EvoLM: Self-Evolving Language Models through Co-Evolved Discriminative Rubrics

Anonymous authors
Paper under double-blind review

Abstract

Language models encode substantial evaluative knowledge from pretraining, yet current post-training methods rely on external supervision (human annotations, proprietary models, or scalar reward models) to produce reward signals. Each source brings a different constraint: human judgment is costly and hard to scale, proprietary APIs create dependencies, and verifiable rewards cover only domains with ground-truth answers. A model's own evolution offers a natural source of supervision for self-improvement, yet remains largely untapped by current methods. We introduce EvoLM, a post-training method that turns this supervision into rewards for further learning. EvoLM trains two capabilities within a single language model in alternation: (1) a rubric generator producing instance-specific evaluation criteria optimized for *discriminative utility*, which maximizes a small frozen judge's ability to distinguish preferred from dispreferred responses; and (2) a policy trained using those rubric-conditioned scores as reward. All preference signals are constructed from the policy's own outputs via temporal contrast with earlier checkpoints, requiring no human annotation or external supervision. The co-trained policy achieves 69.2% average on the OLMo3-Adapt suite, exceeding policies trained with GPT-4.1-prompted rubrics by 2.5 points and those trained with the state-of-the-art 8B reward model Skywork-RM-V2 by 9.5 points. The learned rubrics transfer to judges they were never trained against, raising a Qwen3-8B judge's RewardBench-2 accuracy from 39.7% to 62.4%. Rubrics that co-evolve with the policy therefore supply training signal that external reward sources do not.

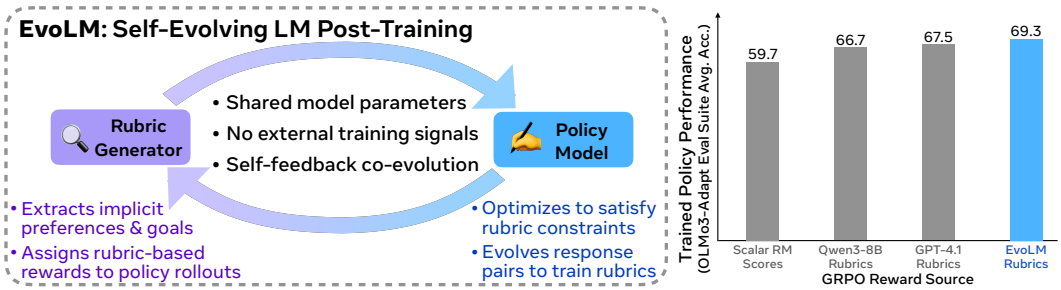

Figure 1: EvoLM Overview. Our method enables a single language model to co-evolve its own evaluation and generation capabilities using its own evolution as supervision. By extracting latent knowledge into explicit, instance-specific rubrics, EvoLM creates an autonomous feedback loop that improves policy performance without requiring human annotations or external teacher models.

1 Introduction

Reinforcement learning (RL) has become central to language model (LM) post-training, driving improvements in instruction following, reasoning, and safety (Ouyang et al., 2022; Bai et al., 2022). The quality of the reward signals fundamentally constrains what the policy can learn. Current approaches derive this signal from external supervision, including human preference annotations (Christiano et al., 2017), scalar reward models susceptible to reward hacking (Stiennon et al., 2020; Gao et al., 2023), verifiable rewards restricted to domains with ground-truth answers (Lambert

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.

et al., 2025; Cobbe et al., 2021), and proprietary LLM-as-a-judge evaluation (Zheng et al., 2023). Yet LMs already encode substantial evaluative knowledge from pretraining (Feng et al., 2026); RL surfaces and sharpens this knowledge rather than introducing it (Shao et al., 2026b). The question is how to turn this knowledge into rewards for self-improvement without externally supplied preference labels. We use the model's own evolution as supervision for learning those rewards.

In non-verifiable domains, evaluation can be factored into rubric generation (specifying what to measure) and judging (scoring against those criteria). This decomposition externalizes the model's evaluative knowledge into structured criteria, allowing even a small judge to score reliably when given concrete criteria that it could not derive on its own. It also provides interpretability, where the rubric makes evaluation criteria inspectable, and modularity, such that a rubric generator trained with one judge can be deployed with another. A growing body of work applies rubrics to RL post-training, but existing methods rely on external supervision such as task-specific verifiers or labeled preference data (Sheng et al., 2026; Xu et al., 2026; Lv et al., 2026), or a proprietary teacher model for rubric generation (Kim et al., 2024; Gunjal et al., 2026; Shen et al., 2026; Shao et al., 2026a). But to our knowledge, no existing method formally defines what makes a rubric useful or trains the rubric generator end-to-end to optimize for it without external supervision.

This gap motivates a natural criterion for rubric quality: a rubric is useful when it helps the judge distinguish good responses from bad ones. Given a preference pair, we can directly test whether the judge, conditioned on a candidate rubric, assigns a higher score to the preferred response. Rubric quality is therefore directly measurable and trainable. We can optimize the rubric generator to maximize this *discriminative utility*, rewarding rubrics that widen the judge's score gap between preferred and dispreferred responses. The resulting training signal requires no human-written rubric annotations, no external labels, and no domain-specific verifiers; only preference pairs derived from the policy's own outputs. These pairs are constructed via temporal contrast, where current policy responses (preferred) are paired against responses from earlier checkpoints (dispreferred). We propose **EvoLM**, a post-training method in which a single language model learns to generate both responses and the evaluation rubrics used to score them. The rubric generator and policy co-evolve through alternating updates (Figure 2). As the policy improves, rubrics must sharpen to remain discriminative; sharper rubrics in turn yield more informative reward, driving further policy improvement. We formalize rubric generation as variational inference over latent rubric variables and derive a principled training objective (Section 2).

Across twelve benchmarks in math, code, general reasoning, knowledge, instruction following, and open-ended generation, EvoLM produces the strongest downstream policy (69.2% average on the OLMo3-Adapt suite), outperforming both GPT-4.1-prompted rubrics (66.7%) and four recent rubric-based RL methods that depend on proprietary APIs or labeled preference data (Section 3.2). A scalar reward model that leads both held-out preference benchmarks (86.4% RewardBench 2, 80.8% JudgeBench) trains the weakest policy, 9.5 points below EvoLM, so static preference accuracy is a poor proxy for training utility (Section 3.2). The self-evolving paradigm and the learned rubrics generalize broadly. The co-evolving framework extends to other model families including OLMo-3-7B, the learned rubrics transfer to unseen policies and judges without retraining, and on out-of-distribution deep research tasks, EvoLM achieves higher pairwise agreement with expert human rubrics than GPT-4.1 on both HealthBench (58.4% vs. 52.5%) and ResearchQA (59.3% vs. 51.0%) (Section 3.3). We further uncover the underlying self-evolving mechanism through qualitative analysis: the trained generator learns to decompose evaluation into individually checkable sub-tasks, embedding expected intermediate values directly into criteria and transforming holistic judgment into pattern matching that even a 1.7B-parameter judge performs reliably.

2 EvoLM

2.1 Overview

Given a question q, a **policy** π_θ generates responses $a \sim \pi_\theta(\cdot \mid q)$. Standard approaches train a scalar reward model $R(q, a) \in \mathbb{R}$ that scores each response; the evaluation criteria are implicit in model weights and cannot be inspected. We instead decompose reward computation into two components with an explicit natural-language interface. A **rubric generator** ρ_ϕ produces evaluation criteria r, expressed in natural language, for each question:

$$r = \rho_\phi(q). \tag{1}$$

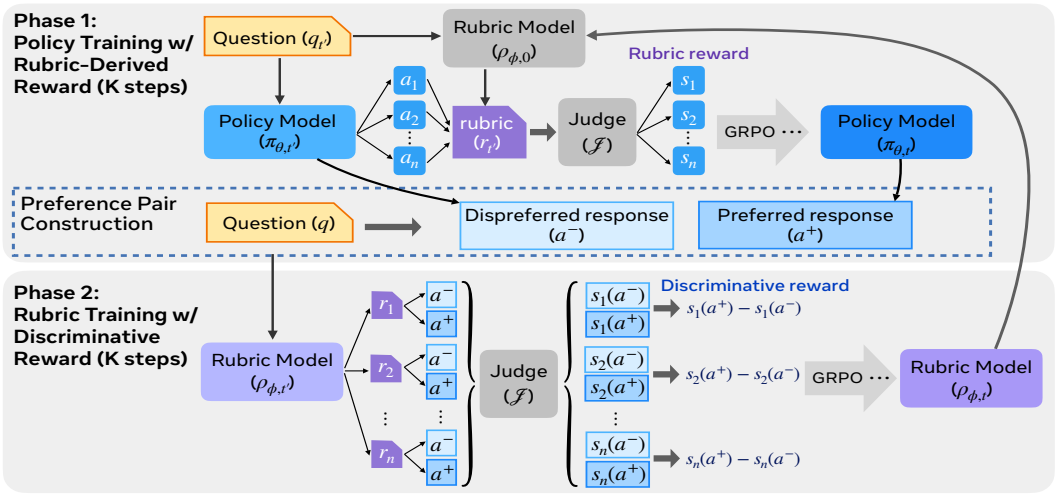

Figure 2: EvoLM alternates between two K-step phases. In Phase 1, a frozen rubric model ρ_ϕ generates a rubric r for each question; the policy π_θ samples n responses, which a frozen judge $\mathcal{J}$ scores to produce rubric-derived rewards used for policy updates. In Phase 2, the updated policy constructs preference pairs (a^+, a^-) from its own outputs; the rubric model samples n candidate rubrics, each scored by the judge on both responses, and the score margin $s(a^+) - s(a^-)$ serves as discriminative reward for rubric updates. The judge remains frozen throughout.

A **judge** J scores responses against those criteria:

$$s = J(q, r, a) \in [0, 1]. \tag{2}$$

We train the rubric generator and policy jointly, alternating between (a) policy updates using rubric-based judge scores as reward and (b) rubric generator updates using ranking accuracy of preference pairs from the policy's outputs as reward, with the judge's parameters fixed throughout (Figure 2).

