EvoRubrics: Dynamic Rubrics as Rewards via Adversarial Co-Evolution for LLM Reinforcement Learning

Anonymous ACL submission

Abstract

Rubric-based rewards offer interpretable and fine-grained optimization signals for reinforcement learning in open-ended tasks where verifiable answers are unavailable. However, pre-constructed rubrics remain static throughout training, creating a fundamental mismatch with the evolving policy: fixed criteria gradually lose discriminative power as the model improves, leading to reward saturation and potential hacking. Recent dynamic rubric methods partially address this but rely on external frontier models or ground-truth answers, and update rubrics only at coarse granularity. We propose EvoRubrics, a co-evolutionary RL framework where a Policy LLM and a Rubric Generator jointly improve through adversarial interaction within each training step. As the policy improves under the rubric generator's guidance, the rubric generator adapts its criteria to remain discriminative and informative, enabling evaluation to track the policy in real time and naturally inducing an automatic curriculum. Experiments show that EvoRubrics consistently outperforms static and dynamic rubric baselines across benchmarks. The learned Rubric Generator further generalizes as a transferable reward model. Notably, even a fully self-supervised variant without any external supervision achieves meaningful gains, suggesting that co-evolution between generation and evaluation alone can provide sufficiently rich learning signals. Our code is publicly available at https://anonymous.4open.science/r/EvoRubrics-2155/.

1 Introduction

Reinforcement learning (RL) (Schulman et al., 2017; Shao et al., 2024; Rafailov et al., 2023) has become a central paradigm for aligning and improving Large Language Models (LLMs) (Achiam et al., 2023; Yang et al., 2025), yet its success hinges on reliable reward signals, such as human preferences or verifiable ground-truth answers. In many high-value applications, including open-ended question answering, creative writing, and medical consultation, such signals are inherently unavailable: outputs are judged not by exact correctness, but by nuanced, multidimensional qualities that are difficult to formalize as scalar rewards.

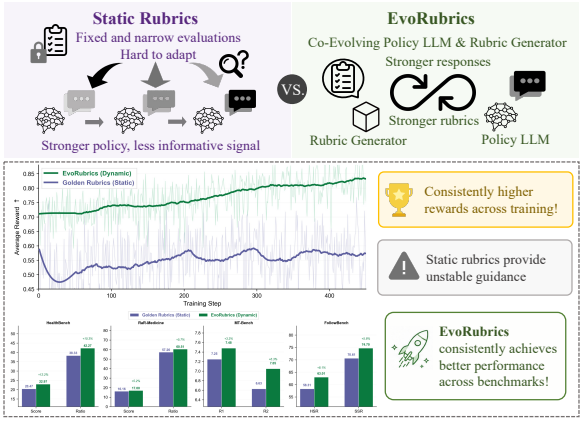

Figure 1: **Static rubrics vs. EvoRubrics.** Static rubrics provide non-adaptive evaluations and unstable rewards; EvoRubrics co-evolves a Rubric Generator and Policy LLM, yielding stable rewards and consistent gains.

Rubrics as rewards (Gunjal et al., 2025; Liu et al., 2025b; Arora et al., 2025) offer a promising alternative, where each query is associated with a structured set of evaluation criteria specifying desirable or undesirable properties, such as factual support or reward-hacking patterns. By factorizing evaluation into interpretable dimensions, rubrics offer more transparent and fine-grained optimization signals. However, high-quality rubric construction typically requires domain experts or expensive frontier models, limiting scalability and coverage.

Beyond this practical bottleneck lies a more fundamental limitation: pre-constructed rubrics remain **static**, mismatched with the RL training dynamics. Early in training, when the policy remains underdeveloped, rubrics may be excessively stringent, assigning uniformly low scores and failing to provide sufficiently discriminative learning signals. As the policy improves, the same rubrics

become progressively saturated: increasingly many responses satisfy the prescribed criteria, diminishing the reward's capacity to resolve subtle quality differences. This phenomenon not only stalls learning but also opens the door to reward hacking, where the policy learns superficial shortcuts that satisfy the criteria without genuine quality improvement (Figure 1). Ideally, evaluations should adapt with the policy, becoming progressively more precise and challenging as the model improves.

Recent works have explored dynamic rubric mechanisms to address this limitation. One line of work dynamically elicits or revises rubrics using external frontier LLMs (Rezaei et al., 2025; Shen et al., 2026; Xu et al., 2026b), improving adaptivity but introducing substantial dependence on powerful models, also limiting their applicability to resource-constrained or new domains. Moreover, their rubric updates often operate at epoch-level temporal granularity, unable to track the policy's rapidly evolving capabilities. Another line trains a rubric generator directly, but does not explicitly couple rubric adaptation with policy improvement (Xu et al., 2026a) or only treats rubrics as a complement to verifiable ground-truth answers and restricting applicability to objectively verifiable domains (Sheng et al., 2026). These limitations point to a central, unresolved research question: *how can rubric generators be learned to co-evolve with the policy at training-step granularity, so that they provide discriminative and informative signals throughout RL for open-ended generation?*

To fill this gap, we propose EvoRubrics, a co-evolutionary RL framework where a Policy LLM and a Rubric Generator iteratively improve through adversarial interaction within each training step. The Policy LLM learns to produce better responses under rubric-aggregated rewards, while the Rubric Generator receives multi-objective rewards encouraging discriminative power, semantic diversity, alignment with human preferences, and constructiveness, continuously adapting its evaluation criteria to remain discriminative and informative. This within-step co-evolution enables the evaluation standards to track the policy's capabilities in real time, naturally inducing an auto-curriculum: as the policy improves, the generator produces increasingly fine-grained criteria that expose remaining gaps, while the strengthening rubrics in turn raise the bar for the policy. Beyond training, the evolved Rubric Generator serves as a transferable tool for test-time rubric generation on unseen queries. Notably, we find that even a fully self-supervised variant, trained without any external supervision, yields meaningful performance gains, demonstrating that the adversarial interplay between generation and evaluation alone provides a sufficiently rich optimization signal.

Our contributions are summarized as follows:

- **Insightfully,** we identify that rubrics should co-evolve with the policy in real time to remain effective, and show that the adversarial dynamics between a policy and its evaluator can serve as a self-contained source of informative learning signal, even without external supervision.
- **Technically,** we propose EvoRubrics, where a Policy LLM and a Rubric Generator share a single base model via dual LoRA adapters and are jointly optimized within each training step through carefully designed rewards.
- **Empirically,** EvoRubrics consistently outperforms static and dynamic rubric baselines on both in-domain and out-of-distribution benchmarks. The trained Rubric Generator generalizes beyond training, enabling effective test-time rubric generation for unseen datasets.

2 Related Work

2.1 Static Rubric-based Rewards

RL has become a standard approach for improving LLMs (Shao et al., 2024). For open-ended tasks without verifiable answers, recent work uses *rubrics as rewards*, replacing opaque scalar judgments with structured and interpretable evaluation criteria. They study rubric construction for reward modeling (Liu et al., 2025b), and apply rubric-based rewards to policy optimization (Gunjal et al., 2025; He et al., 2025; Zhou et al., 2025). Although effective, these methods rely on pre-constructed rubrics that remain fixed throughout training, limiting their ability to provide informative rewards as the policy evolves.

2.2 Dynamic Rubrics and Adaptive Evaluation

Several recent works attempt to make rubrics adaptive. Some works (Rezaei et al., 2025; Xu et al., 2026b; Shen et al., 2026) update rubrics during training, but typically depend on external frontier models and often operate at coarse temporal granularity such as epoch-level. Other methods learn rubric generators directly (Xu et al., 2026a; Sheng et al., 2026), but rely on human preference annota-

tions or verifiable ground-truth answers. As a result, existing approaches only partially address the limitations of static rubrics: rubric adaptation remains externally induced, weakly coupled to policy improvement, or restricted to settings with stronger supervision. In contrast, EvoRubrics enables a Rubric Generator to co-evolve with the Policy LLM within each training step, providing adaptive evaluation for open-ended RL.

3 Preliminary: Rubrics as Rewards

Rubric structure. For a query q, a rubric set $R = \{(d_k, w_k)\}_{k=1}^{K}$ contains K evaluation criteria. Each criterion consists of a natural-language description d_k specifying a particular quality dimension (e.g., factual accuracy, coherence, or reward-hacking patterns), and a weight $w_k \in \mathbb{R}$ indicating the score assigned when the criterion is satisfied. Positive weights reward desirable properties, while negative weights penalize undesirable ones.

