a single point prediction when constructing rewards. Appendix A.1 gives more implementation details.

Interval-weighted pairwise comparisons. To construct interval-based rewards, ideally, when two score intervals overlap, the intervals do not clearly establish which response has the higher coarse score. We therefore assign smaller weights to comparisons with overlapping score intervals when computing the response advantages. Specifically, for a query q, the current policy generates G responses $\mathbf{b}(q) = (b_1, \ldots, b_G)$. Each response b_n has a predicted coarse score $m_n = m(q, b_n)$ and a score interval $[l_n, u_n] = [l(q, b_n), u(q, b_n)]$. For $n, p \in \{1, \ldots, G\}$ with $n \neq p$, the comparison weight between responses b_n and b_p is:

$$w_{np} = 1 - \frac{\big[\min(u_n, u_p) - \max(l_n, l_p)\big]_+}{\max\{\min(u_n - l_n, u_p - l_p), \epsilon_C\}}, \tag{8}$$

where $[x]_+ = \max\{x, 0\}$ and $\epsilon_C > 0$ stabilizes the denominator. The fraction measures the overlap relative to the narrower interval. Thus, disjoint intervals receive weight one, while greater relative overlap gives a smaller weight. For example, intervals $[0.5, 1]$ and $[0.7, 1.3]$ overlap by 0.3; the narrower interval has width 0.5, giving $w_{np} = 1 - 0.3/0.5 = 0.4$.

Averaging the weighted pairwise comparisons gives the response advantages. We use them directly in GRPO to compute the reference gradient at $\theta = \theta_t$, averaged over a reference query batch $\mathcal{B}_{\mathrm{val}}$ separate from the model-update queries:

$$A_{\mathrm{val},n}(q) = \frac{1}{G - 1} \sum_{p \neq n} w_{np} \operatorname{sign}(m_n - m_p), \tag{9}$$

$$g_{\mathrm{ref}} = \frac{1}{|\mathcal{B}_{\mathrm{val}}|} \sum_{q \in \mathcal{B}_{\mathrm{val}}} \nabla_\theta \mathcal{J}_{\mathrm{GRPO}}\big(\theta; q, \mathbf{b}(q), A_{\mathrm{val}}(q)\big). \tag{10}$$

4.4 Learning and Selecting Rubric Sets through Alignment

Based on $g_k(q)$ and g_{ref}, we use their cosine similarity as the alignment reward:

$$R_k^{\mathrm{align}}(q) = \frac{\langle g_k(q), g_{\mathrm{ref}} \rangle}{\|g_k(q)\| \, \|g_{\mathrm{ref}}\|}. \tag{11}$$

To preserve query-specific rubric learning, we retain the underlying method's base reward $R_k^{\mathrm{base}}(q)$, which may encourage diversity or response discrimination, and add the alignment term:

$$R_k(q) = R_k^{\mathrm{base}}(q) + \lambda R_k^{\mathrm{align}}(q), \tag{12}$$

where $\lambda \geq 0$ controls the alignment term. This additive formulation is model-agnostic and can be incorporated into different rubric-learning methods. We alternate between updating the rubric generator with GRPO using $R_k(q)$ over all K candidates and updating the policy using response scores under the best-aligned rubric set. Both updates are restricted to queries with $\max_{1 \leq k \leq K} R_k^{\mathrm{align}}(q) > 0$, while the evaluator remain frozen. The training algorithm and update schedule are detailed in Appendices A.2 and A.3.

4.5 THEORETICAL ANALYSIS

In this section, we investigate when coarse guidance helps select a better update from the same candidate rubric sets, comparing our coarse-guided gradient $\hat{g}_q$ with only the query-specific rubric gradient g_f. Along candidate update directions, we model the task gradient $g_\star$ as a fixed query-specific component d plus an unresolved shared component z. The coarse reference y measures z with independent noise ξ:

$$g_\star = d + z, \quad y = z + \xi, \quad z \sim \mathcal{N}(0, \tau^2 I), \quad \xi \sim \mathcal{N}(0, \sigma^2 I/M). \tag{13}$$

Here $M = |\mathcal{B}_{\text{val}}|$; τ^2 and σ^2/M measure shared uncertainty and reference noise, with $\tau, \sigma > 0$. Coarse-guided selection may sacrifice some query-specific benefit, bounded by the range across candidate updates, and we include $g_0 = 0$ to represent skipping an update:

$$\Delta_q = \max_k \langle d, g_k \rangle - \min_k \langle d, g_k \rangle. \tag{14}$$

Theorem 4.1 (Information benefit of coarse-guided updates)**.** *Under this model, with at least one nonzero candidate gradient and a locally L-smooth task objective F, any step $\eta > 0$ within the smooth neighborhood satisfies*

$$\mathbb{E}[F(\theta_t + \eta \hat{g}_q) - F(\theta_t + \eta g_f)] \geq \eta\big(A_q(M) - \Delta_q\big) - L\eta^2 G_q^2. \tag{15}$$

The information benefit is $A_q(M) = m_q \tau^2 W_q / \sqrt{\tau^2 + \sigma^2/M}$. The candidate-gradient norm bounds m_q, G_q, directional spread W_q, and proof are detailed in Appendix B.

Thus coarse guidance helps when its information benefit exceeds the possible loss in query-specific benefits, i.e., $A_q(M) > \Delta_q$, and the update step is sufficiently small.

5 EXPERIMENTS

In this section, we conduct a comprehensive empirical evaluation to demonstrate the efficacy of our proposed method, centered on the following research questions (RQs): **RQ1**: *Are the learned rubric sets better?* In Subsection 5.1, we compare the discriminative ability of different rubric generators. **RQ2**: *Does our method improve policy performance?* In Subsection 5.2, we evaluate downstream policies performance. **RQ3**: *Does coarse-preference guidance improve policy performance across datasets?* In Subsection 5.3, we evaluate performance under varying degrees of domain shift. **RQ4**: *Is the coarse-preference evaluator reliable?* In Subsection 5.4, we examine score and interval trends across model scales. **RQ5**: *How sensitive is our method to hyperparameters?* In Subsection 5.5, we assess the sensitivity of the rubric and policy model.

Datasets and models. Following (Li et al., 2026), we train both the rubric generator and the answer policy on Tulu 3 (Lambert et al., 2025). For the coarse-preference evaluator, we use the annotated *helpfulness*, *correctness*, and *coherence* dimensions from HelpSteer (Wang et al., 2024b) as the supervision source. Further dataset information is provided in Appendix A.4. The rubric generator and policy are separate Qwen3-4B models (Yang et al., 2025) with independent LoRA adapters ($r = 16$); the judge uses a Qwen3-8B backbone, and the coarse preference predictor uses a frozen FsfairX-LLaMA3-RM-v0.1 encoder (sfairXC, 2024). The performance of the models is evaluated through API calls to DeepSeek V4-Flash, GPT5.6-Luna, and Gemini 3.7-Flash.