2.2 Rubrics as Latent Variables

A rubric is good if it *discriminates*: when the judge applies it, preferred responses score higher. We can test whether a candidate rubric leads the judge to recover that ordering. This motivates treating the rubric as a latent variable that explains observed preferences. We formalize this through variational inference, deriving a training objective that rewards rubrics for reconstructing the preference ordering (Eq. 5).

For each observed preference pair, we introduce a latent variable $r \sim \rho_{\text{ref}}(r \mid q)$, representing the prior over rubrics from the base model. The likelihood of the observed preference given a latent rubric is:

$$p(a^+ \succ a^- \mid q, r) = \sigma\big(J(q, r, a^+) - J(q, r, a^-)\big), \tag{3}$$

where σ is the sigmoid function. The joint distribution over responses, preferences, and rubrics is:

$$p(a^+, a^-, r \mid q) = p(r \mid q)\, \pi(a^+|q)\, \pi(a^-|q)\, p(a^+ \succ a^- \mid q, r). \tag{4}$$

Here π denotes the distribution from which responses are drawn; it can be a policy or an offline preference dataset. In our setting, $\pi = \pi_\theta$ is the current policy, and responses are held fixed when optimizing the rubric generator parameters ϕ (Section 2.3). Eq. 4 generalizes the Bradley–Terry model (Bradley & Terry, 1952), which generates preferences from a latent scalar reward (Rafailov et al., 2023). We instead condition preference generation on a latent rubric variable, using natural language criteria rather than a single score.

We seek the posterior $p(r \mid a^+, a^-, q)$, which is intractable. Applying amortized variational inference (Kingma & Welling, 2014), we maximize the evidence lower bound (ELBO):

$$\log p(a^+, a^- \mid q) \geq \mathcal{L} = \mathbb{E}_{r \sim \rho_\phi(r|q)}\big[\log p(a^+ \succ a^- \mid q, r)\big] - \text{KL}\big(\rho_\phi(r \mid q) \| \rho_{\text{ref}}(r \mid q)\big), \tag{5}$$

where $\rho_\phi(r \mid q)$ is the variational distribution. Maximizing the ELBO brings ρ_ϕ close to the true posterior. Because rubrics are discrete text, we optimize this objective with policy gradient, treating

the rubric generator as an agent that receives the log-likelihood of correct preference reconstruction as reward:

$$\mathcal{L}_{\text{rubric}}(\phi) = \mathbb{E}_{r\sim\rho_\phi(r|q)}\big[R(r;q,a^+,a^-)\big] - \text{KL}\big(\rho_\phi(r \mid q)\|\rho_{\text{ref}}(r \mid q)\big), \tag{6}$$

where $R(r;q,a^+,a^-) = \log\sigma\big(J(q,r,a^+) - J(q,r,a^-)\big)$ encourages rubrics that, under a fixed judge, assign higher scores to preferred responses.

Practical reward design. We train with a margin-based reward combined with a format reward in place of the log-sigmoid reward in Equation (6):

$$R(r;q,a^+,a^-) = \alpha \cdot \big(J(q,r,a^+) - J(q,r,a^-)\big) + (1-\alpha)\cdot R_{\text{format}}(r) \tag{7}$$

where $\alpha = 0.7$. The margin provides continuous signal proportional to the score gap. The format reward $R_{\text{format}}(r) \in \{0,1\}$ verifies that the rubric conforms to a structured JSON schema the judge can parse (Appendix A.3). This reward design is ablated in Appendix D.1.

2.3 Joint Training

Both the rubric generator and the policy are trained using GRPO (Shao et al., 2024), a policy gradient method that computes advantages from group-relative rewards without a learned value function. The policy's loss is:

$$\mathcal{L}_{\text{policy}}(\theta) = -\mathbb{E}_{q,a\sim\pi_\theta}\left[A(q,a)\log\pi_\theta(a|q)\right] \tag{8}$$

where advantages $A(q,a)$ are derived from judge scores $J(q,\rho_\phi(q),a)$ normalized within each group of responses to the same question. The rubric generator's loss $\mathcal{L}_{\text{rubric}}(\phi)$ (Equation 6) is optimized analogously, with advantages computed over groups of rubrics sampled for the same query.

Training alternates between two phases: the policy π_θ is updated for K steps while the rubric generator ρ_ϕ is held fixed, then ρ_ϕ is updated for K steps using preference pairs constructed from π_θ's outputs (Section 2.4). The two phases reinforce each other. As the policy evolves, the rubric generator must learn increasingly specific criteria to distinguish quality under the current output distribution. As rubrics become more discriminative, the policy receives sharper reward signal and improves further. The alternating frequency K controls coupling strength and is ablated in Section 3.4. We freeze judge J throughout training, isolating the rubric generator as the sole source of improvement in training signal.

2.4 Constructing Preference Pairs

The rubric generator's objective (Eq. 6) requires preference pairs (a^+,a^-). We study three ways to construct these pairs entirely from the policy's own outputs without human annotation. The main configuration uses temporal contrast; inferred-question and rubric-conditioned pairs are evaluated as alternative signals, both separately and in combination, in Section 3.4.

1. Temporal contrast. We store policy rollouts with their training step. To construct a pair for question q, we generate a current response $a^+ \sim \pi_\theta(\cdot \mid q)$ at step t and sample an earlier response a^- from step $t' < t$. Since π_θ improves over training, the later response is treated as preferred. The step gap $t - t'$ controls the difficulty of discrimination: a large gap produces easy pairs with clear quality differences, while a small gap produces harder pairs that require more fine-grained rubrics. This creates a natural curriculum as training progresses, since the earliest responses are gradually replaced by stronger ones. The assumption holds in aggregate, and individual pairs are sometimes reversed. Downstream quality at intermediate checkpoints of the main run rises from 67.3% at step 50 to 69.2% at step 500 (Appendix A.10); Appendix D.6 ablates the step gap.

2. Inferred question (IQ). Given a preferred response a^+ to question q, the policy itself infers what question a^+ appears to address, producing $\hat{q}$, following Wang et al. (2026). A dispreferred response is then generated as $a^- \sim \pi_\theta(\cdot|\hat{q})$. This provides signal about whether the rubric captures question-relevance: rubrics that check whether the response addresses the intended question will score a^+ higher than a^-.

3. Rubric-conditioned (RC). Given question q, we sample a rubric $r \sim \rho_\phi(\cdot \mid q)$ from the current rubric generator and generate a preferred response $a^+ \sim \pi_\theta(\cdot \mid q,r)$ conditioned on both the question and rubric, and a dispreferred response $a^- \sim \pi_\theta(\cdot \mid q)$ conditioned on the question alone. This

provides direct signal about rubric utility: if rubric-conditioned responses are consistently better, the rubric contains actionable criteria. The rubric generator is thus incentivized to produce criteria that concretely improve response quality when provided to the policy. Full prompts for all three methods are in Appendix E.2.

3 EXPERIMENTS

Experiments are structured around three questions: (1) Are trained rubric generators better than prompting, and does co-evolving training improve over sequential training (Section 3.2)? (2) Do learned rubrics generalize across domains, policy architectures, and judges (Section 3.3)? (3) What design choices most affect downstream policy quality (Section 3.4)?

3.1 SETUP

Data. Prompts are drawn from the Tulu 3 preference mixture (Lambert et al., 2025) and deduplicated, comprising approximately 271K prompts across general-purpose chat (UltraFeedback, WildChat), instruction following with IFEval-style constraints, math, code, scientific literature understanding, and persona-driven synthetic instructions.

Models. Qwen3-8B (Yang et al., 2025) is used as both policy and rubric generator in a parameter-sharing configuration, where a single model handles both roles via different prompts. This minimizes memory requirements; a two-model configuration is ablated in Section 3.4. Qwen3-1.7B serves as the default frozen judge. The self-evolving framing predicts that even small judges should provide effective reward signal when given discriminative rubrics; using a 1.7B judge tests this directly. The judge size is ablated in Section 3.4.

Training. GRPO is used with learning rate 10^{-6}, KL coefficient 0.001, and 8 response samples per question. The policy and rubric generator each receive 500 GRPO update steps, for 1,000 alternating updates in total. The alternating frequency $K = 50$ controls how many steps each component trains before switching; this is ablated in Section 3.4. The temporal contrast step gap $[20, 100]$ controls the temporal distance between paired responses; this is also ablated in Section 3.4. Other implementation details are in Appendix A.

Baselines. We compare against three baseline families.

1. Prompted rubrics. Two training-free baselines isolate the effect of learning the rubric generator. *GPT-4.1 prompted* generates rubrics with GPT-4.1 and scores them with the same Qwen3-1.7B judge as EVOLM, representing a strong proprietary baseline. *Qwen3-8B prompted* matches our rubric-generator architecture but uses prompting rather than training. Both prompted baselines use the generic 5–10-criterion prompt in Appendix E.3; the dealbreaker prompt in Appendix E.1 is part of our trained method and is not supplied to GPT-4.1.

2. Scalar reward model. An alternative to natural language rubrics is a scalar reward model that encodes evaluation criteria implicitly in its weights. We use Skywork-RM-V2 (Liu et al., 2026), a state-of-the-art general-purpose Bradley-Terry reward model, to assign rewards to rollouts during GRPO. This baseline isolates whether rubric factorization provides value beyond a strong scalar reward signal.

3. Rubric-based RL methods. We compare against four recent methods that also use rubrics during RL training. **RAR** (Gunjal et al., 2026) uses GPT-4.1 to generate rubrics (we replace the original proprietary judge with our Qwen3-1.7B judge for controlled comparison). **RRD** (Shen et al., 2026) has GPT-4.1 iteratively refine rubrics and combines per-item scores with whitened-uniform weighting. Both methods rely on a proprietary API for rubric generation and train only the policy; our method replaces this dependency with a trained local rubric generator. **RLCER** (Sheng et al., 2026) co-trains the rubric generator and policy but requires a task verifier for correctness filtering, limiting it to domains with ground-truth answers. **RUBRIC-ARM** (Xu et al., 2026) co-trains the rubric generator and the judge using reference pairs, restricting it to tasks with ground-truth preference labels at training time. Our method requires neither verifiers nor preference annotations during policy training.