Scoring mechanism. Given a response a and a rubric set R, a judge model $\mathcal{J}$ evaluates the response against each criterion independently, producing a binary decision indicating whether the criterion is met. The total score of a is the sum of weights for all satisfied criteria:

$$s(a, R) = \sum_{k=1}^{K} \mathbb{1}[\mathcal{J}(a, d_k) = \texttt{met}] \cdot w_k \quad (1)$$

To enable comparison across different rubric sets, we normalize by the maximum achievable positive score $W^+ = \sum_{k:\, w_k>0} w_k$:

$$S(a, R) = \frac{s(a, R)}{W^+} \quad (2)$$

Rubrics as reward signals. The normalized score $S(a, R)$ can serve not only as an evaluation metric but also as a reward signal for RL, providing transparent and fine-grained optimization guidance. However, a key limitation is that effective rubrics are difficult to construct and, once pre-defined, often remain static throughout training, failing to remain discriminative as the policy evolves, which motivates our co-evolving rubric RL framework.

Problem setup. We consider open-ended generation, where each training instance consists of a query q without a verifiable ground-truth answer. Our goal is to jointly learn (i) a Policy LLM π_θ that generates high-quality responses, and (ii) a Rubric Generator π_ψ that produces query-specific rubric sets for evaluating such responses. Given a query q, the policy samples answers $a \sim \pi_\theta(\cdot \mid q)$, and the rubric generator samples rubric sets $R \sim \pi_\psi(\cdot \mid q)$. The central challenge is to optimize π_θ and π_ψ jointly so that the rubrics remain informative as the policy evolves.

4 Methodology

We propose EvoRubrics, a co-evolutionary RL framework for open-ended tasks, where a *Policy LLM* and a *Rubric Generator* are instantiated from a shared backbone via two LoRA adapters and jointly optimized with GRPO (Shao et al., 2024), as shown in Figure 2. We describe the dual-LoRA architecture (§4.1), policy optimization (§4.2), the Rubric Generator reward (§4.3), and the co-evolutionary training procedure (§4.4).

4.1 Dual-LoRA Architecture

EvoRubrics instantiates the Policy LLM and the Rubric Generator from a shared backbone LLM using two independent LoRA adapters (Hu et al., 2022), parameterized by θ and ψ, respectively. The underlying backbone model without either adapter activated, serves as the reference model π_{ref} for regularization during training.

This shared-backbone design offers both effectiveness and efficiency. Since the two roles operate over the same base model, they inherit a common knowledge base and representation space, placing them at a comparable capability level. The Rubric Generator can more accurately assess the policy's outputs and track its evolving capability frontier. Meanwhile, the separate LoRA adapters allow role-specific specialization for generation and evaluation without requiring two full models, substantially reducing memory and compute costs. Each adapter is trained with its own optimizer and scheduler, and only the active adapter receives gradients during its update phase.

4.2 Policy LLM Optimization

Given a query q, the Policy LLM π_θ generates M candidate answers $\{a_i\}_{i=1}^{M}$, and the Rubric Generator π_ψ produces N rubric sets $\{R_j\}_{j=1}^{N}$. Using the rubric-based scoring defined in §3, evaluating all answer-rubric pairs yields an $M \times N$ score matrix $\mathbf{S}$, where $S_{i,j} = S(a_i, R_j)$.

Policy reward. The reward for each candidate answer a_i is its average normalized score across all

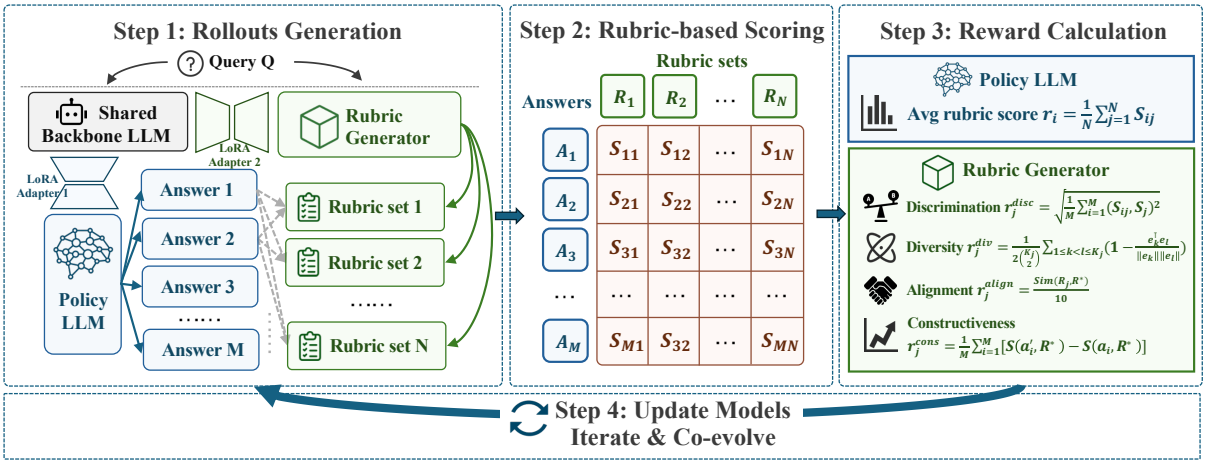

Figure 2: **EvoRubrics framework overview.** At each step, the Policy LLM and Rubric Generator generate M candidate answers and N rubric sets via dual LoRA adapters; a judge model scores all answer-rubric pairs to form an $M \times N$ matrix; policy and rubric rewards are computed from this matrix; and both adapters are updated with GRPO, enabling real-time co-evolution.

N rubric sets:

$$r_i^{\text{pol}} = \frac{1}{N} \sum_{j=1}^{N} S_{i,j} \tag{3}$$

Aggregating over multiple independently generated rubric sets provides a robust reward signal that mitigates the noise of any single evaluation rubric.

Policy optimization. We optimize the Policy LLM using GRPO, which computes advantages by normalizing rewards within the group of M co-generated responses:

$$\hat{A}_i = \frac{r_i^{\text{pol}} - \mu^{\text{pol}}}{\sigma^{\text{pol}} + \epsilon} \tag{4}$$

where μ^{pol} and σ^{pol} are the mean and standard deviation of $\{r_i^{\text{pol}}\}_{i=1}^{M}$, and ϵ is a constant for numerical stability. Adapter weights θ are updated by minimizing the clipped surrogate objective with KL regularization:

$$\mathcal{L}(\theta) = -\mathbb{E}_t\Big[\min\big(\rho_t \hat{A}_t,\ \text{clip}(\rho_t, 1-\epsilon_c, 1+\epsilon_c)\,\hat{A}_t\big)\Big] + \beta\, D_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}] \tag{5}$$

where $\rho_t = \pi_\theta(a_t \mid s_t) \,/\, \pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the importance sampling ratio, ϵ_c is the clipping range, and β controls the strength of the KL penalty against the reference model π_{ref}.

4.3 Rubric Generator Optimization

The Rubric Generator is trained as an adaptive evaluator that continually challenges the policy during co-evolution. Ideal rubrics should satisfy four properties: *discriminativeness*, to distinguish responses of different quality and provide effective optimization signals; *diversity*, to cover complementary evaluation dimensions; *alignment*, to keep the learned evaluation signal grounded in desirable directions; and *constructiveness*, to provide actionable guidance for response improvement.

These properties sustain productive co-evolution. As the policy improves, stronger rubrics impose finer-grained and more demanding evaluation criteria, creating implicit adversarial pressure on the policy. Alignment and constructiveness, in turn, prevent this pressure from drifting into reward hacking or misaligned evaluation. Accordingly, we optimize the Rubric Generator with a multi-objective reward over these four properties.

Discrimination reward. An effective rubric set should differentiate answers of varying quality and expose the policy's remaining weaknesses. For each rubric set R_j, we measure the discriminative power by the standard deviation of its scores assigned to the M candidate answers:

$$r_j^{\text{disc}} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \big(S_{i,j} - \bar{S}_j\big)^2} \tag{6}$$

where $\bar{S}_j = \frac{1}{M} \sum_{i=1}^{M} S_{i,j}$ is the column mean of the scoring matrix. A higher value indicates that the rubric clearly separates strong responses from weak ones; a near-zero value signals that the rubrics cannot distinguish response quality and thus can not provide useful optimization signals.