Implementation details. We set the alignment-reward weight in Equation 12 to $\lambda = 0.1$. Both modules are optimized using GRPO with a learning rate of 10^{-6} and a KL coefficient of 0.001, where 4 responses (roll-outs) are sampled per question. In each round, the rubric generator and the answer policy are updated together. Additional training and implementation details can be found in Appendix A.3.

Baselines. We compare three categories of baselines: (i) *backbone*, the Qwen3-4B model; (ii) *prompted induction*, namely RaR (Gunjal et al., 2026), RRD (Shen et al., 2026), and Auto-Rubric (Xie et al., 2025); and (iii) *trained rubric generators*, OpenRubrics (Liu et al., 2026), EvoLM (Li et al., 2026) and Rubric-ARM (Xu et al., 2026). Our method is designed as a plug-and-play module and is implemented on both EvoLM and Rubric-ARM under the same training protocol. Further baseline details are in Appendix A.5.

Table 1: Evaluation of rubric-set discriminative ability on RewardBench 2 and JudgeBench under three judges. We report RewardBench 2 scores and JudgeBench accuracy (%). Each +Ours variant is paired with its original framework, and all rubric generators share one evaluator per panel. **Bold** and underline mark the best and second-best entries per column within each evaluator.

	RewardBench2							JudgeBench				
Method	Fact	IF	Math	Safe	Focus	Ties	Avg.	Know	Reas	Math	Code	Overall
DeepSeek v4												
Backbone												
Qwen3-4B	34.98	43.94	73.32	50.91	36.72	83.29	53.86	31.01	35.98	43.89	33.14	35.29
Trained rubric generator												
OpenRubrics	38.98	46.88	75.32	51.28	39.91	82.07	55.74	30.22	38.93	42.89	34.73	35.66
EvoLM	41.12	47.40	78.28	52.69	48.89	83.98	58.73	32.52	36.20	44.44	34.51	36.37
EvoLM+Ours	44.23	**50.18**	**78.47**	53.96	**51.91**	92.29	**61.84**	35.17	38.91	45.11	35.13	38.56
Rubric-ARM	39.86	46.82	74.93	50.80	39.10	84.24	55.96	32.16	37.22	44.91	36.00	36.69
Rubric-ARM+Ours	**47.57**	48.69	75.74	**55.27**	46.15	**93.20**	61.10	**36.21**	**42.52**	**46.46**	**37.76**	**40.51**
GPT 5.6												
Backbone												
Qwen3-4B	55.59	52.47	79.69	49.32	69.59	73.26	63.32	64.07	73.95	80.91	84.61	73.32
Trained rubric generator												
OpenRubrics	60.65	55.16	80.46	49.44	74.95	74.19	65.81	65.14	78.59	85.56	86.15	75.89
EvoLM	62.51	62.64	80.65	57.65	71.54	81.85	69.47	67.10	82.89	88.46	89.18	78.75
EvoLM+Ours	64.83	**73.40**	79.80	**59.32**	80.18	**89.75**	**74.55**	73.89	**91.04**	91.03	88.75	**84.57**
Rubric-ARM	60.84	54.19	83.32	52.28	70.76	81.42	67.14	68.18	80.84	82.00	88.26	77.72
Rubric-ARM+Ours	**67.82**	64.04	**84.63**	53.15	**83.30**	86.55	73.25	**75.15**	84.59	**91.78**	**89.67**	83.84
Gemini 3.7												
Backbone												
Qwen3-4B	53.79	51.25	80.20	49.26	69.54	80.18	64.04	72.65	83.59	73.67	54.26	75.33
Trained rubric generator												
OpenRubrics	56.49	54.46	82.43	51.26	71.31	81.22	66.20	73.73	85.77	74.33	55.66	76.69
EvoLM	57.30	55.03	83.29	52.80	73.91	82.23	67.43	74.88	86.93	74.26	57.26	77.74
EvoLM+Ours	**58.81**	**57.59**	**86.33**	53.90	74.59	84.36	69.26	76.94	**90.43**	75.94	**57.88**	79.98
Rubric-ARM	55.94	55.50	83.26	51.62	75.00	83.75	67.51	75.95	87.26	76.67	55.82	78.56
Rubric-ARM+Ours	57.11	56.87	85.09	**54.92**	**75.58**	**86.82**	**69.40**	**78.95**	88.33	**77.14**	56.46	**80.50**

5.1 Evaluation of Rubric Sets (RQ1)

Following EvoLM (Li et al., 2026), we evaluate rubric-set discrimination on RewardBench 2 and JudgeBench. RewardBench 2 contains six subsets: factuality, instruction following, mathematics, safety, focus, and ties, each providing queries, candidate responses, and preference labels. JudgeBench covers knowledge, reasoning, mathematics, and coding, with response pairs labeled by correctness. For each query, each generator produces a rubric set that a frozen judge uses to evaluate the candidate responses. We measure agreement with the benchmark labels under the respective evaluation protocols. In Table 1, Avg. denotes the unweighted mean of the six RewardBench 2 subset scores, while Overall denotes accuracy across all JudgeBench examples. Prompted rubric induction methods are omitted because they do not train a rubric generator for evaluation queries. Results across three judges show that our method improves response discrimination.

5.2 Performance Comparison on Policy (RQ2)

To evaluate whether the generated rubric sets improve policy performance, we assess policies trained with each method using the supplied rubrics from ResearchQA and UltraFeedback. Each policy answers the same test queries, and three frozen judges evaluate the resulting responses. We report two metrics: score, the mean response score assigned under the supplied rubrics, and win rate, the proportion of comparisons in which the judge prefers the trained policy's response over the base policy's response to the same query. Each comparison requires a binary choice, with no ties allowed. Table 2 shows that our method achieves higher scores and win rates in most settings.

5.3 Policy Transfer Across Datasets (RQ3)

The coarse preferences used in our method come from HelpSteer. To examine their transferability, we select target datasets with supplied rubrics and investigate how their distance from HelpSteer

Table 2: Downstream policy evaluation on ResearchQA and UltraFeedback under three frozen judges. For each judge and benchmark we report the score and the win rate (%) against the base policy. **Bold** and underline mark the best and second-best per column.