All methods train for 500 policy update steps. Per-step compute varies: RRD and RLCER incur significantly more overhead from recursive refinement and per-rollout rubric generation, respectively. A detailed per-step cost breakdown is in Appendix A.7. Appendix A.6 provides full implementation details.

Evaluation. Downstream policy quality is the primary outcome: policies trained with learned rubrics are evaluated on twelve benchmarks from the OLMo3-Adapt suite, covering math reasoning (GSM8K, MATH), code generation (HumanEval+, MBPP+), general reasoning (BBH, GPQA, ZebraLogic, AGI-Eval), knowledge (MMLU, PopQA), instruction following (IFEval), and open-ended generation (AlpacaEval v3). We also evaluate *rubric quality* by measuring discriminative accuracy on RewardBench 2 (Malik et al., 2026) and JudgeBench (Tan et al., 2025), which test whether rubric-based evaluation correctly ranks preferred over dispreferred responses across diverse domains. We report single training runs rather than multi-seed averages; reruns at this scale exceed our compute budget. Downstream evaluation follows the OLMES protocol (Gu et al., 2025).

3.2 Main Comparisons

In addition to the baselines described above, we compare two training regimes for our method. **(1) Sequential training:** The rubric generator is trained first with the policy frozen, learning from preference pairs constructed via inferred question and rubric-conditioned methods (temporal contrast requires a changing policy). After rubric training converges, the policy is trained with the rubric generator frozen. This isolates the value of rubric training from the value of joint adaptation. **(2) Co-evolving training (EvoLM):** The rubric generator and policy are trained jointly with alternating updates. This enables co-adaptation: as the policy improves, it generates more varied responses that provide richer preference signals; as the rubric generator improves, it provides sharper reward signals. All three preference signal methods, including temporal contrast, can be used. We present our results in Table 1.

Table 1: Main comparison of training approaches. Each method produces a rubric generator (evaluated by RewardBench-2 and JudgeBench discriminative accuracy) and a policy (evaluated on OLMo3-Adapt downstream benchmarks). HE+ = HumanEval+. All scores are percentages.

	RewardBench2							JudgeBench					
Method	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
GPT-4.1 (prompted)	35.2	27.9	42.7	38.4	59.6	50.5	42.4	41.6	48.0	30.4	45.2	42.0	41.3
Qwen3-8B (prompted)	27.8	26.5	38.1	25.0	30.2	9.7	26.2	34.4	38.8	30.4	42.9	36.0	36.6
Scalar RM (Skywork-RM-V2)	**88.2**	**67.8**	**83.1**	**97.3**	**99.2**	**83.1**	**86.4**	**77.3**	**75.5**	**89.3**	**81.0**	**79.1**	**80.8**
RAR	N/A (frozen GPT-4.1 rubric)							N/A (frozen GPT-4.1 rubric)					
RRD	N/A (frozen GPT-4.1 rubric)							N/A (frozen GPT-4.1 rubric)					
RLCER	33.1	29.8	52.2	48.3	51.9	54.8	45.0	43.5	40.8	48.2	64.3	46.0	49.2
RUBRIC-ARM[1]	48.8	39.1	63.5	60.8	66.1	58.2	56.1	43.5	42.9	62.5	57.1	48.0	51.5
Sequential (IQ)	37.0	32.0	57.1	42.3	59.0	55.6	47.2	37.7	55.1	44.6	57.1	46.0	48.6
Sequential (RC)	31.5	30.8	57.0	47.9	49.6	59.2	46.0	34.4	39.8	44.6	50.0	39.4	42.2
EvoLM	37.2	30.6	54.0	43.5	54.4	56.1	46.0	45.5	39.8	30.4	61.9	43.4	44.4

	Downstream Policy Quality (OLMo3-Adapt)												
Method	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
GPT-4.1 (prompted)	95.7	94.2	58.7	65.1	**71.7**	84.8	34.4	30.7	54.9	81.9	82.9	45.5	66.7
Qwen3-8B (prompted)	95.5	94.6	73.7	58.4	66.8	83.4	35.3	31.7	56.9	**84.3**	83.9	**45.9**	67.5
Scalar RM (Skywork-RM-V2)	95.6	94.3	47.9	66.2	54.9	82.0	37.3	25.2	46.2	56.4	81.6	28.3	59.7
RAR	95.7	93.8	77.4	**68.6**	62.5	84.8	36.0	32.2	52.5	84.1	87.3	34.7	67.5
RRD	95.3	92.2	75.2	59.2	69.4	84.8	36.8	32.6	56.0	82.6	**88.9**	38.5	67.6
RLCER	94.0	**94.7**	80.5	67.9	68.6	84.8	**39.2**	30.4	49.6	69.1	84.3	37.2	66.7
RUBRIC-ARM	95.6	93.3	78.4	62.3	64.4	83.8	35.7	**32.8**	56.9	83.1	87.2	35.9	67.5
Sequential (IQ)	95.9	92.7	76.5	66.7	64.6	85.0	38.1	32.0	54.0	83.0	86.9	41.1	68.0
Sequential (RC)	**96.0**	93.4	80.2	67.7	64.7	**85.8**	36.8	32.3	**57.1**	83.2	87.4	35.1	68.3
EvoLM	95.8	92.2	**86.2**	68.5	67.6	85.3	37.7	32.1	54.2	82.3	86.4	42.2	**69.2**

The policy model trained by EvoLM achieves the highest downstream average (69.2%), outperforming all baselines without proprietary APIs, external labels, or task-specific verifiers. Prompted rubrics from GPT-4.1 (66.7%) and Qwen3-8B (67.5%) are competitive on individual benchmarks but fall behind on the aggregate, as are the four rubric-based RL methods (66.7–67.6%). The largest gains are on code generation (HumanEval+ 86.2% vs. 80.5% for the next best method), where the fine-grained criteria produced by trained rubrics provide particularly sharp signal. Co-evolving training outperforms sequential training on policy quality (69.2% vs. 68.3%), despite sequential training achieving higher static rubric accuracy (RB2 47.2% vs. 46.0%). Downstream policy quality depends on how well rubrics adapt to the evolving policy distribution, precisely the advantage co-evolution provides.

As a secondary rubric-discrimination evaluation, the learned generator scores 46.0% on RewardBench-2, compared with 42.4% for GPT-4.1-prompted rubrics.

Static preference accuracy is not predictive of downstream training utility in these experiments. Skywork-RM-V2 obtains 86.4% on RewardBench-2 and 80.8% on JudgeBench, but the policy trained with its rewards has the lowest downstream average (59.7%), 9.5 points below EvoLM. This result is consistent with prior work showing that held-out reward-model accuracy need not predict behavior under policy optimization (Gao et al., 2023; Ivison et al., 2024). Co-evolution instead optimizes rubric discrimination on the current policy distribution.

Across held-out checkpoints and saved training generations, repetition, prompt copying, lexical diversity and completion length remain stable (Appendix A.11).

3.2.1 Rubrics Evolve from Abstract Labels to Verifiable Checks

Qualitative analysis across four domains (math, emotional support, scientific explanation, and constrained writing) reveals a pattern: early rubrics use short labels or generic checks, while trained rubrics pack specific, verifiable expectations into each criterion, reducing the interpretive burden on the judge model. The form this takes varies by domain. In math, for instance, the rubric concentrates 80% of the weight on a single criterion that embeds the expected answer (e.g., "the correct maximum area of 144, derived from the given perimeter of 48"), transforming proof verification into answer checking; in constrained writing, it consolidates formatting and keyword requirements into explicit, countable checks. Aggregate statistics over the 100 evaluation prompts confirm this shift: label-only criteria fall from 21.9% to 0.3%, criteria embedding specific expected values rise from 6.9% to 19.3%, and constraint-type criteria rise from 7.7% to 20.3%, all relative to prompted Qwen3-8B (Appendix B.3). The common effect is moving evaluation from holistic semantic judgment, which small judges perform unreliably, to pattern matching over concrete criteria (Appendix B.1). The margin objective (Eq. 6) rewards any criterion that widens the judge's score gap between preferred and dispreferred responses, which favors concrete criteria over abstract ones. This enrichment progresses over time. Criteria length grows monotonically (59 to 112 characters on average) while criteria count remains stable at $\sim$3–4, and in some domains rubrics briefly decompose constraints into many fine-grained items before reconsolidating them into fewer, denser criteria (Appendix B.2).

3.3 Generalization

A rubric generator has practical value only if it generalizes beyond the training distribution. The self-evolving process optimizes rubrics for a specific policy and judge; we now test whether it nonetheless captures transferable evaluation principles across unseen domains, policy architectures, and judges. Appendix C examines whether training against several judges improves this transfer.

3.3.1 Alignment to Expert Deep Research Rubrics

EvoLM is trained on the Tulu 3 dataset (Lambert et al., 2025), which contains general-purpose tasks (Section 3.1). Deep research tasks that require long-form, multi-step responses are entirely unseen during rubric generator training. We test whether the learned rubric generator nonetheless produces evaluation criteria that align with expert human rubrics on these out-of-distribution tasks.

We evaluate on HealthBench (Arora et al., 2025) and ResearchQA (Yifei et al., 2026), two benchmarks that provide expert-written rubrics for each question. For each question, we generate rollouts from DR Tulu-8B SFT and RL checkpoints (Shao et al., 2026a) to ensure response quality diversity, then obtain ground-truth pairwise rankings by grading rollouts against the human expert rubrics using GPT-4.1. We generate rubrics using EvoLM, prompted Qwen3-8B, and prompted GPT-4.1, then score rollouts using Qwen3-1.7B to obtain pairwise rankings. We report pairwise ranking accuracy and Acc@δ, restricted to pairs where the expert-rubric score gap exceeds threshold δ to filter near-ties where the ground-truth ranking is ambiguous.