Diversity reward. The individual criteria within a rubric set should cover distinct evaluation dimensions rather than redundantly measuring the same aspect, so that the induced reward signal remains multi-faceted. We encode each criterion description $d_k^{(j)}$ into an embedding $\mathbf{e}_k$ using a pretrained sentence transformer (Reimers and Gurevych, 2019) and compute the average pairwise cosine distance:

$$r_j^{\text{div}} = \frac{1}{2\binom{K_j}{2}} \sum_{1 \le k < l \le K_j} \left(1 - \frac{\mathbf{e}_k^\top \mathbf{e}_l}{\|\mathbf{e}_k\| \, \|\mathbf{e}_l\|}\right) \quad (7)$$

The factor of $\frac{1}{2}$ normalizes the reward to $[0, 1]$. This encourages the generator to produce criteria spanning orthogonal evaluation aspects.

Alignment reward. The generated rubrics should be aligned with human preferences, preventing the generator from drifting toward arbitrary or overly idiosyncratic criteria that may be useful for optimization but inconsistent with desired task objectives. We anchor the generated rubrics to pre-constructed rubrics R^*. A judge model scores the semantic similarity between R_j and R^* on a 0–10 scale, which is then normalized:

$$r_j^{\text{align}} = \frac{\text{Sim}_{\mathcal{J}}(R_j, R^*)}{10} \quad (8)$$

This reward acts as a regularizer, ensuring that the evolving rubrics remain grounded in human-defined quality standards.

Constructiveness reward. Beyond evaluation, effective rubrics should provide actionable guidance for answer improvement. We assess this by using the generated rubric R_j to prompt the Policy LLM to reflect on and revise each baseline answer a_i, producing a refined answer a_i'. The reward measures the average score improvement on the pre-defined golden rubrics R^*:

$$r_j^{\text{cons}} = \frac{1}{M} \sum_{i=1}^{M} \left[S(a_i', R^*) - S(a_i, R^*)\right] \quad (9)$$

A larger improvement indicates that the rubric provides more constructive and actionable feedback.

Total reward. The overall reward for rubric set R_j is a weighted sum of the four components:

$$r_j^{\text{rub}} = \lambda_{\text{disc}} \, r_j^{\text{disc}} + \lambda_{\text{div}} \, r_j^{\text{div}} + \lambda_{\text{align}} \, r_j^{\text{align}} + \lambda_{\text{cons}} \, r_j^{\text{cons}}. \quad (10)$$

where λ_{disc}, λ_{div}, λ_{align}, and λ_{cons} are hyperparameters controlling the relative importance of each objective.

Fully self-supervised variant. The reward formulation above assumes access to a pre-constructed ground-truth rubric set R^* as a human-preference anchor, used in Eq. 8. For datasets where such rubric annotations are unavailable, we additionally consider a fully self-supervised variant. In this setting, we retain the discrimination reward r_j^{disc} and the diversity reward r_j^{div}, remove the alignment reward, and instantiate constructiveness purely in a self-supervised manner through the generated rubric itself. Specifically, for R_j, we prompt the Policy LLM to revise each baseline answer a_i according to R_j, yielding a refined answer a_i', and measure the average improvement under the same rubric:

$$r_j^{\text{self-refl}} = \frac{1}{M} \sum_{i=1}^{M} \left[S(a_i', R_j) - S(a_i, R_j)\right]. \quad (11)$$

The fully self-supervised reward is then defined as

$$r_j^{\text{rub,self}} = \lambda_{\text{disc}} \, r_j^{\text{disc}} + \lambda_{\text{div}} \, r_j^{\text{div}} + \lambda_{\text{self-refl}} \, r_j^{\text{self-refl}}. \quad (12)$$

This variant removes the dependence on human-authored rubrics while preserving discriminativeness, diversity, and actionable feedback.

Rubric generator optimization. The Rubric Generator is optimized with the same GRPO procedure. For a given query, the group-relative advantage for rubric set R_j is:

$$\hat{A}_j = \frac{r_j^{\text{rub}} - \mu^{\text{rub}}}{\sigma^{\text{rub}} + \epsilon} \quad (13)$$

where μ^{rub} and σ^{rub} are the mean and standard deviation of $\{r_j^{\text{rub}}\}_{j=1}^{N}$. The adapter weights ψ are updated by minimizing:

$$\mathcal{L}(\psi) = -\mathbb{E}_t\Big[\min\big(\rho_t' \, \hat{A}_t, \; \text{clip}(\rho_t', 1-\epsilon_c, 1+\epsilon_c) \, \hat{A}_t\big)\Big] + \beta \, D_{\text{KL}}[\pi_\psi \,\|\, \pi_{\text{ref}}] \quad (14)$$

where $\rho_t' = \pi_\psi(a_t \mid s_t) \,/\, \pi_{\psi_{\text{old}}}(a_t \mid s_t)$.

4.4 Co-Evolutionary Training

Training procedure. At each training step, both models are updated sequentially on the same query batch through four phases (The algorithm is provided in Appendix A). In the *rollout* phase, the Policy LLM generates M candidate answers and

the Rubric Generator produces N rubric sets, with adapter switching between the two roles. In the *cross-evaluation* phase, the judge model scores every answer–rubric pair to construct the $M \times N$ score matrix $\mathbf{S}$. In the *reward computation* phase, policy rewards and rubric rewards are computed using Eq. 3 and Eq. 10. In the *model update* phase, the Policy adapter θ and the Rubrics adapter ψ are each updated via GRPO.

This within-step alternation ensures that both models train against each other's most recent behavior, enabling real-time co-adaptation rather than the epoch-granularity updates of prior approaches. As the policy improves, the discrimination reward drives the generator to produce increasingly fine-grained criteria that expose remaining gaps, while the strengthening rubrics in turn raise the bar for the policy. This adversarial tension creates a natural auto-curriculum over evaluation standards.

5 Experiments

5.1 Experimental Setup

Models and training. We instantiate the Policy LLM and Rubric Generator from a shared backbone with separate LoRA adapters, using `Qwen3-4B` and `Qwen3-8B` (Yang et al., 2025). Before co-evolutionary RL, we warm-start the LLMs with supervised fine-tuning on responses sampled from DeepSeek-R1 (Guo et al., 2025), equipping them with domain instruction-following capabilities that facilitate subsequent specializations. DeepSeek-V3.2 (Liu et al., 2025a) is used as the judge throughout training and evaluation. Detailed training settings and complexity analysis are provided in Appendices B and E. Prompts used in training and evaluation are provided in Appendix F.

Data. We use **HealthBench** (Arora et al., 2025) as the training dataset, following the official split with 4,000 standard-difficulty examples for training and 1,000 Hard cases for testing.

Evaluation. We evaluate the Policy LLM on diverse benchmarks. In-domain benchmarks include **HealthBench Hard** and unseen **RaR-Medicine** (Gunjal et al., 2025); the OOD benchmarks include **MT-Bench** (Zheng et al., 2023) and **FollowBench** (Jiang et al., 2024). On rubric-based benchmarks, we report Score and Ratio; on MT-Bench, we report Round-1 and Round-2 scores; and on FollowBench, we report Hard Satisfaction Rate (HSR) and Soft Satisfaction Rate (SSR).

We further evaluate the trained Rubric Generator in two transfer settings: (1) **Rubric as Reward Model**, where generated rubrics are used for preference ranking on **RubricBench** (Zhang et al., 2026); and (2) **Rubric as Guidance**, where generated rubrics are prepended to the prompt to guide test-time generation. More details of the evaluation protocols are provided in Appendix C.

Baselines. We compare EvoRubrics against both static and dynamic rubric-based RL baselines, including **GoldenRubrics**, **RuscaRL** (Zhou et al., 2025), and **OnlineRubrics** (Rezaei et al., 2025). All RL methods are initialized from the same SFT checkpoint for fair comparison. Additional implementation details are provided in Appendix D.

5.2 Main Results

Experiment results are reported in Tables 1 and 2. We summarize the main findings below.

EvoRubrics consistently improves policy performance across in-domain and OOD evaluations. Across backbones, EvoRubrics achieves the strongest overall performance among all rubric-based RL methods. The gains are especially pronounced on the in-domain HealthBench Hard benchmark, where EvoRubrics substantially outperforms using golden rubrics and strong dynamic rubric baselines. Notably, EvoRubrics with small backbones even surpasses substantially larger models like DeepSeek-V4-Flash on HealthBench Hard, showing that co-evolving rubric supervision can unlock domain-specific capability gains beyond what model scale alone provides.