	DeepSeek v4				GPT 5.6				Gemini 3.7			
	ResearchQA		UltraFeedback		ResearchQA		UltraFeedback		ResearchQA		UltraFeedback	
Method	score	win	score	win	score	win	score	win	score	win	score	win
Backbone												
Qwen3-4B	37.12	–	59.64	–	39.46	–	65.57	–	40.15	–	72.24	–
Prompted rubric-set induction												
RaR	39.86	63.45	60.14	61.84	40.81	68.75	69.87	72.25	39.44	59.86	73.12	72.55
RRD	38.43	65.48	62.26	64.40	41.06	61.83	66.19	76.29	41.73	72.01	74.75	75.79
Auto-Rubric	41.61	56.55	62.91	65.17	45.65	66.74	70.95	74.27	42.19	**81.21**	73.31	74.28
Trained rubric generator												
OpenRubrics	42.31	66.52	62.89	70.87	44.46	64.60	69.36	69.09	41.47	68.24	72.73	69.91
EvoLM	44.88	71.49	63.46	68.54	45.97	69.21	70.97	75.06	42.27	66.49	77.39	76.90
EvoLM+Ours	**47.11**	**76.18**	65.58	71.57	**46.83**	**72.96**	71.36	74.77	**45.48**	70.75	**78.85**	78.99
Rubric-ARM	42.66	69.56	67.88	65.17	43.79	67.60	70.08	79.71	41.20	68.21	75.84	78.11
Rubric-ARM+Ours	45.97	71.23	**69.94**	**74.10**	44.23	70.95	**72.84**	**82.66**	42.51	69.14	76.13	**81.44**

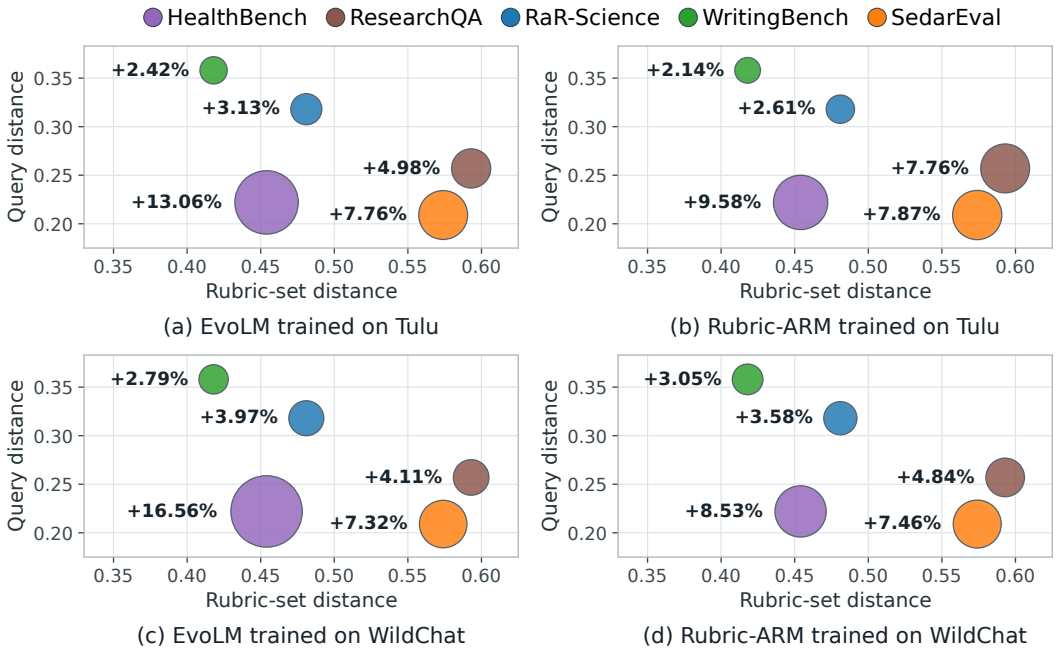


Figure 4: Policy transfer under the DeepSeek judge. Axes show cosine distances to HelpSteer in rubric (x) and query (y) representations. Bubble areas indicate relative score gains over baselines.

relates to the gains from our method. We use Qwen3-Embedding-8B to embed queries and rubrics separately, then compute cosine distances between the mean embeddings of HelpSteer and each target dataset in both representation spaces. The DeepSeek judge evaluates policy responses using the target dataset's supplied rubrics. We compare policy scores with and without our method and examine how the gains vary with query and rubric-set distances. Figure 4 shows generally larger gains on datasets whose queries are closer to HelpSteer.

5.4 Evaluator Reliability (RQ4)

To assess the reliability of the coarse-preference evaluator, we have Qwen3-1.7B, 4B, 8B, and 14B answer the same queries from HelpSteer 2 and ResearchQA. The same frozen evaluator predicts a score and interval for each response, and we summarize the mean scores, interval bounds, and

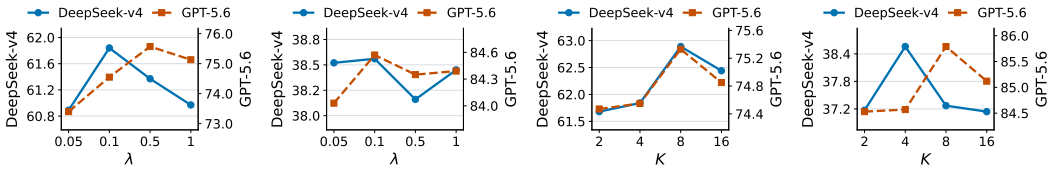
Table 3: Evaluator predictions for responses generated by Qwen3 models of different scales on HelpSteer2 and ResearchQA. $\bar{m}$, $\bar{l}$, and $\bar{u}$ are mean predicted scores and interval bounds; Width is the mean interval width.

(a) HelpSteer 2

Policy model	$\bar{m}$	$\bar{l}$	$\bar{u}$	Width
Qwen3-1.7B	3.36	2.81	3.79	0.98
Qwen3-4B	3.47	2.94	3.84	0.89
Qwen3-8B	3.54	3.03	3.88	0.85
Qwen3-14B	3.55	3.05	3.89	0.84

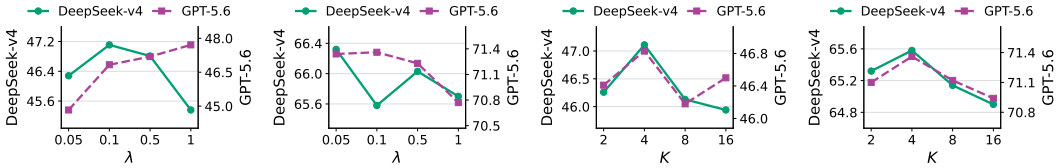
(b) ResearchQA

Policy model	$\bar{m}$	$\bar{l}$	$\bar{u}$	Width
Qwen3-1.7B	3.69	3.19	3.97	0.79
Qwen3-4B	3.77	3.29	3.99	0.70
Qwen3-8B	3.80	3.33	4.00	0.67
Qwen3-14B	3.81	3.35	4.00	0.65



(a) RB2 (λ) (b) JB (λ) (c) RB2 (K) (d) JB (K)

Figure 5: Sensitivity of rubric-set discriminative accuracy to λ and K under two judges.



(a) RQA (λ) (b) UF (λ) (c) RQA (K) (d) UF (K)

Figure 6: Sensitivity of downstream policy scores to λ and K under two judges.

widths for each model size. Table 3 shows that mean predicted scores and lower bounds increase with model size on both datasets, while intervals narrow. These trends are consistent with the expected improvement in response quality at larger model sizes, supporting the reasonableness of the evaluator's predictions.