As shown in Table 2, EvoLM achieves the highest pairwise ranking accuracy on both benchmarks: 58.4% on HealthBench and 59.3% on ResearchQA, outperforming prompted Qwen3-8B (53.0%, 57.2%) and GPT-4.1 (52.5%, 51.0%) by 5–8 points. The advantage holds and widens on the filtered metrics, reaching 59.0% Acc@0.1 on HealthBench and 68.7% Acc@0.05 on ResearchQA. That a rubric generator trained on general-purpose tasks produces criteria better aligned with expert judg-

ment than GPT-4.1 on health and research domains provides strong evidence that the variational training objective learns transferable evaluation structure. This strong OOD performance, despite the disconnect between static preference benchmarks and downstream rubric utility observed in Section 3.2, further supports evaluating rubric generators on their ability to align with expert judgment rather than on held-out preference accuracy alone.

3.3.2 CO-EVOLVED RUBRICS PROVIDE EFFECTIVE TRAINING SIGNALS TO UNSEEN POLICIES

To test whether learned rubrics encode general evaluation criteria rather than policy-specific ones, we freeze the rubric generator from the main experiment and use it to train two unseen policies from scratch: Qwen3-4B (same family, smaller) and Llama-3.1-8B (different family). Each policy is also trained with GPT-4.1-prompted rubrics as a controlled comparison.

Table 2: Alignment to expert rubrics on OOD tasks.

	HealthBench		ResearchQA	
Rubric Generator	**Acc**	**Acc@0.1**	**Acc**	**Acc@0.05**
EvoLM	**58.4**	**59.0**	**59.3**	**68.7**
Qwen3-8B (prompted)	53.0	55.0	57.2	59.0
GPT-4.1 (prompted)	52.5	53.6	51.0	65.3

Table 3: Cross-model transfer. The rubric generator (frozen from the main Qwen3-8B experiment) is used to train policies of different sizes and families. Downstream quality measures whether the rubric provides effective reward signal for models not seen during rubric training. Rubric quality (RewardBench-2/JudgeBench) is not reported because the rubric generator is frozen. All scores are percentages.

	Downstream Policy Quality (OLMo3-Adapt)												
Configuration	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
Qwen3-4B + GPT-4.1 rubric	**94.8**	91.9	70.1	67.7	61.9	**82.8**	**37.0**	**26.5**	45.5	79.7	**84.6**	**30.5**	64.4
Qwen3-4B + EvoLM rubric	94.5	**92.4**	**77.9**	**68.0**	**65.3**	78.8	36.8	25.5	**49.3**	**79.9**	83.3	30.4	**65.2**
Llama-3.1-8B + GPT-4.1 rubric	73.0	43.7	**49.1**	**42.3**	**56.8**	58.8	67.5	25.8	33.7	11.3	**63.6**	**22.6**	45.7
Llama-3.1-8B + EvoLM rubric	**79.8**	**47.2**	40.5	38.9	56.0	**71.4**	**68.4**	**31.6**	**35.5**	**12.0**	63.4	18.4	**46.9**

As shown in Table 3, EvoLM rubrics outperform GPT-4.1-prompted rubrics on both architectures: Qwen3-4B (65.2% vs. 64.4%) and Llama-3.1-8B (46.9% vs. 45.7%). Within the Qwen3 family, the largest gains appear on code generation (HE+ 77.9% vs. 70.1%) and general reasoning (GPQA 49.3% vs. 45.5%, BBH 65.3% vs. 61.9%). These results confirm that the rubric generator captures transferable evaluation criteria that provide effective reward signal for models not seen during rubric training.

3.3.3 TRAINED RUBRICS GENERALIZE ACROSS JUDGE MODELS

We test whether trained rubrics remain effective when paired with judges not seen during training. We freeze the EvoLM rubric generator (trained with Qwen3-1.7B judge) and evaluate discriminative accuracy with three unseen inference-time judges: Qwen3-8B, OLMo-3-7B-Instruct, and Mistral-7B (Table 4).

With the Qwen3-8B judge, trained rubrics reach 62.4% on RewardBench-2 against 39.7% for prompted rubrics (+22.7), and 62.3% against 46.6% on JudgeBench Overall (+15.7). A more capable judge extracts more from the trained rubrics than from a generic prompt. Transfer across model families is smaller: OLMo-3-7B-Instruct gains +3.4 on RewardBench-2, and Mistral-7B gains +11.1 on JudgeBench while losing 1.4 on RewardBench-2. A rubric generator trained with one judge therefore serves other judges without retraining.

3.4 ABLATIONS

We ablate seven design dimensions: reward shaping, alternation frequency, model configuration, preference signal source, judge size, temporal step gap, and cross-architecture transfer. Table 17 summarizes them and Appendix D gives per-benchmark breakdowns. Downstream policy quality is remarkably robust to most choices. Alternation frequency, preference signal, and step gap all yield policies within 1–2 points of each other, suggesting that the self-evolving dynamic itself, rather than any particular configuration, is the primary driver. Format validity is the clearest requirement: at step 1000, only 1% of main-prompt rubrics without the format reward satisfy the schema, compared with 100% with the format reward. The 1.7B judge gives the highest downstream average even though larger judges achieve substantially higher static rubric accuracy. The single-model configuration halves memory and slightly improves the downstream average. Cross-architecture results are

Table 4: Cross-judge evaluation. RewardBench-2 and JudgeBench accuracy with different inference-time judges. The rubric generator was trained with Qwen3-1.7B. Bold indicates the higher value within each prompted/trained pair.

Inference Judge	Rubric	RewardBench2							JudgeBench					
		Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
Qwen3-1.7B (training judge)	Prompted	27.8	26.5	38.1	25.0	30.2	9.7	26.2	34.4	38.8	30.4	42.9	36.0	36.6
	Trained	**37.2**	**30.6**	**54.0**	**43.5**	**54.4**	**56.1**	**46.0**	**45.5**	**39.8**	30.4	**61.9**	**43.4**	**44.4**
Qwen3-8B	Prompted	37.4	33.6	51.5	49.9	45.5	20.3	39.7	48.1	33.7	53.6	61.9	46.6	49.3
	Trained	**49.9**	**41.7**	**74.3**	**55.3**	**71.0**	**82.2**	**62.4**	**57.1**	**68.4**	**66.1**	61.9	**62.3**	**63.4**
OLMo-3-7B-Instruct	Prompted	36.2	24.7	**57.7**	**48.7**	**43.2**	32.5	40.5	40.9	**32.7**	**37.5**	40.5	38.0	37.9
	Trained	**37.9**	**26.4**	54.0	48.0	39.9	**57.6**	**43.9**	**42.9**	31.6	26.8	**59.5**	**39.1**	**40.2**
Mistral-7B	Prompted	27.8	22.9	**28.6**	**38.5**	28.5	**34.2**	**30.1**	31.2	17.3	28.6	11.9	24.6	22.2
	Trained	**29.4**	**25.4**	28.5	31.9	**35.0**	22.1	28.7	**36.4**	**34.7**	**33.9**	**38.1**	**35.7**	**35.8**

mixed: OLMo co-evolution matches its prompted baseline downstream while Llama co-evolution underperforms it, despite competitive rubric accuracy. Within each controlled ablation that holds model architecture fixed, the variant with the highest RewardBench-2 does not produce the best policy, reinforcing that static rubric discrimination and downstream training utility capture different properties.

4 Related Work

During post-training, rewards have predominantly come from either trained reward models that generate scalar scores (Christiano et al., 2017; Ouyang et al., 2022) or prompts LMs to judge responses (Zheng et al., 2023; Dubois et al., 2023; Bai et al., 2022; Lee et al., 2024). Both encode evaluation criteria implicitly or via fixed prompts, whereas our rubric generator produces explicit criteria learned from preference signal. Self-improvement methods let models generate their own training signal. Self-Rewarding Language Models (Yuan et al., 2024) judge with a fixed evaluation prompt, SPIN (Chen et al., 2024) distinguishes model outputs from human references and so needs external data, Meta-Rewarding (Wu et al., 2025) adds a meta-judge over the model's own judgments, and SAO (Yin et al., 2026) generates prompts, responses and preference labels for preference optimization. In all four, evaluation criteria stay implicit in model weights. EvoLM makes evaluation the learned object, co-evolving explicit rubrics with the policy under a discriminative-utility objective and a frozen judge, which yields reward structure that transfers to unseen policies and judges. Concurrent work applies rubrics to RL training. Prompt-based approaches include DR Tulu (Shao et al., 2026a), RaR (Gunjal et al., 2026), RRD (Shen et al., 2026), OpenRS (Jia et al., 2026), and Dineen et al. (2025); DR Tulu also targets open-ended deep research but maintains and edits a fixed set of text rubrics instead of training a generator, and we use its rollouts as an out-of-distribution test bed (Section 3.3.1). Among trained rubric generators, RLCER (Sheng et al., 2026) requires verifiable correctness labels, Rubric-ARM (Xu et al., 2026) reference preference labels, RLAC (Wu et al., 2026) captures a single failure point, and RIFL (He et al., 2026) relies on human annotation. Temporal contrast pairs current and earlier policy responses, so EvoLM needs neither labels nor verifiers.

5 Conclusion

We introduce EvoLM, a post-training method that co-evolves explicit discriminative rubrics with the policy they score. Formalizing rubric quality as discriminative utility and optimizing it end-to-end structures the model's evaluative knowledge into criteria that a small frozen judge can apply reliably, without human annotations, proprietary APIs, or task-specific verifiers. EvoLM outperforms GPT-4.1-prompted rubrics on both rubric quality and downstream policy performance. External reward sources bring practical constraints: human judgment is costly and difficult to scale, while high held-out preference accuracy does not guarantee useful reward during policy optimization. A reward signal that co-evolves with the policy addresses both constraints. The learned evaluation structure transfers across unseen policies, judges, and out-of-distribution domains without retraining, and the variational framework extends to any domain where preferences can be observed but ground-truth answers do not exist.

LIMITATIONS

EVOLM has been validated on general-purpose post-training data; behavior on domain-specialized mixtures such as medicine or law is an open question. The rubric enrichment mechanism is most clearly observed in tasks with verifiable intermediate steps, and its effect on purely subjective evaluation criteria is less characterized. Freezing the judge bounds the complexity of the criteria the rubric generator can learn to produce. Finally, this paper instantiates the latent reward representation as natural-language rubrics. The same variational framing could in principle apply to non-textual structures such as executable code tests or formal specifications, where execution replaces LM judging, but we do not evaluate those representations here.