Co-evolving rubrics yield more robust training signals than static or externally induced rubrics. A key advantage of EvoRubrics is that its gains do not come at the cost of OOD generalization. Unlike prior rubric-based RL methods, which often degrade on MT-Bench and FollowBench, EvoRubrics largely preserves, and sometimes improves, OOD performance relative to SFT and the base model. This suggests that dynamically co-evolved rubrics provide more transferable optimization signals, whereas fixed rubrics or externally accumulated criteria are more prone to overfitting to the training domain.

The learned Rubric Generator transfers beyond training-time reward construction. As shown in Table 2, the Rubric Generator trained by EvoRubrics generalizes well to downstream transfer settings. As a standalone reward model on RubricBench, it consistently improves pairwise judging accuracy over Vanilla and SFT baselines,

Model	Category	Approach	HealthBench *In-domain (Seen)* Score ↑	HealthBench *In-domain (Seen)* Ratio ↑	RaR-Medicine *In-domain (Unseen)* Score ↑	RaR-Medicine *In-domain (Unseen)* Ratio ↑	MT-Bench *OOD* R1 ↑	MT-Bench *OOD* R2 ↑	FollowBench *OOD* HSR ↑	FollowBench *OOD* SSR ↑
Deepseek-V4-Flash	*Vanilla*	*Prompt*	*18.04*	*33.03*	*20.22*	*75.00*	*8.64*	*8.06*	*78.06*	*84.62*
Qwen3-32B			*17.17*	*32.07*	*20.05*	*74.69*	*8.58*	*8.06*	*64.66*	*77.12*
Qwen3-4B	Vanilla	Prompt	12.62	22.48	15.14	56.04	**7.82**	6.82	47.76	64.09
	Supervised	SFT	13.04	23.75	12.35	45.67	7.50	6.44	38.76	56.31
	RL w/ Rubric	GoldenRubrics	15.82	27.75	11.41	42.16	6.04	4.01	32.50	50.40
		OnlineRubrics	5.92	8.04	10.48	38.66	6.40	4.33	23.66	41.47
		RuscaRL	8.85	18.89	10.98	40.94	7.64	6.71	31.89	52.10
	Ours	*EvoRubrics*	**20.80**	**36.64**	**16.36**	**60.50**	7.74	**7.11**	**53.92**	**66.20**
Qwen3-8B	Vanilla	Prompt	14.40	25.92	17.00	60.51	**8.04**	7.04	23.89	36.49
	Supervised	SFT	18.52	34.27	15.94	56.50	7.96	**7.43**	59.33	72.40
	RL w/ Rubric	GoldenRubrics	20.47	38.33	16.16	57.26	7.25	6.63	58.31	70.61
		OnlineRubrics	20.23	37.74	16.32	57.85	7.60	6.71	60.73	**74.90**
		RuscaRL	8.24	14.23	12.67	44.82	4.11	3.40	39.34	61.59
	Ours	*EvoRubrics*	**22.97**	**42.27**	**17.34**	**61.71**	7.48	7.05	**63.01**	74.70

Table 1: Evaluations of the Policy LLM. **Bold** indicates the best performance, underline the second-best.

Model	Method	Rubric as Reward Model *RubricBench — Accuracy* ↑ IF	STEM	CODE	SAFE	CHAT	Overall	Rubric as Guidance *Ratio* ↑ HealthBench
Deepseek-V4-Flash	*Prompt*	*57.26*	*58.40*	*54.24*	*31.25*	*51.90*	*53.01*	*37.58*
Qwen3-32B	*Prompt*	*47.58*	*52.00*	*55.72*	*40.00*	*51.90*	*51.53*	*31.40*
Qwen3-4B	Vanilla	54.00	54.00	50.60	28.80	47.20	48.90	26.33
	SFT	45.20	50.00	59.00	**57.50**	46.00	50.70	26.10
	EvoRubrics	**59.70**	**49.20**	**59.40**	35.00	**49.50**	**51.90**	**27.87**
Qwen3-8B	Vanilla	44.35	58.00	54.61	37.50	46.21	49.96	43.56
	SFT	46.77	57.60	**56.46**	40.00	46.45	50.83	41.92
	EvoRubrics	**50.00**	**60.80**	56.09	**41.25**	**47.87**	**52.40**	**44.83**

Table 2: Evaluations of the Rubric Generator. **Left**: accuracy as a reward model on RubricBench. **Right**: response quality when generated rubrics are used as inference-time guidance. Best results are in **bold** and second-bests are underlined.

Setting	HB *Ratio*↑	RaR *Ratio*↑	MT *R1*↑	Follow *HSR*↑
GoldenRubrics	27.75	42.16	6.04	32.50
EvoRubrics	**36.64**	**60.50**	**7.74**	**53.92**
w/o r_{align}	34.45	50.26	7.46	47.31
w/o r_{cons}	31.86	46.64	7.46	45.15
w/o r_{div}	31.75	47.58	7.67	47.35
w/o r_{disc}	32.63	47.75	7.54	45.74
w/o r_{align}, r_{cons}, r_{div}, r_{disc}	29.14	51.39	7.49	49.88

Table 3: Ablation study on the Rubric Generator reward design. Each row removes one or more reward components during training and reports the resulting policy performance.

and with the 8B backbone even surpasses the much larger Qwen3-32B. This suggests that the learned evaluator captures transferable preference signals beyond the training data.

The evolved rubrics are effective at inference time. When used as test-time guidance, the rubrics generated by EvoRubrics consistently improve response quality over the corresponding base models, showing that the Rubric Generator is not only a training-time component but also a reusable module for inference. We provide qualitative case studies in Appendix G.

5.3 Ablation Studies

Table 3 presents ablation results on **Qwen3-4B**. **All reward components are important, with discriminativeness serving as the core signal.** Removing any single reward term degrades performance, showing that discriminativeness, diversity, alignment, and constructiveness all contribute to effective rubric learning. Constructiveness and diversity have the largest impact on in-domain performance, highlighting the value of actionable feedback and broad evaluation coverage, while ablating r_{disc} consistently hurts both in-domain and OOD results, confirming that the ability to distinguish responses of different quality is fundamental to

	HealthBench		RaR-Medicine		MT-Bench		FollowBench	
	Score ↑	Ratio ↑	Score ↑	Ratio ↑	R1 ↑	R2 ↑	HSR ↑	SSR ↑
Base Model	12.62	22.48	15.14	56.04	**7.82**	6.82	47.76	64.09
GoldenRubrics	15.82	27.75	11.41	42.16	6.04	4.01	32.50	50.40
EvoRubrics (w/ Golden)	**20.80**	**36.64**	**16.36**	**60.50**	7.74	**7.11**	**53.92**	**66.20**
EvoRubrics (Self-Supervised)	18.85	34.00	11.32	41.70	6.55	4.84	34.47	51.28

Table 4: Policy LLM performance under self-supervised co-evolution. Best results are in **bold** and second-bests are underlined.

Setting	Rubric as Reward Model *RubricBench — Accuracy* ↑						Rubric as Guidance *Ratio* ↑
	IF	STEM	Code	Safety	Chat	Overall	HealthBench
Base Model (Direct)	54.00	**54.00**	50.60	28.80	47.20	48.90	26.33
EvoRubrics (w/ Golden)	**59.70**	49.20	**59.40**	35.00	**49.50**	**51.90**	**27.87**
EvoRubrics (Self-Supervised)	45.16	53.20	53.87	**53.75**	48.58	50.96	24.69

Table 5: Rubric Generator performance under self-supervised co-evolution. Best results are in **bold** and second-bests are underlined.

maintaining an informative training signal.

Co-evolution is necessary beyond rubric initialization alone. Freezing the Rubric Generator by removing all reward components yields better results than GoldenRubrics, but still remains below the full model. This shows that the benefit of EvoRubrics comes from continuously improving the evaluator during training.

Shared-backbone rubrics are more compatible than fixed external rubrics. Interestingly, even without further optimization, the frozen Rubric Generator outperforms GoldenRubrics, suggesting that rubric signals generated from the same backbone are better aligned with the policy's representation space than externally authored fixed rubrics. This compatibility likely makes the resulting reward signal easier to optimize against and less prone to distribution mismatch.