5.5 Hyperparameter Sensitivity (RQ5)

To examine hyperparameter sensitivity, we vary the alignment-reward weight $\lambda \in \{0.05, 0.1, 0.5, 1\}$ and the number of candidate rubric sets per query $K \in \{2, 4, 8, 16\}$ for EvoLM+Ours, changing one parameter at a time while holding other settings fixed. Following the protocols in RQ1 and RQ2, we measure rubric-set discrimination on RewardBench 2 and JudgeBench and policy scores on ResearchQA and UltraFeedback under the DeepSeek and GPT judges. Figures 5 and 6 show modest fluctuations in both metrics, suggesting limited sensitivity within the tested ranges. Appendix A.8 provides additional sensitivity experiments.

6 Conclusion

We presented a framework that anchors query-specific rubric-set learning on coarse preferences through policy-update alignment. By comparing the gradients induced by candidate rubric sets with a shared reference derived from existing coarse scores, our method connects rubric evaluation to the behavior encouraged by its training rewards. The alignment signal augments the generator's existing reward and selects rubric sets for policy updates, preserving query-specific criteria while providing a common behavioral target. Our analysis characterizes when the information supplied by coarse preferences outweighs the cost of trading off query-specific details. Experiments with two rubric-learning frameworks demonstrate improvements in response discrimination and downstream policy performance across multiple benchmarks. One possible limitation is that transfer to specialized domains may benefit from coarse-score data that more closely match the target tasks.

AI USE STATEMENT

We used large language model (LLM) tools to edit and polish parts of the manuscript, format tables, and draft and revise scripts for experimental figures. The authors reviewed and verified all AI-assisted text, code, figures, and interpretations, and take responsibility for the final content.

ETHICS STATEMENT

This study uses publicly available datasets and existing human ratings and does not involve collecting new data from human subjects. The work primarily focuses on model training and does not raise any ethical concerns.

REPRODUCIBILITY STATEMENT

A link to the anonymized codebase is provided in the abstract. Experimental details, including hyperparameters and model configurations, are provided in the Appendix.

REFERENCES

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. HealthBench: Evaluating Large Language Models Towards Improved Human Health. *arXiv preprint arXiv:2505.08775*, 2025.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting Language Models With Scaled AI Feedback. In *International Conference on Machine Learning*, 2024.

Zhiyuan Fan, Weinong Wang, Xing Wu, and Debing Zhang. SedarEval: Automated Evaluation Using Self-Adaptive Rubrics. In *Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing*, 2024.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. In *International Conference on Learning Representations*, 2026.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. LLM-Rubric: A Multidimensional, Calibrated Approach to Automated Evaluation of Natural Language Texts. In *Annual Meeting of the Association for Computational Linguistics*, 2024.

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. In *International Conference on Learning Representations*, 2024.

Roger Koenker and Gilbert Bassett, Jr. Regression Quantiles. *Econometrica*, 46(1):33–50, 1978. doi: 10.2307/1913643.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, et al. Tulu 3: Pushing Frontiers in Open Language Model Post-Training. In *Conference on Language Modeling*, 2025.

Shuyue Stella Li, Rui Xin, Teng Xiao, Yike Wang, Rulin Shao, Zoey Hao, Melanie Sclar, Sewoong Oh, Faeze Brahman, Pang Wei Koh, and Yulia Tsvetkov. EvoLM: Self-Evolving Language Models Through Co-Evolved Discriminative Rubrics. *arXiv preprint arXiv:2605.03871*, 2026.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. OpenRubrics: Towards Scalable Synthetic Rubric Generation for Reward Modeling and LLM Alignment. In *Annual Meeting of the Association for Computational Linguistics*, 2026.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing Reward Model Evaluation. In *International Conference on Learning Representations*, 2026.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule Based Rewards for Language Model Safety. In *Advances in Neural Information Processing Systems*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions With Human Feedback. In *Advances in Neural Information Processing Systems*, 2022.

MohammadHossein Rezaei, Robert Vacareanu, Zihao Wang, Clinton Wang, Bing Liu, Yunzhong He, and Afra Feyza Akyürek. Online Rubrics Elicitation From Pairwise Comparisons. In *International Conference on Machine Learning*, 2026.

sfairXC. FsfairX-LLaMA3-RM-V0.1. Hugging Face model card, 2024.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen-tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. DR Tulu: Reinforcement Learning With Evolving Rubrics for Deep Research. In *International Conference on Machine Learning*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

William F. Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. Rethinking Rubric Generation for Improving LLM Judge and Reward Modeling for Open-Ended Tasks. *arXiv preprint arXiv:2602.05125*, 2026.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to Summarize With Human Feedback. In *Advances in Neural Information Processing Systems*, 2020.

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. In *International Conference on Learning Representations*, 2025.

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. In *Advances in Neural Information Processing Systems*, 2025.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. HelpSteer2: Open-Source Dataset for Training Top-Performing Reward Models. In *Advances in Neural Information Processing Systems*, 2024a.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. HelpSteer: Multi-Attribute Helpfulness Dataset for SteerLM. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2024b.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. WritingBench: A Comprehensive Benchmark for Generative Writing. In *Advances in Neural Information Processing Systems*, 2025.

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. *arXiv preprint arXiv:2510.17314*, 2025. doi: 10.48550/arXiv.2510.17314.

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tuo Zhao, and Haoyu Wang. Alternating Reinforcement Learning for Rubric-Based Reward Modeling in Non-Verifiable LLM Post-Training. In *International Conference on Machine Learning*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Li S. Yifei, Allen Chang, Chaitanya Malaviya, and Mark Yatskar. ResearchQA: Evaluating Scholarly Question Answering at Scale Across 75 Fields With Survey-Mined Questions and Rubrics. *Transactions of the Association for Computational Linguistics*, 14:1365–1389, 2026. doi: 10.1162/tacl.a.732.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. WildChat: 1M ChatGPT Interaction Logs in the Wild. In *International Conference on Learning Representations*, 2024.

A Additional Method and Experimental Details

A.1 Quantile Evaluator Construction

The evaluator in Section 4.3 uses a frozen encoder to map each query–response pair to a fixed-length embedding. A score head $m(q, a)$ is trained on existing coarse scores by minimizing $\big(m(q, a) - y\big)^2$. After fitting, this head is frozen. A separate set of examples with known coarse scores supplies the residual targets $y - m(q, a)$ for the two quantile heads q_l and q_h. Both heads take the same frozen embedding as input and are trained with Equation 6, at quantile levels $\alpha/2$ and $1 - \alpha/2$, respectively. The resulting QR bounds are given in Equation 7 and are used directly for the pairwise comparisons in Equation 8. The score head, quantile heads, and encoder remain frozen throughout rubric-generator and policy training.

A.2 Algorithm of Training

Algorithm 1 describes an update of the active module, either the rubric generator or the policy. Training details are provided in Appendix A.3. If either gradient is zero, we assign zero alignment. Alignment rewards are held fixed during GRPO, with no backpropagation through the gradient comparison.