ACKNOWLEDGMENT

This research was developed in part with funding from the Defense Advanced Research Projects Agency's (DARPA) SciFy program (Agreement No. HR00112520300). The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This research was supported by the Meta AIM program, Coefficient Giving, and Amazon Health. This work was supported by the Singapore National Research Foundation and the National AI Group in the Singapore Ministry of Digital Development and Information under the AI Visiting Professorship Programme (award number AIVP-2024-001) and the AI2050 program at Schmidt Sciences.

REFERENCES

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

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback. arXiv preprint arXiv:2212.08073, 2022. URL https://arxiv.org/abs/2212.08073.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. doi: 10.2307/2334029. URL https://doi.org/10.2307/2334029.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6621–6642, 2024. URL https://proceedings.mlr.press/v235/chen24j.html.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Jacob Dineen, Aswin Rrv, Qin Liu, Zhikun Xu, Xiao Ye, Ming Shen, Zhaonan Li, Shijie Lu, Chitta Baral, Muhao Chen, and Ben Zhou. QA-LIGN: Aligning LLMs through constitutionally decomposed QA. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 20619–20642, 2025. doi: 10.18653/v1/2025.findings-emnlp.1123. URL https://aclanthology.org/2025.findings-emnlp.1123/.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S. Liang, and Tatsunori B. Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 30039–30069, 2023. doi: 10.52202/075280-1308. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5fc47800ee5b30b8777fdd30abcaaf3b-Abstract-Conference.html.

Shangbin Feng, Wenhao Yu, Yike Wang, Hongming Zhang, Yulia Tsvetkov, and Dong Yu. Don't throw away your pretrained model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1TTeOEufHz.

Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971. doi: 10.1037/h0031619. URL https://doi.org/10.1037/h0031619.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866, 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5020–5048, 2025. doi: 10.18653/v1/2025.findings-naacl.282. URL https://aclanthology.org/2025.findings-naacl.282/.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c1bTcrDmt4.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G. Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Kumar Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Hassan Awadalla, and Manaal Faruqui. AdvancedIF: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18003–18022, 2026. doi: 10.18653/v1/2026.acl-long.820. URL https://aclanthology.org/2026.acl-long.820/.

Shengyi Huang, Jiayi Weng, Rujikorn Charakorn, Min Lin, Zhongwen Xu, and Santiago Ontañón. Cleanba: A reproducible and efficient distributed reinforcement learning platform. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Diq6urt3lS.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. Unpacking DPO and PPO: Disentangling best practices for learning from preference feedback. In *Advances in Neural Information Processing Systems*, volume 37, pp. 36602–36633, 2024. doi: 10.52202/079017-1154. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/404df2480b6eef0486a1679e371894b0-Abstract-Conference.html.

Ruipeng Jia, Yunyi Yang, Wen Wang, Yuxin Wu, Yongbo Gai, Siyuan Tao, Mengyu Zhou, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Open rubric system: Scaling reinforcement learning with pairwise adaptive rubric. arXiv preprint arXiv:2602.14069, 2026. URL https://arxiv.org/abs/2602.14069.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8euJaTveKw.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6114.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26874–26901, 2024. URL https://proceedings.mlr.press/v235/lee24t.html.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, and Yang Liu. Skywork-Reward-V2: Scaling preference data curation via human-AI synergy. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ofgxkMLqic.

Changze Lv, Jie Zhou, Wentao Zhao, Jingwen Xu, Shihan Dou, Zisu Huang, Muzhao Tian, Xiaohua Wang, Zhengkang Guo, Yang Liu, Pluto Zhou, Tao Gui, Le Tian, Xiao Zhou, Xiaoqing Zheng, Xuanjing Huang, and Jie Zhou. Learning query-specific rubrics from human preferences for DeepResearch report generation. arXiv preprint arXiv:2602.03619, 2026. URL https://arxiv.org/abs/2602.03619.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

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 *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=97NEP1pyS3.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in RLVR. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=tqTNOpkP5j.

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. URL https://arxiv.org/abs/2402.03300.

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. URL https://arxiv.org/abs/2602.05125.

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

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*, volume 33, pp. 3008–3021, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/1f89885d556929e98d3ef9b86448f951-Abstract.html.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Yike Wang, Faeze Brahman, Shangbin Feng, Teng Xiao, Hannaneh Hajishirzi, and Yulia Tsvetkov. Small reward models via backward inference. arXiv preprint arXiv:2602.13551, 2026. URL https://arxiv.org/abs/2602.13551.

Mian Wu, Gavin Zhang, Sewon Min, Sergey Levine, and Aviral Kumar. RLAC: Reinforcement learning with adversarial critic for free-form generation tasks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dBmjnRR1bC.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E. Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11537–11554, 2025. doi: 10.18653/v1/2025.emnlp-main.583. URL https://aclanthology.org/2025.emnlp-main.583/.

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 *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=3SDg8dtkS8.

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

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*, 14:1344–1368, 2026. doi: 10.1162/tacl.a.732. URL https://doi.org/10.1162/tacl.a.732.

Shangjian Yin, Zhepei Wei, Xinyu Zhu, Wei-Lin Chen, and Yu Meng. Aligning large language models via fully self-synthetic data. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34553–34568, 2026. doi: 10.18653/v1/2026.acl-long.1595. URL https://aclanthology.org/2026.acl-long.1595/.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923, 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

A Implementation Details

A.1 Model Configuration

We use Qwen3-8B (8.2B parameters) as both the policy and rubric generator in a single-model configuration, where the two roles share weights and are distinguished only by their input prompts. The frozen rubric judge is Qwen3-1.7B by default, with judge size ablated from 0.6B to 32B in Section D.5. The alternating frequency is set to $K = 50$ steps per training phase, meaning the policy trains for 50 GRPO steps before switching to 50 steps of rubric generator training. Both the single-model versus two-model configuration and the alternating frequency are ablated in Sections D.3 and D.2, respectively.

A.2 Training Hyperparameters

Parameter	Value
Unique prompts per step	64
Samples per prompt	8
Effective batch size	**512 responses/step**
Per-device batch size	1
Gradient accumulation steps	1

Table 5: Batch configuration. With 64 training GPUs, each GPU processes 8 packed sequences per step.

Batch Configuration.

Parameter	Value
Learning rate	1×10^{-6}
LR scheduler	Constant
Warmup steps	0
Optimizer	AdamW (fused)
KL coefficient (β)	0.001
PPO clip range (ϵ)	0.2
KL estimator	KL3

Table 6: Optimization hyperparameters.

Optimization.

Parameter	Value
Response length (max)	16,384 tokens
Prompt length (max)	2,048 tokens
Pack length	18,500 tokens
Temperature (policy)	1.0
Temperature (question inference)	0.3
Temperature (rubric judge)	0.0 (greedy)

Table 7: Generation parameters. Long response length supports chain-of-thought reasoning.

Generation.

A.3 Format Reward Schema

The format reward $R_{\text{format}}(r) \in \{0, 1\}$ checks whether the generated rubric is well-formed JSON that the judge can reliably interpret. Specifically, the output must parse as a JSON object containing a

Table 8: Design-space comparison of rubric-based methods. Our method is the only one that trains the rubric generator, co-evolves it with the policy, and requires no proprietary API or external labels.

Method	Trains rubric generator	No proprietary API	No external labels	Not restricted to verifiable domains	Co-evolves with policy
RAR	✗	✗	✓	✓	✗
RRD	✗	✗	✓	✓	✗
RLCER	✓	✓	✗	✗	✓
RUBRIC-ARM	✓	✓	✗	✓	✗
EVOLM (OURS)	✓	✓	✓	✓	✓

`criteria` array of at least two entries, each with a `criterion` description (string) and a positive `weight` (float), where weights sum to 1.0 ± 0.15. A rubric must satisfy this schema for the judge to apply per-criterion scoring.

A.4 ASYNC TRAINING AND ACTIVE SAMPLING

Asynchronous Pipeline. We use `async_steps`$= 4$, meaning 4 batches (256 prompts total) are in-flight at any time. The trainer processes batch t while the generator produces batches $t + 1$ through $t + 4$. This overlaps generation and training to maximize GPU utilization. The policy used for generation can be up to 4 steps behind the current training policy, following the Cleanba paradigm (Huang et al., 2024).

Active Sampling. GRPO requires reward variance within each prompt group to compute meaningful advantages. When all 8 samples for a prompt receive identical scores (zero variance), that batch provides no learning signal. With `active_sampling` enabled, such zero-variance batches are filtered and the generator continues sampling until obtaining 64 prompts with non-zero reward variance. This ensures every training step uses informative data.

A.5 HARDWARE CONFIGURATION

- **Training**: 8 nodes $\times$ 8 H100 GPUs = 64 GPUs
- **DeepSpeed**: ZeRO Stage 3 with gradient checkpointing
- **vLLM Engines**: 56 total (40 policy + 16 judge, shared in single-model mode)
- **Tensor Parallelism**: 1 (single GPU per engine)

A.6 BASELINE CONFIGURATIONS

All baselines use the same hardware allocation (8 nodes $\times$ 8 H100 GPUs) and Qwen3-8B as the policy model. We describe the key design choices for each below. Table 8 highlights the key design-space differences.

Prompted, sequential, and scalar baselines. The two prompted baselines and the sequential stage-2 runs share the same fixed-rubric policy training loop; they differ only in the source of the frozen rubric generator (GPT-4.1 API, base Qwen3-8B, or a stage-1 sequential checkpoint). GPT-4.1 and prompted Qwen3-8B use the generic 5–10-criterion prompt in Appendix E.3; the main dealbreaker prompt in Appendix E.1 is reserved for our trained rubric generator. In sequential stage 1, the policy is frozen while the rubric generator trains on preference pairs from either the inferred-question or rubric-conditioned method. In stage 2, the rubric generator is frozen at the stage-1 checkpoint and the policy trains with temporal contrast signal. The scalar reward-model baseline replaces rubric-based evaluation entirely with a Skywork Bradley-Terry reward model trained on the same preference data.