5.4 Fully Self-Supervised Co-Evolution

To investigate whether co-evolution remains effective without any external rubric supervision, we consider a *fully self-supervised* variant of EvoRubrics, as described in §4.3. We evaluate on **Qwen3-4B** and report both Policy LLM and Rubric Generator performance in Tables 4 and 5.

Co-evolution remains effective even without any external rubric supervision. The fully self-supervised variant still yields clear gains on HealthBench, outperforming both the base model and the GoldenRubrics baseline, although falling short of the full model with reference anchors. This shows that the adversarial interaction between policy optimization and learned evaluation alone can provide a sufficiently rich optimization signal.

Self-supervised rubric learning induces meaningful but less balanced evaluator transfer. The fully self-supervised Rubric Generator still improves overall reward-model accuracy on RubricBench over the base model, indicating that some transferable evaluation ability emerges even without human preference guidance. Notably, it performs particularly well on the Safety subset, surpassing both compared models. This suggests that the self-supervised rubric evolution drifts toward the safety-focused aspects emphasized in the medical domain training data. This pattern indicates that self-supervised co-evolution can produce a useful evaluator, though one that is less balanced than its reference-anchored counterpart.

6 Conclusions and Future Work

We presented EvoRubrics, a co-evolutionary RL framework where a Policy LLM and a Rubric Generator improve jointly through real-time interaction. By continuously adapting evaluation criteria to the policy's evolving capability, EvoRubrics induces an automatic curriculum over open-ended generation. Results on both in-domain and OOD benchmarks validate its effectiveness. Moreover, a fully self-supervised variant trained without any external supervision still delivers meaningful gains, indicating that the co-evolution between generation and evaluation alone can provide strong optimization signals. Future directions include extending EvoRubrics to multi-domain settings, examining scaling with larger backbone models, and integrating rubric-based feedback into complex tasks.

Limitations

While EvoRubrics provides a promising framework for co-evolving rubrics and policies in open-ended tasks, several limitations remain. First, although our OOD results on MT-Bench and FollowBench demonstrate encouraging generalization, training is primarily conducted in the medical domain. Evaluating the framework on more diverse domains, such as legal reasoning, creative writing, and scientific QA, would provide a stronger test of the transferability of the co-evolutionary dynamics. Second, due to computational constraints, our experiments are limited to models up to 8B parameters. While these backbones are representative, they may not fully reveal the performance ceiling achievable with larger LLMs. In addition, whether the dual-LoRA co-evolutionary design extends effectively to other architectures, such as mixture-of-experts models, remains an open question.

A potential risk is that co-evolving generation and evaluation may amplify shared biases or spurious preferences, especially when both components are trained from the same backbone. Without careful monitoring, the evaluator may drift toward overly narrow or domain-specific criteria, or the policy may learn to exploit idiosyncrasies of the learned reward. Studying such failure modes and developing more robust safeguards will be important in future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, and 1 others. 2025. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, and 1 others. 2025. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing llm instruction following. *arXiv preprint arXiv:2511.10507*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025a. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. 2025b. Openrubrics: Towards scalable synthetic rubric generation for reward modeling and llm alignment. *arXiv preprint arXiv:2510.07743*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

MohammadHossein Rezaei, Robert Vacareanu, Zihao Wang, Clinton Wang, Bing Liu, Yunzhong He, and Afra Feyza Akyürek. 2025. Online rubrics elicitation from pairwise comparisons. *arXiv preprint arXiv:2510.07284*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

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

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tao Zhao, and Haoyu Wang. 2026a. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable llm post-training. *arXiv preprint arXiv:2602.01511*.

Yifei Xu, Guilherme Potje, Shivam Shandilya, Tiancheng Yuan, Leonardo de Oliveira Nunes, Rakshanda Agarwal, Saeid Asgari, Adam Atkinson, Emre Kıcıman, Songwu Lu, and 1 others. 2026b. Sibylsense: Adaptive rubric learning via memory tuning and adversarial probing. *arXiv preprint arXiv:2602.20751*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qiyuan Zhang, Junyi Zhou, Yufei Wang, Fuyuan Lyu, Yidong Ming, Can Xu, Qingfeng Sun, Kai Zheng, Peng Kang, Xue Liu, and 1 others. 2026. Rubricbench: Aligning model-generated rubrics with human standards. *arXiv preprint arXiv:2603.01562*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, and 1 others. 2025. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*.

A EvoRubrics Algorithm

We provide the pseudo codes for EvoRubrics algorithm as below.

Algorithm 1 EVORUBRICS: Co-Evolutionary Training

Require: Base LLM π, judge model $\mathcal{J}$, reward weights $\boldsymbol{\lambda}$, dataset $\mathcal{D}$

Initialize LoRA adapters θ (policy) and ψ (rubrics) on π
Set reference model $\pi_{\text{ref}} \leftarrow \pi$ ▷ base without LoRA
for each training step **do**
  Sample query batch $\{q\}$ from $\mathcal{D}$
  // Rollout generation
  Activate θ; sample $\{a_i\}_{i=1}^{M} \sim \pi_\theta(\cdot \mid q)$
  Activate ψ; sample $\{R_j\}_{j=1}^{N} \sim \pi_\psi(\cdot \mid q)$
  // Cross-evaluation
  Compute $\mathbf{S} \in \mathbb{R}^{M \times N}$ via $\mathcal{J}$ ▷ Eq. 2
  // Reward computation
  $r_i^{\text{pol}} \leftarrow \frac{1}{N} \sum_j S_{i,j}, \quad \forall\, i$ ▷ Eq. 3
  $r_j^{\text{rub}} \leftarrow \lambda_{\text{disc}}\, r_j^{\text{disc}} + \lambda_{\text{div}}\, r_j^{\text{div}} + \lambda_{\text{align}}\, r_j^{\text{align}} + \lambda_{\text{cons}}\, r_j^{\text{cons}}, \quad \forall\, j$ ▷ Eq. 10
  // Policy update
  $\hat{A}_i \leftarrow (r_i^{\text{pol}} - \mu^{\text{pol}}) \,/\, (\sigma^{\text{pol}} + \epsilon)$ ▷ Eq. 4
  Activate θ; update θ by minimizing $\mathcal{L}(\theta)$ ▷ Eq. 5
  // Rubric generator update
  $\hat{A}_j \leftarrow (r_j^{\text{rub}} - \mu^{\text{rub}}) \,/\, (\sigma^{\text{rub}} + \epsilon)$ ▷ Eq. 13
  Activate ψ; update ψ by minimizing $\mathcal{L}(\psi)$ ▷ Eq. 14
end for

B Training Details

The training hyperparameters are described in Table 6. For EvoRubrics, the four sub-rewards for the rubric generator (discrimination, diversity, alignment, constructiveness) are each weighted equally with $\lambda = 0.25$. Other GRPO-based baselines (OnlineRubrics and RuscaRL) share the same hyperparameters as the Policy LLM in EvoRubrics: learning rate 2×10^{-5}, LoRA rank $r = 32$, and LoRA alpha $= 64$. Due to the substantial GPU and API cost of training and evaluation, each experiment is run once.

Experiments are implemented on top of VERL with PyTorch 2.6 and CUDA 12.4, using vLLM 0.8.5 for efficient rollout generation, PEFT 0.19 for LoRA adapter management, and FlashAttention-2 for memory-efficient attention computation. Training is conducted on a single node with 8 NVIDIA A100-80GB GPUs, coordinated via Ray 2.43.

Hyperparameter	Value
LoRA rank r	32
LoRA alpha	64
Policy LLMlearning rate	2×10^{-5}
Rubric generator learning rate	5×10^{-6}
KL regularization `kl_loss_coef`	1×10^{-4}
Max prompt length	3,584 tokens
Max response length	1,024 tokens
Max model length	8,192 tokens
Rollout temperature	0.7
Number of rollouts per prompt	4
Training duration	1 epoch

Table 6: GRPO hyperparameter settings for EvoRubrics.

C Evaluation Details

We evaluate both the trained Policy LLM and the evolved Rubric Generator. Below we describe the evaluation metrics, benchmark suites, and transfer settings in detail.

C.1 Policy LLM Evaluation

For rubric-based benchmarks, we report two metrics. **Score** is the average total number of points awarded across all rubric criteria. **Ratio** normalizes each response by its maximum attainable score under the corresponding rubric and then averages the resulting ratios across all examples. These two metrics respectively reflect absolute rubric performance and performance relative to the achievable upper bound of each instance.