A.3 Implementation Details

The rubric generator and answer policy use Qwen3-4B backbones with independent LoRA adapters ($r = 16$), and the judge uses a Qwen3-8B backbone. We set $\lambda = 0.1$ in Equation 12 and optimize both modules with GRPO using a learning rate of 10^{-6} and a KL coefficient of 0.001. For each prompt, we sample $K = 4$ candidate rubric sets and $G = 4$ policy responses. The rubric generator and answer policy each receive 200 update steps in total, with a batch size of 64 prompts per step. All methods use the same policy-update budget. A separate set of 64 prompts is used to compute the reference gradient at each step. The rubric-generator and policy adapters are updated during training; Rubric-ARM also retains its judge updates.

The coarse-preference evaluator separately predicts helpfulness, correctness, and coherence using the corresponding raw 0–4 labels from HelpSteer. The three attributes are used as separate training targets rather than combined into a scalar target y. At inference time, the three predicted scores are averaged with equal weights. The three dimensions share the same prompt-disjoint partition: approximately 68.5% of prompts are allocated to score-head fitting, and the remaining 31.5% to

Algorithm 1 One update step of Preference-Anchored Rubric Set Learning

Require: Rubric generator ρ_{ϕ_t}, policy π_{θ_t}, judge J, frozen evaluator, disjoint query batches $\mathcal{B}_{\mathrm{tr}}, \mathcal{B}_{\mathrm{val}}$, weight λ

for each query $q \in \mathcal{B}_{\mathrm{val}}$ **do**
  Sample G responses from π_{θ_t} and evaluate their coarse scores and intervals
  Compute $A_{\mathrm{val}}(q)$ using Equations 8–9
end for
Compute g_{ref} using Equation 10
for each query $q \in \mathcal{B}_{\mathrm{tr}}$ **do**
  Sample K rubric sets from ρ_{ϕ_t} and G shared responses from π_{θ_t}
  Compute the score matrix $S(q)$ and row-wise advantages $A_{k,:}(q)$
  Compute $g_k(q)$ and $R_k^{\mathrm{align}}(q)$ for each candidate
  Select $k^*(q) = \arg\max_{1 \le k \le K} R_k^{\mathrm{align}}(q)$
  if $R_{k^*(q)}^{\mathrm{align}}(q) > 0$ **then**
    Retain all K candidate rubric sets with rewards $R_k(q) = R_k^{\mathrm{base}}(q) + \lambda R_k^{\mathrm{align}}(q)$
    Retain the response group with advantages $A_{k^*(q),:}(q)$
  end if
end for
Update the active module with its GRPO objective on the retained groups

quantile-head fitting and reference residual construction. Quantile regression uses $\alpha = 0.2$, with lower and upper quantile levels of $\tau_{\mathrm{low}} = 0.1$ and $\tau_{\mathrm{high}} = 0.9$, respectively.

Model performance is evaluated through API calls to DeepSeek V4-Flash[2], GPT-5.6-Luna[3], and Gemini 3.7-Flash[4]. For RQ1, the RewardBench 2 accuracy is the mean of its six category scores, whereas JudgeBench overall accuracy is the mean of its four category accuracies weighted by the number of examples in each category. For RQ2, we report normalized pointwise scores and win rates against the base policy. On ResearchQA, each judge scores individual rubric items on a 0–4 scale; we average the item scores within each response, divide by 4, and multiply by 100. On UltraFeedback, each judge scores individual assessment dimensions on a 1–5 scale; we average the dimension scores within each response, divide by 5, and multiply by 100.

A.4 Dataset Descriptions

We summarize the datasets used throughout the paper.

- **Tulu 3** (Lambert et al., 2025)[5] provides a diverse collection of instruction-following data. Our Tulu 3 prompt pool contains 271k prompts, from which we randomly sample subsets to train the rubric generator and policy.
- **WildChat** (Zhao et al., 2024)[6] contains naturally occurring user–assistant conversations. The linked public release includes 529,428 conversations, whose user queries provide an additional source of training prompts.
- **HelpSteer** (Wang et al., 2024b)[7] contains 37,120 prompt–response examples annotated with five human-rated attributes on a 0–4 scale. We use helpfulness, correctness, and coherence to supervise the coarse-preference evaluator.
- **HelpSteer 2** (Wang et al., 2024a)[8] contains 21,362 prompt–response examples annotated with the same five human-rated attributes. We use its prompts to generate responses from Qwen3 models of different scales.

[2] https://www.deepseek.com/en/news/v4-preview/
[3] https://openai.com/index/gpt-5-6/
[4] https://ai.google.dev/gemini-api/docs/models/gemini-3.7-flash
[5] https://huggingface.co/datasets/allenai/tulu-3-sft-mixture
[6] https://huggingface.co/datasets/allenai/WildChat
[7] https://huggingface.co/datasets/nvidia/HelpSteer
[8] https://huggingface.co/datasets/nvidia/HelpSteer2

- **RewardBench 2** (Malik et al., 2026)[9] contains 1,865 evaluation examples covering factuality, instruction following, mathematics, safety, focus, and ties. We evaluate using its official scoring protocol.
- **JudgeBench** (Tan et al., 2025)[10] evaluates response discrimination across knowledge, reasoning, mathematics, and coding. Its public release includes 350 GPT-4o response pairs and 270 Claude-3.5-Sonnet response pairs.
- **ResearchQA** (Yifei et al., 2026)[11] provides scholarly questions and associated evaluation rubrics mined from survey papers across 75 fields. Its public release includes a test set of 3,750 questions and a test-mini subset of 776 questions. We use the full validation split of 703 questions exclusively for evaluation, with its rubrics supporting policy evaluation and its prompts supporting evaluator analysis.
- **UltraFeedback** (Cui et al., 2024)[12] contains approximately 64,000 prompts with four responses per prompt and feedback on instruction following, truthfulness, honesty, and helpfulness. We use these assessment dimensions for downstream policy evaluation.
- **HealthBench** (Arora et al., 2025)[13] contains 5,000 health conversations with physician-written evaluation rubrics. These expert annotations serve as the rubric reference for gradient comparison and as evaluation criteria for transfer experiments.
- **RaR-Science** (Gunjal et al., 2026)[14] contains 22,917 science questions with reference answers and checklist-style rubrics. We use the supplied criteria to evaluate policy responses.
- **WritingBench** (Wu et al., 2025)[15] covers diverse writing tasks with query-specific evaluation criteria. Its updated public release contains 1,000 queries. We use the supplied criteria for transfer evaluation.
- **SedarEval** (Fan et al., 2024)[16] contains 1,000 Chinese queries across eight task categories with self-adaptive scoring rubrics. We use the supplied rubrics to evaluate policy responses.

A.5 Baseline Descriptions

Prompted rubric-set induction.

- **RaR** (Gunjal et al., 2026). Uses an off-the-shelf LLM to generate prompt-specific checklist criteria grounded in reference answers. An LLM judge scores policy responses against these criteria, and the aggregated scores serve as rewards for on-policy reinforcement learning.
- **RRD** (Shen et al., 2026). Prompts an LLM to propose rubrics from a task prompt and sample responses, then recursively decomposes criteria that fail to distinguish responses. It filters misaligned and redundant criteria and applies correlation-aware weights when aggregating the resulting rubric scores.
- **Auto-Rubric** (Xie et al., 2025). Iteratively induces criteria from preference pairs, verifies whether they recover the observed preferences, and refines failures. It then compresses the candidate pool into a compact, hierarchical rubric set using a coding-rate objective, without updating model weights.