RewardBench-2 scoring. For each RewardBench-2 item, the rubric generator receives the question and candidate responses and produces one prompt-level rubric. The Qwen3-1.7B judge scores each candidate independently against the complete rubric in a single call, applying the criterion weights to produce a scalar score in $[0, 1]$. The candidate with the highest score is selected, with fractional credit when the preferred response is tied for the maximum. We report the unweighted

mean across the six RewardBench-2 subsets. This whole-rubric protocol preserves criterion context and weights; it does not split generated rubrics into separately judged text fragments.

RaR and RRD. Our RAR implementation uses the implicit-judge variant: GPT-4.1 generates instance-specific rubrics for each prompt and a local Qwen3-1.7B judge assigns the holistic Likert score used as reward. The original RAR method (Gunjal et al., 2026) uses proprietary models for both rubric generation and judging; our re-implementation replaces the proprietary judge with Qwen3-1.7B for controlled comparison. Our RRD implementation uses the whitened-uniform (RRD-WU) variant: GPT-4.1 proposes and iteratively refines rubrics, Qwen3-1.7B scores each rubric item, and item scores are combined with whitened-uniform weighting. Both methods train only the policy; rubric generation is handled by the frozen GPT-4.1 proposer with no rubric-generator gradient updates.

RLCER. We use the evolving variant on DAPO-Math-17k: the shared Qwen3-8B model serves as both policy and rubric generator trained in alternation with single-step phases, Qwen3-1.7B judges rubric satisfaction, and the reward combines correctness-filtered chain-of-thought with outcome verification. In the rubric phase, cached rubric generations and verifier results from the preceding policy step are reused rather than resampled.

Rubric-ARM. We use the pretrained rubric generator and judge released by Xu et al. (2026), which were trained on labeled preference pairs via their alternating RL procedure. Both models are kept frozen; only the Qwen3-8B policy is trained against this fixed reward source. This baseline tests whether their learned rewarding artifacts transfer to an effective signal our training framework, rather than re-running their full training algorithm.

Multi-judge ablations. These ablations replace the single Qwen3-1.7B judge with a five-judge ensemble: Qwen3-1.7B, Llama-3.2-1B-Instruct, OLMo-2-0425-1B-Instruct, Gemma-3-1B-IT, and Qwen3-4B. We compare three aggregation rules with these five models.

A.7 PER-STEP COMPUTE ACCOUNTING

We report online compute per update in terms of P prompt groups, N rollouts per prompt, R rubric items per prompt, and J judges in an ensemble. In all experiments, P=64 and N=8. We separately count policy generations, rubric generations, and reward-side evaluations. Offline costs (e.g., pretraining a scalar reward model) are excluded. Table 9 provides a compact summary of per-prompt reward cost for each baseline method.

Table 9: Per-prompt reward computation cost for each method. N is the number of rollout responses per prompt and R is the number of rubric items.

Method	Rubric construction	Answer scoring	Extra checks
RAR	1 rubric proposal	N answer scores	–
RRD	initial proposal + recursive refinement	$N \times R$ item-level scores	data-dependent refinement checks
RLCER	N rubric generations (one per rollout)	N^2 judge calls	N verifier checks
RUBRIC-ARM	1 rubric generation (frozen model)	N answer scores	–
OURS	1 rubric generation	N answer scores	–

For RRD, recursive refinement expands and filters rubric candidates until convergence, so its construction cost is data-dependent. For our method, one prompt-level rubric is reused across all N responses.

A.8 TEMPORAL CONTRAST CONFIGURATION

- **Buffer size**: 2,048 experiences
- **Age gap**: $[20, 100]$ policy steps (default)
- **Sampling**: Uniform random within valid age range

The age gap ensures rejected samples come from a sufficiently different policy version (at least 20 steps old) while avoiding excessively stale samples (at most 100 steps old). Each experience stores (step, question, answer), and step gap is logged for analysis.

A.9 CHECKPOINTING

- **Save frequency**: Every 25 steps (2 per alternating phase)
- **Retention**: All checkpoints kept (no deletion)
- **Format**: HuggingFace-compatible model weights

A.10 TEMPORAL CONTRAST SANITY CHECK

Temporal contrast treats later policy responses as preferred to earlier responses for the same prompt. This is an average-case assumption, not a claim that every later response is better than every earlier response. To check whether the aggregate direction is meaningful in the main run, we evaluate intermediate policy checkpoints on the same OLMo3-Adapt suite used in the main results. Table 10 shows steady aggregate improvement from step 50 to step 500, with the largest gains on HumanEval+ and AlpacaEval. This pattern is inconsistent with temporal contrast being dominated by reversals or self-confirming noise.

Table 10: Intermediate checkpoint evaluation for the main co-evolving run. Scores are percentages on OLMo3-Adapt benchmarks. The average improves from 67.3% at step 50 to 69.2% at step 500.

Step	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
50	95.4	93.6	80.2	68.7	63.8	82.8	38.1	29.7	53.8	82.6	84.8	34.4	67.3
200	95.3	93.4	80.7	68.6	63.0	82.2	37.7	29.9	53.8	82.2	85.2	37.8	67.5
350	95.7	93.9	85.3	69.0	64.6	82.1	37.7	29.6	53.6	82.6	84.5	42.1	68.4
500	95.8	92.2	86.2	68.5	67.6	85.3	37.7	32.1	54.2	82.3	86.4	42.2	69.2

Several individual benchmarks are flat or slightly non-monotonic, which is expected for online RL and reinforces that temporal contrast is noisy at the example level. The method relies on the aggregate direction being useful; the checkpoint trajectory above and the step-gap ablation in Appendix D.6 support that condition.

A.11 OUTPUT STABILITY ANALYSIS

We analyze saved policy generations for three visible failure modes that can emerge during reward optimization: exact repetition, prompt copying, and length collapse. Exact repetition is measured using the repository's n-gram/block repetition filter. Prompt copying is measured as the presence of any 20-token contiguous span from the prompt in the generated text. Because Qwen-style outputs include explicit reasoning traces that often restate the user request, repetition and copy rates are computed after removing the reasoning trace.

Table 11: Policy generation diagnostics on 100 fixed held-out prompts across training. "Words" and TTR are measured on the full completion including reasoning traces; repetition and copy rates are answer-only after stripping reasoning traces. Generation remains stable across training, with no systematic growth in repetition, prompt copying, or short templated outputs.

Step	N	Mean words	Mean TTR	Repetition (%)	Prompt copy (%)
50	100	1,507	0.314	3.0	11.0
150	100	1,621	0.300	1.0	10.0
250	100	1,659	0.298	1.0	11.0
350	100	1,592	0.299	1.0	10.0
450	100	1,320	0.324	0.0	11.0
550	100	1,392	0.322	0.0	10.0
650	100	1,356	0.324	1.0	12.0
750	100	1,292	0.335	0.0	9.0
850	100	1,277	0.326	0.0	10.0
950	100	1,455	0.314	2.0	9.0
Training logs	21,183	1,620	0.310	0.6	8.7

Across held-out checkpoints, mean completion length stays within 1,277–1,659 words and lexical diversity stays within 0.298–0.335 type-token ratio. The exact-repetition rate remains 0–3% on held-out prompts and 0.6% in 21,183 saved training generations. Answer-level prompt-copy overlap

is also stable rather than increasing, falling from 11.0% at step 50 to 9.0% at step 950. These diagnostics show no systematic increase in repetition or prompt copying and no collapse to short templates.

B RUBRIC EXAMPLES

We compare rubrics generated by three sources for the same questions: prompted Qwen3-8B (base model, no training), prompted GPT-4.1, and our trained co-evolving rubric generator (step 1000 of the V3 margin+format experiment). The trained generator produces fewer but substantially longer criteria (3.0 criteria averaging 112 characters each, vs. 5.0 criteria at 57 characters for Qwen3-8B), with 19.3% of criteria embedding specific expected values (vs. 6.9% for Qwen3-8B and 7.5% for GPT-4.1) and only 0.3% label-only criteria (vs. 21.9% for Qwen3-8B). Aggregate statistics are in Section B.3. Below, we show representative examples that illustrate the qualitative differences and the mechanism by which trained rubrics simplify evaluation for a small judge.

B.1 RUBRIC COMPARISON: PROMPTED VS. TRAINED

Example 1: Math Problem

Question: The perimeter of a rectangle is 48. What is the largest possible area of the rectangle?

Prompted Qwen3-8B (5 criteria, equal weights):

0.20 Correctly applies the perimeter formula to relate length and width.
0.20 Correctly expresses the area as a function of one variable.
0.20 Correctly uses a method to find the maximum area (e.g., completing the square, calculus).
0.20 Correctly calculates the maximum area.
0.20 Explains that the maximum area occurs when the rectangle is a square.

Trained rubric (3 criteria, concentrated weights):

0.80 The answer is the correct maximum area of 144, derived from the given perimeter of 48.
0.15 Uses the perimeter and area formulas correctly in the solution.
0.05 Provides a logical explanation or steps to arrive at the solution.

The trained rubric takes a fundamentally different approach: rather than listing five generic steps for the judge to verify independently, it concentrates 80% of the weight on a single criterion that embeds the expected answer (144) and its derivation context (perimeter of 48). This transforms evaluation from "verify this optimization proof" into "check if the response contains 144 derived from the perimeter constraint," a task a 1.7B judge performs reliably. The prompted rubric requires the judge to independently assess whether each method step is "correct," which demands mathematical understanding the small judge may lack.

The trained rubric is also more concise: 3 criteria at 71 characters each vs. 5 criteria at 65 characters each. The enrichment strategy packs the essential verification into fewer, more information-dense criteria rather than decomposing into many fine-grained checks (the V2 prompt variant produces the latter pattern; see Section B.3).

Example 2: Constrained Writing

Question: Create a summary discussing the impact of artificial intelligence in software development, tailored for a software developer audience. The summary should consist of exactly three paragraphs, each separated by "—". Each paragraph should begin with a title wrapped in square brackets. Include the keyword "innovation" at least twice in each paragraph. Do not include the words "obsolete" or "manual" in any paragraph.

Prompted Qwen3-8B (6 criteria):

0.20 Structure and Formatting.
0.20 Keyword Frequency.
0.10 Exclusion of Forbidden Words.
0.20 Audience Relevance.
0.15 Content Depth and Accuracy.