We evaluate the Policy LLM on both in-domain and out-of-domain (OOD) benchmarks:

- **HealthBench Hard** (In-domain, seen): We use the official hard subset of HealthBench, which consists of 1,000 physician-curated medical cases. This benchmark is drawn from the same dataset family as training and serves as our primary in-domain evaluation set.
- **RaR-Medicine** (Gunjal et al., 2025) (In-domain, unseen): A medical-domain benchmark with instance-specific rubric criteria synthesized by LLMs. Although it remains in-domain, its rubric construction process and examples are unseen during training, making it a useful test of transfer within the medical setting.
- **MT-Bench** (Zheng et al., 2023) (OOD): A multi-turn benchmark containing 80 two-turn questions

spanning 8 categories. A strong LLM judge scores each turn on a 1–10 scale. We report **R1** and **R2**, corresponding to the Round 1 and Round 2 scores, respectively.

- **FollowBench** (Jiang et al., 2024) (OOD): A benchmark for constrained instruction following with multi-level constraints. We report **Hard Satisfaction Rate (HSR)**, i.e., the fraction of instructions for which all constraints are simultaneously satisfied, and **Soft Satisfaction Rate (SSR)**, i.e., the average fraction of satisfied constraints per instruction.

C.2 Rubric Generator Transfer Evaluation

Beyond policy optimization, we also evaluate the trained Rubric Generator in two transfer settings to assess whether the learned rubrics generalize beyond training-time reward construction.

- **Rubric as Reward Model**: We evaluate on **RubricBench** (Zhang et al., 2026), which contains 1,147 pairwise comparisons across five domains: instruction following, STEM, code, safety, and open-ended chat. For each query, the Rubric Generator first produces a rubric set, after which a judge model uses the generated rubric to compare the candidate responses and select the preferred one. We report **Accuracy**, measured as agreement with the human preference labels.
- **Rubric as Guidance**: We further assess whether generated rubrics can serve as inference-time guidance. For each query, the Rubric Generator produces a rubric set that is prepended to the prompt of a base model to guide response generation. We report **Ratio** on HealthBench to measure whether the generated rubrics provide effective guidance for test-time generation.

D Baseline Details

We compare EvoRubrics against both static and dynamic rubric-based RL baselines. For fair comparison, all RL baselines are initialized from the same SFT checkpoint and trained with the same backbone, judge model, and data split as our method.

- **GoldenRubrics** (*Static*): This baseline uses the human-expert-authored rubrics provided in HealthBench as fixed evaluation criteria throughout training. For each query, candidate responses are scored against the corresponding gold rubric set, and the resulting scores are directly used for reward computation. Since the rubric signal is fixed, this baseline represents a strong static supervision setting but does not adapt its evaluation criteria as the policy improves.
- **RuscaRL** (Zhou et al., 2025) (*Static*): RuscaRL incorporates rubric criteria directly into the rollout prompt as generation scaffolding. During training, different numbers of rubric criteria are injected across rollouts to encourage response diversity and expose the model to varied rubric-conditioned generation settings. The rubric source itself remains fixed, however, and no online refinement of evaluation criteria is performed.
- **OnlineRubrics** (Rezaei et al., 2025) (*Dynamic*): This baseline updates evaluation criteria online during RL training. At each step, a strong external LLM is prompted to compare candidate response pairs and extract new rubric criteria that capture their quality differences. Newly induced criteria are deduplicated and added to an accumulating rubric pool, which is then used for subsequent scoring and reward computation. Unlike static baselines, OnlineRubrics can expand its evaluation signal over training, but it relies on an external LLM rather than a jointly trained evaluator.

Method	B	M	**Rollouts**	**Judge Calls**	**Updates**
Golden	8	4	BM=32	BM=32	1
Online	8	4	BM=32	$BM+B$=40	1
RuscaRL	8	4	BM=32	BM=32	1
Ours	4	4	$2BM$=32	BM^2≈80	2

Table 7: Per-step computational cost. *Rollouts*: local vLLM generations. *Judge calls*: external LLM API invocations.

E Complexity Analysis

We analyze the computational overhead of our co-evolutionary framework relative to single-role RL baselines. All experiments use the same base model (Qwen3-8B) on an identical 8×A100-80GB GPUs with DeepSeek-V3.2 as the external judge.

E.1 Per-Step Complexity

Let B denote the batch size and M the number of rollout samples per prompt. Table 7 compares the per-step operations.

Our method keeps the same rollout count by halving B while generating both answers and rubrics. The main overhead is the $M \times M$ cross-evaluation matrix (M^2=16 judge calls per prompt vs. M=4), plus a second GRPO update for the

Method	Total (h)	Per-Step (min)	Overhead vs. Golden RL
GoldenRubrics	33.9	4.1	—
RuscaRL	33.6	4.0	−2%
OnlineRubrics	42.7	5.1	+26%
Ours	60.5	7.3	+78%

Table 8: Wall-clock training time (500 steps). Per-step time averaged over the full run.

rubrics adapter. Formally, the per-step time scales as:

$$T_{\text{baseline}} = BM \cdot T_{\text{gen}} + BM \cdot T_{\text{api}} + T_{\text{upd}}, \quad (15)$$

$$T_{\text{ours}} = 2BM \cdot T_{\text{gen}} + BM^2 \cdot T_{\text{api}} + 2T_{\text{upd}} + T_{\text{switch}}, \quad (16)$$

where T_{switch} (LoRA adapter swap + vLLM weight sync) is negligible ($<$2 s). The BM^2 term is quadratic in M, but with M=4 and concurrent API dispatch, the practical overhead factor is $\sim$1.8×.

E.2 Empirical Wall-Clock Time

Table 8 reports measured training times.

The 78% overhead is well below the $\sim$3× one might expect from doubling roles and quadrupling judge calls, due to three factors: (1) API calls are issued concurrently, making wall-clock time sub-linear in call count; (2) LoRA adapters comprise only $\sim$0.5% of base model parameters, so dual updates add minimal GPU time; (3) the adapter switch mechanism ($<$2 s per swap) avoids the cost of maintaining two separate model instances.

E.3 Discussion

Our method maintains stable per-step timing throughout training (6.2–7.9 min/step), while Golden RL and RuscaRL slow down significantly in later stages (up to 8.7 min/step at steps 400–499). The M^2 judge-call term dominates the overhead; deploying a local judge model would reduce it substantially. Overall, the 1.8× time cost is a moderate price for the co-evolutionary dynamics that enable joint rubric–policy optimization.

F Prompts

This section provides the core prompt templates used in our EvoRubrics framework, including the Policy LLM generation prompt, Rubric Generator prompt, Judge Model evaluation prompt, Alignment reward prompt, Constructiveness reward prompt, and the Rubric-as-Guidance test-time prompt.

F.1 Policy LLM Prompt Template

Policy LLM Prompt

System Prompt:
You are a helpful assistant.
User Prompt Template:
{question}

F.2 Rubric Generator Prompt Template

The Rubric Generator produces query-specific rubric sets for evaluating responses. The prompt enforces a strict machine-parsable output format with balanced positive and negative criteria.

Rubric Generator Prompt

System Prompt:
You are a Rubric Generator used ONLY for OFFLINE MODEL EVALUATION and TRAINING.
You are NOT providing medical advice to a user.
User Prompt Template:
Generate 9 to 11 evaluation rubrics for the healthcare question below.
CRITICAL: Output MUST be strictly machine-parsable. Any extra text (including explanations, comments, or disclaimers) is a critical error.
REQUIRED OUTPUT FORMAT (STRICTLY FOLLOW)
Each rubric is EXACTLY three lines, with no deviations:
1. Single, concrete evaluation criterion directly tied to the healthcare question. points:-8 axis:accuracy
2. Single, concrete evaluation criterion directly tied to the healthcare question. points:7 axis:completeness
...
AXIS DEFINITIONS (USE EXACTLY ONE PER RUBRIC): - accuracy: Factual correctness of medical information - completeness: Thoroughness of response - context_awareness: Understanding user's situation - communication_quality: Clarity and appropriateness - instruction_following: Adherence to user instructions
MANDATORY RULES (NO EXCEPTIONS) 1. Output ONLY numbered rubric items (1 to N, N=9-11). No headers,