In our implementation, all three prompted baselines use Qwen3-4B with fixed generator parameters to produce rubrics from prompts constructed using Tulu 3 training examples. These methods generate fixed rules tailored to specific scenarios, which cannot be directly transferred to the general rubric evaluation setting in RQ1. We therefore exclude them from Table 1. However, their rubrics can still provide reward signals for updating policy parameters, allowing us to evaluate the resulting policies in RQ2.

[9] https://huggingface.co/datasets/allenai/reward-bench-2
[10] https://github.com/ScalerLab/JudgeBench
[11] https://huggingface.co/datasets/realliyifei/ResearchQA
[12] https://huggingface.co/datasets/openbmb/UltraFeedback
[13] https://huggingface.co/datasets/openai/healthbench
[14] https://huggingface.co/datasets/ScaleAI/RaR-Science
[15] https://github.com/X-PLUG/WritingBench
[16] https://github.com/wwn1233/sedareval

Table 4: Training cost with the judge fixed to Qwen3-8B and the rubric generator and answer policy fixed to Qwen3-4B.

Method	Training hours	Peak single-GPU memory
Rubric-ARM	46.83	79.34 GiB
Rubric-ARM + Ours	52.82	83.34 GiB
EvoLM	40.35	84.12 GiB
EvoLM + Ours	44.26	88.17 GiB

Trained rubric generators.

- **OpenRubrics** (Liu et al., 2026). Constructs training rubrics from ranked responses by extracting explicit hard rules and implicit quality principles, then filters examples by preference-label consistency. It uses the resulting data to fine-tune a prompt-conditioned rubric generator and a rubric-conditioned pairwise reward model. In our implementation, we distill rubrics generated by GPT-5.6-Luna into the rubric generator.
- **EvoLM** (Li et al., 2026). Alternates between training a rubric generator and a policy within a single language model. The generator learns criteria that help a frozen judge distinguish policy-generated preference pairs, while the policy learns from the judge's rubric-conditioned scores.
- **Rubric-ARM** (Xu et al., 2026). Warm-starts a rubric generator and a pairwise judge with supervised fine-tuning, then alternates reinforcement-learning updates to the two components. The generator is rewarded for producing rubrics that help the fixed judge recover preference labels; the judge is trained to predict those labels under rubrics from the fixed generator.

For the experiments in this paper, EvoLM and Rubric-ARM use separate rubric-generator and answer-policy LoRA adapters with the training schedule above. EvoLM+Ours keeps the judge frozen, while Rubric-ARM trains the judge in both the baseline and its +Ours variant.

A.6 Case Study

As shown in Figure 7, the human rubric emphasizes each reagent's role and how they work together. EvoLM combines the roles and assigns weight to terminology and structure, leaving the joint mechanism weakly specified. EvoLM+Ours instead evaluates the roles separately and emphasizes their interaction, leading to a response that explains oxidation followed by adsorption. The judge rates this response higher, illustrating that our rubric and policy model can better capture the substantive requirements of the query.

A.7 Training Efficiency

Local training runs use $8\times$ NVIDIA H20 GPUs, each with 96 GB of memory. With a Qwen3-8B judge backbone and Qwen3-4B rubric-generator and policy backbones, Table 4 reports elapsed training hours and peak single-GPU memory. Our method incurs modest additional training time.

A.8 Hyperparameter Sensitivity to Response Count

We vary the number of responses per query $G \in \{2, 4, 8, 16\}$ for EvoLM+Ours, holding other settings fixed. Figure 8 reports rubric-set accuracy in (a,b) and policy scores in (c,d). Both metrics show modest fluctuations across the tested settings. Increasing G does not consistently improve either outcome, suggesting limited sensitivity within the tested range.

A.9 Backbone Sensitivity of the Rubric and Policy Models

We vary the rubric-generator and answer-policy backbones across Qwen3-4B, Qwen3-8B, and Llama3.1-8B while keeping the judge backbone and evaluation protocol fixed. Table 5 reports two distinct evaluations: rubric-set discriminative accuracy on RewardBench 2 (RB2) and JudgeBench

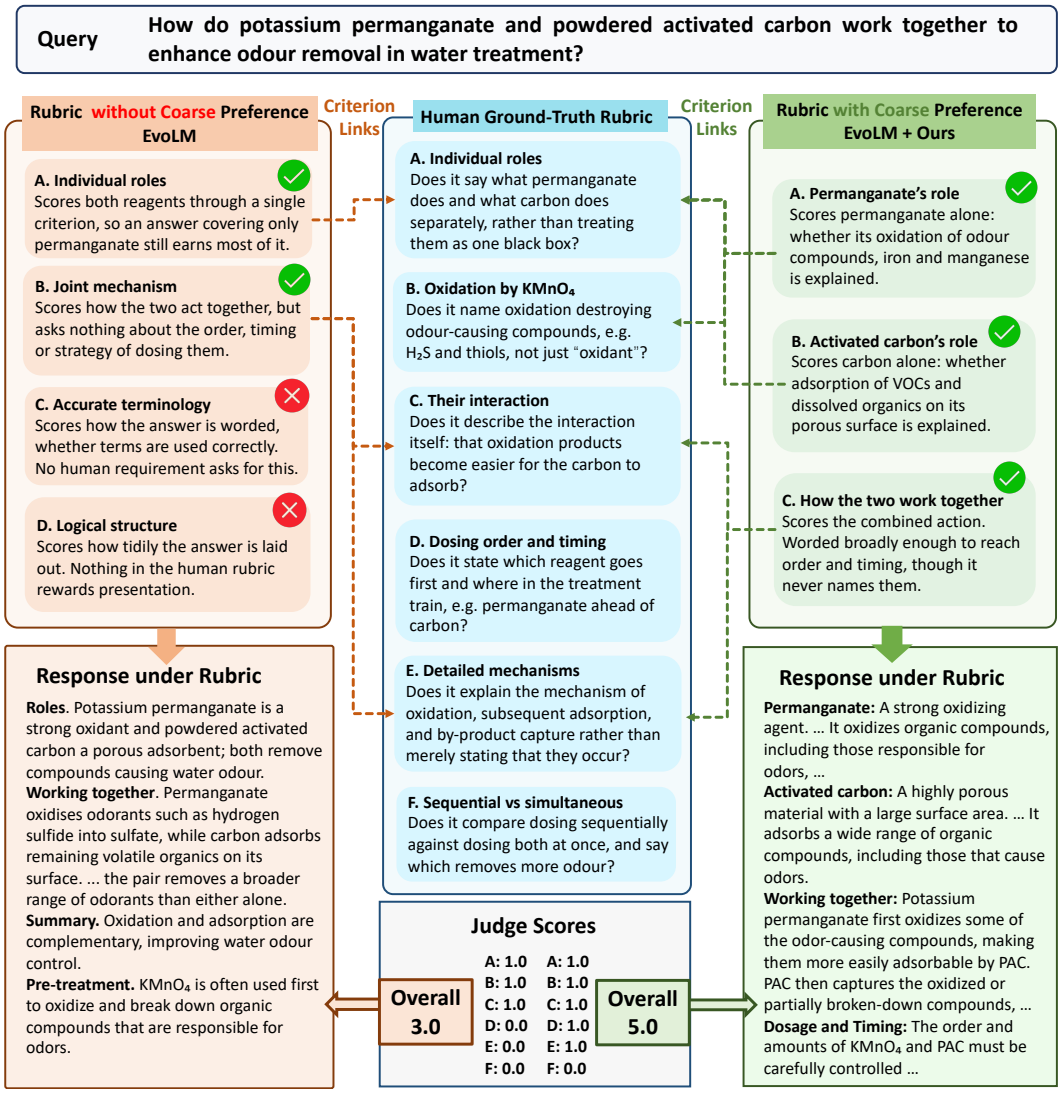
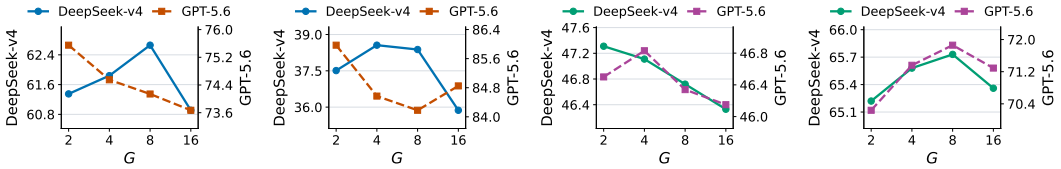