0.15 Flow and Coherence.

Trained rubric (3 criteria):

0.40 The summary has exactly three paragraphs, each starting with a title in square brackets and separated by "—", and adheres to the specified formatting requirements.
0.40 Each paragraph contains the keyword "innovation" at least twice and avoids the words "obsolete" and "manual".
0.20 The content addresses the impact of AI in software development, focusing on innovation, efficiency, and developer-centric benefits.

The prompted rubric uses abstract labels ("Keyword Frequency," "Exclusion of Forbidden Words") that require the judge to infer what keyword, what frequency, and which words. The trained rubric consolidates all formatting and keyword constraints into two concrete criteria (0.40 weight each) that spell out the exact checks: "three paragraphs. . . separated by '—'," "keyword 'innovation' at least twice," "avoids 'obsolete' and 'manual'." Each criterion is self-contained and verifiable by pattern matching. The content criterion (0.20) is deliberately down-weighted, reflecting that constraint satisfaction is more reliably checkable than content quality for a small judge.

B.2 Rubric Evolution Over Training

To illustrate how rubrics evolve during co-evolving training, we show how the rubric for a fixed question changes across training steps in the V3 margin+format experiment. In contrast to the V2 prompt variant (which evolves by increasing criteria count), the V3 variant evolves by *enriching* each criterion—packing more specific, verifiable detail into fewer items.

Math problem. *Question:* The perimeter of a rectangle is 48. What is the largest possible area?

Step	#	Representative criteria
100	3	Top-heavy: "Provides the correct maximum area (144)" (0.80), "Correctly applies the perimeter formula ($2(L + W) = 48$)" (0.10), "Demonstrates understanding of optimization principles" (0.10).
300	5	"Provides the correct maximum area (144) for the given perimeter" (0.80). New meta-criteria: "Organizes the solution in a coherent step-by-step format" (0.03), "Uses appropriate mathematical notation and terminology" (0.02).
500	3	"The answer is the correct maximum area (144) and is equivalent to the square of half the perimeter divided by 2" (0.80). Criterion now embeds the derivation relationship alongside the answer.
700	4	"Uses a valid method (e.g., calculus, algebra) to derive the answer" (0.10). "Explains why a square maximizes the area for a given perimeter" (0.05). "Presents the answer as a numerical value, not a formula" (0.05).
1000	3	"The answer is the correct maximum area of 144, derived from the given perimeter of 48" (0.80). "Uses the perimeter and area formulas correctly in the solution" (0.15).

The answer (144) is already embedded at step 100, but the top criterion evolves from a bare answer check to one that links the answer to its derivation context. By step 1000, the criterion text is 89 characters (vs. 39 at step 100), encoding both the expected value and its relationship to the given perimeter. The weight remains concentrated at 0.80, reflecting the rubric generator's learned strategy: for verifiable math problems, a single answer-embedding criterion with high weight is the most reliable signal for the frozen judge.

This progression suggests a mechanism for the co-evolving advantage: the rubric generator discovers that embedding the expected answer produces higher reward from the frozen judge, and progressively enriches the top criterion with derivation context rather than splitting into many fine-grained checks.

We now show this same enrichment phenomenon across three additional domains—emotional support, scientific explanation, and constrained opinion writing—to demonstrate that rubric evolution is not domain-specific.

Emotional support. *Question:* I'm constantly overwhelmed by a sense of dread, can you help me understand this feeling?

Step	#	Representative criteria
100	4	"Accurately explains the nature of dread, including its psychological and emotional components" (0.40). "Provides clear, empathetic language accessible to someone feeling overwhelmed" (0.30). "Offers meaningful insights into potential causes or coping strategies" (0.10).
300	5	"Addresses the core experience of overwhelming dread and its potential causes (e.g., stress, anxiety, burnout)" (0.40). "Provides accurate, evidence-based information about the psychological or physiological basis of dread" (0.30).
500	5	"Connects the feeling to psychological or situational factors (e.g., anxiety, perfectionism, burnout)" (0.25). "Avoids pathologizing the feeling while acknowledging potential mental health implications" (0.05).
700	4	"Explains the nature of dread as a mix of anxiety and overwhelm, often linked to stress or fear of the unknown" (0.40). "Describes physical or emotional manifestations (e.g., restlessness, racing thoughts, chest tightness)" (0.20).
1000	3	"Addresses the feeling of dread and provides actionable strategies for coping (e.g., grounding techniques, self-compassion, or seeking professional help)" (0.60). "Identifies potential causes or contributing factors to the sense of dread (e.g., anxiety, stress, or past experiences)" (0.30). "Maintains an empathetic, non-judgmental, and supportive tone throughout the response" (0.10).

Despite the subjective nature of emotional support, the rubric evolves from four generic criteria to three information-dense ones. The top criterion at step 1000 (0.60 weight, 158 characters) consolidates what were separate concerns—coping strategies, grounding techniques, professional help—into a single comprehensive check. The rubric also learns to specify *examples* within criteria ("e.g., grounding techniques, self-compassion"), converting an abstract "offers coping strategies" into a concrete checklist.

Scientific explanation. *Question:* Explain, using scientific language and examples, the multistep process of blood coagulation, including the roles of platelets and clotting factors, and elaborate on the significance of this process in wound healing…

Step	#	Representative criteria
100	5	Label-style: "Accuracy of the multistep blood coagulation process" (0.40), "Clarity of platelet and clotting factor roles" (0.20), "Depth of significance in wound healing" (0.15), "Detail in intricate mechanisms" (0.15), "Analysis of disruptions and adverse effects" (0.10).
500	5	"Accurately describes the multistep blood coagulation process, including platelet activation, clotting factors (e.g., fibrinogen, thrombin), and the cascade mechanism" (0.40). "Details intricate mechanisms (e.g., intrinsic/extrinsic pathways, fibrin clot formation, enzymatic reactions) with scientific precision" (0.20).
1000	3	"Comprehensively explains the multistep process of blood coagulation, including the roles of platelets and clotting factors, with accurate scientific terminology and clear sequence of events" (0.60). "Elaborates on the significance of coagulation in wound healing, linking it to preventing excessive blood loss and tissue repair" (0.25). "Analyzes disruptions or alterations in coagulation and their adverse effects on healing (e.g., hemophilia, thrombosis, anticoagulant use)" (0.15).

The evolution from label-only criteria ("Accuracy of the multistep blood coagulation process") to self-contained descriptions is particularly clear here. At step 100, the 41-character labels require the judge to independently determine what constitutes "accuracy" or "depth." By step 1000, each criterion is 150+ characters long and specifies exactly what content to look for: platelet roles, clotting factors, cascade mechanism, specific medical conditions.

Constrained opinion writing. *Question:* Write a 500-word article that discusses the impact of increasing restrictions on religious activity from the perspective of an Austrian Muslim woman. The article should include three sections titled `<<Personal Experiences>>`, `<<Community Impact>>`, and `<<Future Concerns>>`. The first section should contain at least two placeholders for personal anecdotes, the second at least three placeholders for quotes from community leaders, and the third at least two placeholders for hypothetical scenarios.

Step	#	Representative criteria
100	4	Label-style with coarse weights: "Structure and placeholders: Three sections with correct titles and required placeholders" (0.40). "Perspective and cultural sensitivity: Maintains the viewpoint of an Austrian Muslim woman" (0.30). "Coherence and flow" (0.20). "Word count adherence: Exactly 500 words" (0.10).
500	6	Decomposition phase: constraints split into individual criteria—"Includes two distinct personal anecdotes in the `<<Personal Experiences>>` section" (0.10). "Incorporates three quotes from community leaders in the `<<Community Impact>>` section" (0.10). "Presents two hypothetical scenarios in the `<<Future Concerns>>` section" (0.10).
1000	3	Reconsolidation: "The article is structured with three distinct sections (Personal Experiences, Community Impact, Future Concerns), each containing the required placeholders (two anecdotes, three quotes, two hypothetical scenarios)" (0.60). "The Personal Experiences section includes two relevant anecdotes that reflect the perspective of an Austrian Muslim woman and are specific to the topic of increasing restrictions" (0.25). "The Community Impact section includes three quotes from community leaders that address the impact of increasing restrictions on religious activity and are relevant to the Austrian Muslim community" (0.15).

This example shows a distinctive three-phase evolution: at step 100, the rubric uses coarse labels (272 total characters). At step 500, it decomposes into 6 criteria with each constraint getting its own check (572 total characters). By step 1000, it *reconsolidates* into 3 criteria but retains all the specificity—the top criterion (0.60, 218 characters) now packs the structure requirements, section titles, and placeholder counts into a single comprehensive verification. The total rubric text at step 1000 (590 characters) exceeds step 500 (572 characters) despite having half as many criteria, confirming that enrichment rather than decomposition is the dominant V3 strategy.

Across all four domains (math, emotional support, science, constrained writing), the same pattern emerges: early rubrics use short labels or generic checks, while late rubrics pack specific, verifiable expectations into each criterion. The V3 variant achieves this through enrichment (longer criteria, similar count) rather than decomposition (more criteria, shorter each). Both strategies serve the same function: reducing the interpretive burden on the frozen judge by making evaluation criteria self-contained and independently checkable.

B.3 Aggregate Rubric Statistics

To move beyond individual examples, we compute aggregate statistics over all 100 evaluation prompts, comparing the co-evolving method against prompted baselines (Qwen3-8B, GPT-4.1), a static rubric baseline (frozen prompted rubric, never trained), and RLCER (sequential rubric training). We report two types of metrics: *structural* metrics describing the rubric's shape (criteria count, weight distribution) and *content* metrics describing what the criteria say (specificity, constraint checking, label usage).

For content metrics, we classify each criterion by regex into categories: *specific value* (embeds numbers, formulas, or exact strings), *negation* ("avoids," "does not," "without"), *example* (contains "e.g.," "such as," "for example"), and *label-only* (criterion text ≤ 40 characters, indicating an abstract label rather than a concrete check). We also classify criteria by *type*: *correctness* ("correct," "accurate"), *constraint* ("avoid," "exclude," "keyword," "must not"), and *completeness* ("include," "cover," "address").