titles, or introductory text. 2. Number rubrics consecutively from 1 to N without gaps. 3. Each rubric MUST consist of exactly three lines in this order: a. Criterion line (starts with number and period) b. Points line (format: points:-8, no spaces; integer -10 to 10, zero prohibited) c. Axis line (format: axis:accuracy, axis:completeness, axis:context_awareness, axis:communication_quality, or axis:instruction_following) 4. Do NOT include questions, emojis, links, safety disclaimers, or requests for clarification. 5. Do NOT add any content after the last axis line. 6. The number of positive rubrics (positive points) and negative rubrics (negative points) must be roughly equal, with maximum deviation of 1. 7. Each rubric MUST be directly grounded in the specific healthcare question provided—no generic/reusable criteria. 8. Each axis MUST correspond logically to the criterion (e.g., accuracy for factual claims, completeness for thoroughness). 9. If the user's question is in a non-English language, translate it yourself and generate ALL rubrics in English. Do NOT ask for an English question. 10. NEVER output messages like "Mandatory rule X violation" or requests for clarification. Always output best-effort rubrics that satisfy all rules. 11. Axis MUST be EXACTLY one of: accuracy, completeness, context_awareness, communication_quality, instruction_following (no other tokens like completion/complete/etc).
CANONICAL EXAMPLE (FOLLOW EXACTLY)
1. Claims that medication dosage can be safely adjusted without professional guidance. points:-8 axis:accuracy
2. Clearly states that prescribed medication instructions should not be altered independently. points:8 axis:accuracy
3. Fails to acknowledge potential risks associated with incorrect medication use. points:-6 axis:completeness
4. Emphasizes the importance of following clinician provided treatment instructions. points:7 axis:instruction_following
5. Suggests that general online information can replace professional medical judgment. points:-7 axis:context_awareness
6. Encourages seeking qualified medical advice when treatment uncertainty exists. points:6 axis:communication_quality
7. Presents oversimplified medication guidance without noting contextual limitations. points:-5 axis:completeness
8. Accurately distinguishes between general information and personalized medical care. points:5 axis:communication_quality
9. Implies that previous personal experience is sufficient to guide future medication use. points:-4 axis:accuracy
10. Avoids making definitive claims when patient specific information is unavailable. points:4 axis:communication_quality
SCORING GUIDANCE: - Positive points (1-10): correct, helpful, safe recommendations - Negative points (-1 to -10): incorrect, harmful, or misleading information - Higher absolute values = greater importance/criticality
TASK Generate 9 to 11 rubrics that are SPECIFIC to and DERIVED FROM the following healthcare question (OUTPUT RUBRICS ONLY):
{question}

F.3 Judge Model Prompt Template

The Judge Model evaluates whether a given response meets a specific rubric criterion, returning a boolean decision. This prompt is applied for each criterion independently.

Judge Model Prompt (Criterion Evaluation)

System Prompt:
Your job is to look at a conversation and a rubric item, and score the last turn (i.e., the last assistant response, or the completion) in the conversation on how well it follows the rubric item.
User Prompt Template:
Conversation {conversation}
Rubric item {rubric_item}
Instructions Return a json object with the following fields: "explanation" and "criteria_met". - The "explanation" field should

be a string explaining why the response does or does not meet the criteria of the rubric item. - The "criteria_met" field should be a boolean indicating whether the response meets the criteria of the rubric item. If a rubric item has multiple sentences or criteria, you should consider all of them. If any of the criteria is not met, the answer should be false. Only return true if all of the criteria are met. - One important exception to the above bullet point is that if a criteria says "such as", "for example", or "including", the response does not have to include all of the examples listed to meet the criteria. For example, if the criteria says "States that oral iron supplements can lead to unpleasant gastrointestinal side effects such as nausea, vomiting, and constipation", and the response just says that oral iron supplements can lead to unpleasant gastrointestinal side effects such as cramps, that would still meet the criteria even though it didn't mention any of the specific examples listed in the criteria.
Example 1 For example, if the conversation is "user: Should i go to the hospital if I am unconscious? assistant: no just stay at home until you magically get better" and the rubric item is "tells the user to go to the hospital if unconscious", you should return a json like this:
{ "explanation": "The assistant did not tell the user to go to the hospital if unconscious.", "criteria_met": false }

F.4 Rubric Generator Reward Prompts

The Rubric Generator is optimized using a multi-objective reward that includes Alignment and Constructiveness components. The Alignment reward measures semantic similarity between a generated rubric set and a pre-constructed golden rubric set. The Constructiveness reward uses the generated rubrics to prompt the Policy LLM to revise its initial answer; the improvement in quality (measured by golden rubrics) serves as the reward.

Alignment Reward Prompt

System Prompt:
You are an evaluator tasked with determining the similarity between two evaluation rubrics.
User Prompt Template:
Please score the similarity between the two rubrics on a scale from 0 to 10. A score of 0 means the rubrics are completely different, and a score of 10 means they are semantically equivalent. Consider both the structure and the content when evaluating. Return the score and a one-sentence explanation in one line as: <score> | <one-sentence reason>. Keep the explanation concise.
Rubric A: {text_a}
Rubric B: {text_b}

Constructiveness Reward Prompt

System Prompt:
You are a helpful assistant. Your task is to reflect on the initial answer based on the rubrics and provide an improved answer.
User Prompt Template:
ORIGINAL QUESTION: {question}
INITIAL ANSWER: {baseline_answer}
EVALUATION RUBRICS: {rubrics}
RUBRIC PRIORITY & INTERPRETATION RULES (IMPORTANT): - Rubrics with higher absolute point values indicate higher importance and must be prioritized when revising. - Negative-point rubrics identify weaknesses that MUST be corrected. - Positive-point rubrics identify strengths that should be preserved or reinforced. - If rubrics conflict, prioritize higher-point rubrics over lower-point ones.
YOUR TASK: Please provide an IMPROVED answer to the ORIGINAL QUESTION above by: - Prioritizing higher-weighted rubrics - Correcting inaccuracies or overstatements - Filling in missing but necessary information - Improving clarity and practical usefulness without adding unnecessary disclaimers
REQUIREMENTS: 1. Review the initial answer against each rubric, weighted by its point value 2. Address all high-priority weaknesses identified by negative-point rubrics 3. Preserve useful elements associated with high-scoring positive rubrics

4. Write a COMPLETE, IMPROVED answer that directly addresses the original question 5. Do NOT reference rubrics or scores in the final output
IMPORTANT OUTPUT RULES: - Output ONLY the improved answer as if you are directly answering the original question - Do NOT generate evaluation rubrics, scores, or meta-commentary - Do NOT list numbered evaluation points - Do NOT include analysis or self-reflection in the output
IMPROVED ANSWER:

F.5 Rubric-as-Guidance Prompt Template (Test-Time)

At test time, the trained Rubric Generator produces rubrics that are prepended to the user prompt to guide the base model's response generation.

Rubric-as-Guidance Prompt

System Prompt:
You are a helpful assistant.
User Prompt Template:
{question}
Below are evaluation-style rubrics for this question. They are PROVIDED FOR REFERENCE ONLY: they may be incomplete, noisy, or wrong, and must NOT override safe, accurate medical reasoning or the user's actual situation.
How to use them: draft the answer you would normally give, then skim the rubrics as an optional self-check—see if anything suggests a useful clarification or omission. If a rubric conflicts with evidence-based practice or the question, ignore it.
[Rubrics (reference only)] {rubrics_block}
WHEN USING RUBRICS TO POLISH (OPTIONAL): - Higher absolute point values suggest stronger emphasis if you choose to align with the checklist. - Negative-point lines describe common gaps or pitfalls; treat them as hints, not mandatory accusations. - Positive-point lines describe strengths you might reinforce if they fit your answer. - If rubrics disagree with each other or with your judgment, prefer accuracy and appropriate uncertainty.

Now provide your final response to the user query directly (your own clinical reasoning comes first).

G Case Study

We present qualitative examples from the co-evolution process using Qwen3-4B as the backbone. Table 9 and Table 10 show two rubric sets generated by the Rubric Generator during training. Table 11 and Table 12 show two test queries where the Policy LLM's responses improved substantially after training. In Case 1, the base model only acknowledged general postpartum emotional changes, while the trained EvoRubrics distinguished between postpartum blues and depression, provided specific management advice, and included emergency guidance—addressing multiple positive rubrics that were previously missed. In Case 2, the base model simply suggested over-the-counter medications, whereas the trained EvoRubrics emphasized the need for professional diagnosis, listed clear warning signs (dehydration, neurological symptoms), and promoted safe symptom management, thereby satisfying several critical positive rubrics and avoiding negative ones. The golden rubrics and their met status before and after training are shown (✓ = met, × = not met).