Figure 7: A case study in ResearchQA, comparing the generated rubrics, responses, and scores.



(a) RB2 (b) JB (c) RQA (d) UF

Figure 8: Sensitivity to G: (a,b) rubric-set accuracy; (c,d) policy scores.

(JB), corresponding to RQ1, and downstream policy judge scores on ResearchQA (RQA) and UltraFeedback (UF), corresponding to RQ2. The +Ours variants improve over their paired baselines in most settings across the three backbones. These results show that improvements in both rubric-set discrimination and downstream policy quality persist across the tested model sizes and families.

Table 5: Sensitivity to the backbone of the rubric and policy model. The judge backbone is Qwen3-8B, while the rubric-generator and answer-policy backbones are Qwen3-4B, Qwen3-8B, or Llama3.1-8B. Under each evaluation model, **Rubric** reports discriminative accuracy (%) on RB2 and JB, whereas **Policy** reports judge scores on RQA and UF. Higher is better for all metrics. Each +Ours row is paired with the baseline directly above it; **bold** marks the best value per column within each backbone block.

	DeepSeek-v4				GPT-5.6			
	Rubric (RQ1)		Policy (RQ2)		Rubric (RQ1)		Policy (RQ2)	
Method	RB2	JB	RQA	UF	RB2	JB	RQA	UF
Qwen3-4B								
Rubric-ARM	55.96	36.69	42.66	67.88	67.14	77.72	43.79	70.08
Rubric-ARM+Ours	61.10	**40.51**	45.97	**69.94**	73.25	83.84	44.23	**72.84**
EvoLM	58.73	36.37	44.88	63.46	69.47	78.75	45.97	70.97
EvoLM+Ours	**61.84**	38.56	**47.11**	65.58	**74.55**	**84.57**	**46.83**	71.36
Qwen3-8B								
Rubric-ARM	64.02	46.41	44.42	72.03	83.42	82.29	47.99	74.03
Rubric-ARM+Ours	64.83	**47.36**	46.25	**72.56**	**85.78**	86.02	48.08	**77.07**
EvoLM	70.21	43.68	48.76	68.80	73.80	85.41	53.86	72.92
EvoLM+Ours	**72.13**	46.82	**50.71**	70.12	76.69	**87.15**	**55.20**	75.66
Llama3.1-8B								
Rubric-ARM	60.40	47.70	44.09	78.56	73.66	88.05	50.15	70.92
Rubric-ARM+Ours	65.40	**48.64**	44.76	**80.59**	73.77	**88.29**	**51.75**	71.15
EvoLM	69.46	45.13	52.68	71.65	80.28	84.79	49.86	71.72
EvoLM+Ours	**71.10**	47.58	**54.39**	73.51	**85.58**	84.84	50.57	**73.10**

A.10 Backbone Sensitivity of the Judge Model

We fix the rubric generator and answer policy to Qwen3-4B and vary the judge backbone between Qwen3-8B and Llama3.1-8B. Table 6 separately reports rubric-set discriminative accuracy on RB2 and JB (RQ1) and downstream policy judge scores on RQA and UF (RQ2). The +Ours variants improve over their paired baselines in most settings under both judge backbones. These results indicate that improvements in both rubric-set discrimination and downstream policy quality persist when the judge backbone changes.

A.11 Token-Level Definitions for GRPO

In Equation 2, h indexes response tokens and $|a_n|$ is the number of tokens in response a_n. The two probability ratios are

$$v_{n,h}(\theta) = \frac{\pi_\theta(a_{n,h} \mid q, a_{n,<h})}{\pi_{\theta_{\text{old}}}(a_{n,h} \mid q, a_{n,<h})}, \qquad \kappa_{n,h}(\theta) = \frac{\pi_{\text{ref}}(a_{n,h} \mid q, a_{n,<h})}{\pi_\theta(a_{n,h} \mid q, a_{n,<h})}. \tag{16}$$

Here $\pi_{\theta_{\text{old}}}$ is the sampling policy and π_{ref} is the fixed KL reference. For the rubric generator, $\mathcal{J}_{\text{GRPO}}$ has the same form with π_θ, $\pi_{\theta_{\text{old}}}$, π_{ref}, and G replaced by ρ_ϕ, $\rho_{\phi_{\text{old}}}$, ρ_{ref}, and K. The response sequences are replaced by the tokenized rubric sets; the length normalization uses their token counts, rather than their numbers of criteria.

Table 6: Sensitivity to the judge backbone. The rubric generator and answer policy use Qwen3-4B, while the judge backbone is Qwen3-8B or Llama3.1-8B. Under each evaluation model, **Rubric (RQ1)** reports discriminative accuracy (%) on RB2 and JB, whereas **Policy (RQ2)** reports judge scores on RQA and UF. Higher is better for all metrics. Each +Ours row is paired with the baseline directly above it; **bold** marks the best value per column within each judge-backbone block.