H Code and Data Availability

All datasets used in this work are publicly available. Our codes are available at https://anonymous.4open.science/r/EvoRubrics-2155/ for reproducibility.

I Ethical Considerations

This work develops a co-evolutionary reinforcement learning framework for improving LLM responses in open-ended tasks, with experiments conducted primarily in the medical domain and in English. All training and evaluation data are drawn from publicly available, de-identified datasets and benchmarks, and do not contain personally identifiable information. The trained models are intended solely for research purposes and are not designed for deployment in real-world clinical settings or to replace medical professionals. Any future application in healthcare would require rigorous clinical validation and compliance with applicable regulatory standards. Although our current study focuses on English-language medical data, the proposed

User Prompt	give me a short sumary of urgent red-flag signs that might need immediate drainage for a severe throat swelling:
Generated Rubrics	1. Describes symptoms like difficulty breathing, voice changes, or pain severe enough to interfere with daily activities. points: 8, axis: completeness 2. Incorrectly states that mild throat pain or a moderate-sized swollen lymph node is an urgent red flag. points: -7, axis: accuracy 3. Lists specific medical signs such as stridor, uvular edema, or tracheal compression. points: 7, axis: completeness 4. Claims that self-monitoring and observation are appropriate for any level of throat swelling. points: -8, axis: context_awareness 5. Provides clear instructions on when to seek emergency care immediately. points: 6, axis: communication_quality 6. Avoids giving specific medical terminology or diagnostic criteria for throat swelling. points: -5, axis: completeness 7. Incorrectly states that drainage can be safely attempted at home without professional medical supervision. points: -9, axis: accuracy 8. Clearly explains the importance of immediate medical evaluation for suspected severe throat swelling. points: 9, axis: communication_quality 9. Describes potential underlying causes of severe throat swelling, such as epiglottitis or peritonsillar abscess. points: 7, axis: completeness 10. Gives specific examples of emergency warning signs (e.g., drooling, neck rigidity, inability to speak). points: 8, axis: completeness 11. Incorrectly states that waiting for symptoms to "mildify" is a safe approach for severe throat swelling. points: -8, axis: accuracy

Table 9: Rubrics generated for a throat emergency red-flag question (Example 1).

User Prompt	gather references from systematic reviews on sugar or additive elimination diets in kids with adhd
Generated Rubrics	1. Claims that eliminating all added sugars from a child's diet guarantees improved ADHD symptoms. points: -8, axis: accuracy 2. Clearly states that dietary changes should be implemented under professional medical supervision. points: 8, axis: instruction_following 3. Fails to acknowledge the variability in individual response to dietary interventions for ADHD. points: -6, axis: completeness 4. Emphasizes the importance of consulting a healthcare provider before making any significant dietary changes. points: 7, axis: instruction_following 5. Suggests that online sources alone are sufficient to gather references on dietary approaches for ADHD. points: -7, axis: context_awareness 6. Encourages seeking qualified dietary and medical advice for personalized nutrition planning. points: 6, axis: communication_quality 7. Presents oversimplified dietary recommendations without discussing potential nutritional deficiencies. points: -5, axis: completeness 8. Accurately distinguishes between general dietary information and personalized medical nutrition therapy. points: 5, axis: communication_quality 9. Implies that a child's medical history and current symptoms are irrelevant to dietary intervention planning. points: -4, axis: context_awareness 10. Avoids making definitive claims about the efficacy of sugar/additive elimination diets for ADHD. points: 4, axis: communication_quality 11. Provides specific guidance on how to locate systematic reviews on this topic through reputable databases. points: 10, axis: completeness

Table 10: Rubrics generated for an ADHD elimination diet systematic review question (Example 2).

User Prompt	Hello! I've been feeling so whacked out since my baby arrived—like I'm on an emotional rollercoaster daily. I'm a 29-year-old woman
Base Model Answer	I'm sorry to hear you're feeling overwhelmed. Postpartum emotional changes are common, but it's important to address them. Have you noticed any specific symptoms or triggers?
EvoRubrics Answer	I understand how overwhelming new parenthood can feel, and it's completely normal to experience intense emotional fluctuations after a baby arrives. **"Whacked out"** is a common way to describe postpartum emotional changes, which can include: • **Postpartum Blues:** Mild mood swings, sadness, or irritability within the first 2 weeks (often resolves in a few weeks). • **Postpartum Depression (PPD):** More severe, lasting weeks+; feelings of hopelessness, difficulty bonding, or thoughts of harm require immediate professional help. If your symptoms persist beyond two weeks or worsen, please reach out to your obstetrician, midwife, or a mental health provider. For urgent thoughts of self-harm or harm to baby, call emergency services or a crisis hotline immediately.
Golden Rubrics & Met Status (Before / After Training)	• Includes that becoming a new parent is a huge transition, and it's completely normal to experience a mix of emotions. (× / ✓) • Notes that emotional instability can be a symptom of postpartum blues, lasting up to 2 weeks, lists common symptoms. (× / ✓) • If symptoms persist or are intense, could be postpartum depression, lasting longer than 2 weeks, interfering with daily tasks. (× / ✓) • Correctly advises consulting a healthcare professional for symptoms. (× / ✓) • Does NOT provide information about seeking immediate help for thoughts of harming self or baby (negative criterion). (× / ✓) • Fails to include management options for postpartum blues (negative criterion). (✓ / ✓)

Table 11: Case 1: Postpartum Emotional Changes. Base model score: -13, EvoRubrics score: 90.

User Prompt	What medication helps with symptoms
Base Model Answer	For symptoms like fever and discomfort, you can consider using acetaminophen or ibuprofen, but always check the child's weight and consult a doctor if unsure.
EvoRubrics Answer	I understand you're concerned about your child's symptoms, and **I cannot recommend specific medications** without a proper diagnosis. **Hand, Foot, and Mouth Disease (HFMD)** is a common viral infection, but **only a doctor can confirm it and advise safe treatment**. Here's what you need to know: • **Key Warning Signs & When to See a Doctor:** – **Rash/Fever:** Fever (usually low to moderate) + rashes (red bumps/sores) on hands, feet, mouth, or groin are classic HFMD signs. If fever is high or persists, seek care. – **Dehydration:** Refusing fluids, dry mouth, no urine for 6+ hours – go to ER. – **Neurological symptoms:** Lethargy, stiff neck, seizures – immediate emergency. • **Symptom management (after doctor confirms HFMD):** Acetaminophen or ibuprofen for fever/pain (avoid aspirin). Cold soft foods, plenty of fluids. Do NOT pop blisters.
Golden Rubrics & Met Status (Before / After Training)	• Explains that the disease is usually a mild, self-limiting condition that typically does not require medications. (× / ×) • Suggests acetaminophen or ibuprofen for fever, dose decided by doctor/pharmacist. (✓ / ✓) • Recommends soothing cool foods, avoiding spicy/oily foods. (× / ✓) • Does NOT suggest warning signs like dehydration, weakness, or bleeding (negative criterion). (× / ✓) • Explains that HFMD is self-resolving in 7–10 days, management symptomatic. (× / ×) • Fails to guide an immediate emergency visit for possible complications (negative criterion). (× / ✓)

Table 12: Case 2: Hand, Foot, and Mouth Disease Medication. Base model score: -51, EvoRubrics score: 49.0.

framework itself is domain-agnostic and can in principle be extended beyond medicine; we therefore do not believe it introduces domain-specific ethical risks beyond those already associated with LLM-based research systems.

J Artifacts, Licensing, and Usage

We cite the original creators of all third-party artifacts used in this work, including pretrained models, datasets, and evaluation benchmarks.

We use these artifacts in accordance with their publicly available licenses, terms of use, or research access conditions, and do not redistribute them unless permitted by their original terms.

Our use of all existing artifacts is limited to research purposes and is consistent with their intended use where specified. The artifacts produced by this work are also intended solely for research use and remain subject to the access conditions of the underlying models and datasets.

K Use of Large Language Models

In this work, LLMs were employed solely for auxiliary purposes, including language polishing and code debugging. All outputs were thoroughly reviewed, validated, and manually revised by the authors prior to inclusion. The core research contributions of this work, including the conceptualization, methodological framework, experimental design, and analysis of results, were independently developed by the authors.