	DeepSeek-v4				GPT-5.6			
	Rubric (RQ1)		Policy (RQ2)		Rubric (RQ1)		Policy (RQ2)	
Method	RB2	JB	RQA	UF	RB2	JB	RQA	UF
Qwen3-8B judge								
Rubric-ARM	55.96	36.69	42.66	67.88	67.14	77.72	43.79	70.08
Rubric-ARM+Ours	61.10	**40.51**	45.97	**69.94**	73.25	83.84	44.23	**72.84**
EvoLM	58.73	36.37	44.88	63.46	69.47	78.75	45.97	70.97
EvoLM+Ours	**61.84**	38.56	**47.11**	65.58	**74.55**	**84.57**	**46.83**	71.36
Llama3.1-8B judge								
Rubric-ARM	56.62	36.33	41.82	69.40	75.51	84.49	43.22	70.71
Rubric-ARM+Ours	61.28	**41.02**	45.70	**76.29**	**80.45**	**85.56**	44.62	75.67
EvoLM	57.15	34.86	45.73	66.38	72.44	79.95	48.42	76.52
EvoLM+Ours	**62.10**	36.21	**47.92**	68.98	75.14	81.60	**52.86**	**77.03**

B Proof of Theorem 4.1

Proof. Condition throughout on the rubric-side information $\mathcal{H}$, which fixes θ_t, the candidate rubric sets and response groups, their GRPO gradients $g_k(q)$, and the baseline choice $g_f \in \{g_0, \dots, g_K\}$, with $g_0 = 0$. For a randomized baseline, also condition on its randomization, independent of the shared component and reference noise. This conditioning makes explicit the fixed policy, candidate pool, and rubric-only choice in the main-text comparison; its conditional expectation is the expectation written in Theorem 4.1. We suppress q where convenient.

Let $\mathcal{S}_q = \operatorname{span}\{g_1, \dots, g_K\}$, let P_q be its orthogonal projector, and set $p_q = \dim \mathcal{S}_q$. Since every candidate update lies in $\mathcal{S}_q$, its inner product with a gradient equals its inner product with that gradient's projection. The main-text vectors $g_\star$ and y are the orthonormal-coordinate representations of $P_q \nabla F(\theta_t)$ and $P_q g_{\mathrm{ref}}$; inner products and norms are unchanged in these coordinates. The local information model, conditional on $\mathcal{H}$, is

$$\begin{aligned} P_q \nabla F(\theta_t) &= d + z, & y &= P_q g_{\mathrm{ref}} = z + \xi, \\ z &\sim \mathcal{N}(0, \tau^2 P_q), & \xi &\sim \mathcal{N}(0, \sigma^2 P_q / M), \end{aligned} \tag{17}$$

where $d \in \mathcal{S}_q$ is fixed, z and ξ are independent, and $\tau, \sigma > 0$. The noise variance σ^2/M results, for example, from averaging M conditionally independent measurements $z + \xi_j$ with $\xi_j \sim \mathcal{N}(0, \sigma^2 P_q)$. All Gaussian calculations below use orthonormal coordinates on $\mathcal{S}_q$, and expectations average over z and ξ.

For each nonzero candidate, let $u_k = g_k / \|g_k\|$, and define $\mathcal{U}_q = \{0\} \cup \{u_k : g_k \neq 0\}$. The quantities in the theorem are

$$\begin{aligned} m_q &= \min_{g_k \neq 0} \|g_k\|, & G_q &= \max_{0 \le k \le K} \|g_k\|, \\ W_q &= \mathbb{E}_{h \sim \mathcal{N}(0, P_q)} \max_{u \in \mathcal{U}_q} \langle h, u \rangle, \\ \Delta_q &= \max_{0 \le k \le K} \langle d, g_k \rangle - \min_{0 \le k \le K} \langle d, g_k \rangle. \end{aligned} \tag{18}$$

Put $s^2 = \tau^2 + \sigma^2/M$ and $a = \tau^2/s^2$. Completing the square in the Gaussian conditional density gives

$$\begin{aligned} p(z \mid y, \mathcal{H}) &\propto \exp\left(-\frac{\|z\|^2}{2\tau^2} - \frac{M\|y - z\|^2}{2\sigma^2} \right), \\ \mathbb{E}[z \mid y, \mathcal{H}] &= ay, \qquad y \mid \mathcal{H} \sim \mathcal{N}(0, s^2 P_q). \end{aligned} \tag{19}$$

For every nonzero candidate,

$$R_k^{\text{align}}(q) = \frac{\langle y, u_k \rangle}{\|g_{\text{ref}}\|}. \tag{20}$$

The denominator is shared across candidates. Define $H(y) = \max_{u \in \mathcal{U}_q} \langle y, u \rangle \geq 0$. When $H(y) > 0$, our rule selects a nonzero gradient $\widehat{g}_q$ whose normalized direction attains this maximum; when $H(y) = 0$, the positive-alignment filter sets $\widehat{g}_q = 0$. With ties resolved by a fixed rule using $\mathcal{H}$ and y, this yields

$$\langle y, \widehat{g}_q \rangle \geq m_q H(y). \tag{21}$$

Since g_f is fixed conditional on $\mathcal{H}$ and $\mathbb{E}[z \mid \mathcal{H}] = 0$, the posterior mean and homogeneity of H give

$$\begin{aligned} \mathbb{E}[\langle z, \widehat{g}_q - g_f \rangle \mid \mathcal{H}] &= a\, \mathbb{E}[\langle y, \widehat{g}_q \rangle \mid \mathcal{H}] \\ &\geq a m_q\, \mathbb{E}[H(y) \mid \mathcal{H}] = a m_q s W_q = A_q(M). \end{aligned} \tag{22}$$

The query-specific contribution satisfies $\langle d, \widehat{g}_q - g_f \rangle \geq -\Delta_q$ pointwise. Combining the two components gives

$$\mathbb{E}[\langle \nabla F(\theta_t), \widehat{g}_q - g_f \rangle \mid \mathcal{H}] \geq A_q(M) - \Delta_q. \tag{23}$$

For each realization, local L-smoothness supplies lower and upper Taylor bounds along the two update segments:

$$\begin{aligned} &F(\theta_t + \eta \widehat{g}_q) - F(\theta_t + \eta g_f) \\ &\geq \eta \langle \nabla F(\theta_t), \widehat{g}_q - g_f \rangle - \frac{L\eta^2}{2} \left(\|\widehat{g}_q\|^2 + \|g_f\|^2 \right) \\ &\geq \eta \langle \nabla F(\theta_t), \widehat{g}_q - g_f \rangle - L\eta^2 G_q^2. \end{aligned} \tag{24}$$

Taking conditional expectations proves Equation 15. In particular, if $A_q(M) > \Delta_q$ and $L > 0$, every step size $0 < \eta < (A_q(M) - \Delta_q)/(LG_q^2)$ within the smooth neighborhood gives a positive expected gain. For $L = 0$, the positive first-order gain holds throughout that neighborhood.

Finally, Gaussian entropy gives the information representation of $A_q(M)$. With natural logarithms,

$$\begin{aligned} I_q &= I(z; y \mid \mathcal{H}) = h(y \mid \mathcal{H}) - h(y \mid z, \mathcal{H}) \\ &= \frac{p_q}{2} \log \left(1 + \frac{M\tau^2}{\sigma^2} \right), \\ A_q(M) &= m_q \tau W_q \sqrt{1 - \exp(-2I_q/p_q)}. \end{aligned} \tag{25}$$

□