Cross-method comparison. Table 12 compares all methods at their final training step. Reported values are averages over 100 evaluation prompts.

Table 12: Aggregate rubric statistics at final training step. Co-evolving methods with the alternate prompt and main prompt produce rubrics that are markedly more specific, constraint-aware, and descriptive than all baselines. The alternate prompt evolves via decomposition (more criteria, flatter weights); the main prompt evolves via enrichment (fewer but longer criteria). Both dramatically reduce label-only criteria and increase specificity.

	Structure		Weights		Content (%)		Type (%)	
Method	#Crit	Len	MaxW	Top-1%	Spec.	Label	Constr.	Compl.
Qwen3-8B prompted	5.0	57	0.34	33.6	6.9	21.9	7.7	6.5
GPT-4.1 prompted	5.9	73	0.29	29.0	7.5	3.6	8.5	9.7
Static rubric	3.5	37	0.68	67.5	4.6	64.9	2.9	3.5
RLCER	4.6	54	0.37	37.4	4.5	28.1	7.3	6.8
Alt Prompt Co-evolve	**11.4**	73	**0.12**	**14.8**	7.3	5.2	17.2	12.9
Main Prompt Co-evolve	3.0	**112**	0.64	64.7	**19.3**	**0.3**	**20.3**	**14.6**

Several patterns stand out:

- **Label elimination.** Label-only criteria (short abstract labels like "Structure and Section Compliance") drop from 21.9% (Qwen3-8B) to 0.3% in main-prompt co-evolve. The static rubric baseline moves in the *opposite* direction, reaching 64.9%—without training signal, the model regresses to vague labels. RLCER evolving also increases its label rate (15.3% $\rightarrow$ 28.1%), indicating that its sequential rubric training does not produce the same enrichment effect.
- **Specificity.** Criteria embedding specific expected values (numbers, formulas, code patterns) increase from 6.9% (Qwen3-8B base) to 19.3% in main-prompt co-evolve—nearly 3$\times$ the prompted baseline rate. Neither RLCER (4.5%) nor the static baseline (4.6%) show any improvement.
- **Constraint awareness.** Criteria dedicated to checking explicit constraints ("avoids," "excludes," "keyword") rise from 7.7% (Qwen3-8B) to 17.2–20.3% in co-evolving methods. This shift reflects the rubric learning to decompose prompt requirements into individually verifiable checks.
- **Two evolution strategies.** Alternate-prompt co-evolve and main-prompt co-evolve achieve similar functional outcomes through different structural strategies. The alternate prompt *decomposes*: criteria count increases from 4.9 to 11.4 with weights flattening (top-1 share: 32.5% $\rightarrow$ 14.8%). The main prompt *enriches*: criteria count stays at $\sim$3 but each criterion nearly doubles in length (59 $\rightarrow$ 112 characters), packing more detail into fewer items. Both strategies reduce label-only criteria and increase specificity, constraint checking, and completeness coverage.

Evolution trajectory. Tables 13 and 14 show how rubric statistics evolve during training for alternate-prompt and main-prompt co-evolving, respectively, with metrics computed every 100 training steps.

The trajectories reveal several noteworthy patterns:

- **Phase transition in the alternate prompt.** Alternate-prompt co-evolve shows a sharp phase transition around step 700–800, where criteria count jumps from $\sim$7 to $\sim$11 and the example rate triples (5.6% $\rightarrow$ 12.7%). This aligns with the point where the policy has improved enough that response pairs from temporal contrast become difficult to distinguish without fine-grained criteria.
- **Monotonic enrichment in the main prompt.** Main-prompt co-evolve shows steady, monotonic change: criterion length increases from 59 to 112 characters across training, specificity rises from 6.9% to 19.3%, and label-only rate drops from 23.5% to 0.3%. The enrichment is gradual rather than abrupt, suggesting that the main rubric prompt format (which allows fewer, longer criteria) induces a different optimization landscape than the alternate prompt.

Table 13: Alternate-prompt co-evolve rubric evolution trajectory. Criteria count grows sharply after step 700 (phase transition); label-only rate drops monotonically; constraint and completeness criteria steadily increase.

	Structure		Weights		Content (%)				Type (%)		
Step	#Crit	Len	MaxW	Top-1%	Spec.	Neg.	Eg.	Label	Corr.	Cnst.	Cmpl.
100	4.9	62	0.33	32.5	5.1	8.7	7.0	13.0	26.0	9.8	6.6
200	4.9	62	0.34	33.6	6.4	7.5	4.0	13.4	22.4	7.9	6.8
300	4.8	66	0.30	35.9	4.3	10.4	1.9	10.8	19.5	10.8	5.6
400	5.2	67	0.28	31.4	5.1	11.1	2.8	7.9	17.6	11.5	7.1
500	5.7	69	0.26	29.7	4.6	9.9	3.9	5.9	15.6	10.1	9.9
600	6.4	70	0.23	26.4	6.3	13.1	5.2	4.2	12.5	13.5	7.8
700	6.9	64	0.20	24.2	6.0	13.5	5.6	5.7	9.0	13.7	10.0
800	8.6	70	0.16	20.2	7.4	17.2	12.7	6.2	10.1	17.9	10.3
900	11.7	73	0.15	18.2	10.5	17.2	15.2	1.4	7.7	17.2	13.0
1000	11.4	73	0.12	14.8	7.3	17.1	16.5	5.2	9.7	17.2	12.9

Table 14: Main-prompt co-evolve rubric evolution trajectory. Criteria count remains stable ($\sim$3–4) while criterion text length grows from 59 to 112 characters. Label-only rate drops from 23.5% to 0.3%; specificity, negation, and example rates all roughly triple.

	Structure		Weights		Content (%)				Type (%)		
Step	#Crit	Len	MaxW	Top-1%	Spec.	Neg.	Eg.	Label	Corr.	Cnst.	Cmpl.
100	3.7	59	0.59	59.1	6.9	6.1	4.1	23.5	25.1	7.5	5.8
200	3.9	71	0.60	60.0	7.1	8.9	5.8	12.3	24.6	10.2	8.6
300	3.9	77	0.61	60.4	8.8	11.6	8.3	9.0	22.2	12.7	11.1
400	4.1	78	0.58	57.6	8.0	13.3	12.1	8.5	16.9	14.3	7.7
500	4.0	84	0.58	58.3	9.6	14.4	13.2	3.0	23.8	15.9	9.6
600	4.1	86	0.51	51.0	14.3	16.0	16.3	1.7	23.2	17.2	12.1
700	4.0	90	0.51	52.1	12.3	21.6	12.6	0.8	16.8	22.4	12.6
800	3.7	87	0.52	52.9	13.2	22.2	15.4	1.4	15.4	23.0	12.4
900	2.8	104	0.65	65.6	16.4	17.1	20.3	0.7	19.9	18.5	14.2
1000	3.0	112	0.64	64.7	19.3	19.7	20.0	0.3	23.1	20.3	14.6

- **Correctness criterion decline.** In the alternate prompt, the proportion of criteria checking "correctness" drops from 26.0% to 9.7% as training progresses. This does not mean correctness is ignored; rather, the general "is this correct?" criterion is replaced by specific checks ("calculates area correctly as 144") that subsume correctness within a targeted verification.
- **Baseline stagnation.** Neither the static rubric nor RLCER Evolving shows meaningful movement on any content metric across training. RLCER's label-only rate *increases* from 15.3% to 28.1%, and its specificity stays flat at 4.5–7.5%. This confirms that the co-evolving training dynamics—specifically, the pressure from temporal contrast and the feedback loop between policy and rubric—are necessary for rubric improvement.

C Multi-Judge Training

To produce rubric generators that generalize out of the box to unseen judges, we train with multiple judges simultaneously. We use a five-judge ensemble of small models: Qwen3-1.7B, Llama-3.2-1B-Instruct, OLMo-2-0425-1B-Instruct, Gemma-3-1B-IT, and Qwen3-4B. Each judge $j \in \{1, \ldots, J\}$ independently scores the accepted and rejected answers, a^+ and a^-, using the generated rubric, obtaining score_j^+ and score_j^-, and casts a binary vote $v_j = \mathbf{1}[\text{score}_j^+ > \text{score}_j^-]$. The reward for each rollout is

$$R_{\text{MJ}} = w_m \cdot \bar{m} + w_f \cdot \mathbf{1}[\text{valid_format}] + w_\kappa \cdot \hat{\kappa}, \tag{9}$$

where $\bar{m} = \frac{1}{J}\sum_{j=1}^{J}(s_j^+ - s_j^-)$ is the average margin (difference between accepted and rejected scores across judges), $\mathbf{1}[\text{valid_format}]$ is a binary indicator for whether the generated rubric satisfies the required JSON schema, and $\hat{\kappa}$ is the **agreement among the judges**, calculated as $\hat{\kappa} = \text{clip}(\kappa_F(\vec{v}), 0, 1)$, where κ_F is Fleiss's kappa (Fleiss, 1971) computed over the binary vote from

Table 15: Cross-judge evaluation of single-judge and multi-judge rubric generators. RewardBench-2 and JudgeBench accuracy (%) with three inference-time judges. The single-judge generator (EvoLM) was trained with Qwen3-1.7B; multi-judge generators were trained with five judges (Qwen3-1.7B, Llama-3.2-1B-Instruct, OLMo-2-0425-1B-Instruct, Gemma-3-1B-IT, Qwen3-4B). Multi-judge uses Eq. 10; MF and MAF use Eq. 9 with weights $(0.7, 0.3, 0)$ and $(0.5, 0.3, 0.2)$, respectively.

Inference Judge	Rubric Generator	RewardBench2							JudgeBench					
		Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
Qwen3-1.7B	Prompted	27.8	26.5	38.1	25.0	30.2	9.7	26.2	34.4	38.8	30.4	42.9	36.0	36.6
	EvoLM (single judge)	37.2	**30.6**	54.0	43.5	**54.4**	56.1	46.0	**45.5**	39.8	30.4	**61.9**	43.4	44.4
	MF (Eq. 9)	**37.4**	29.4	**56.5**	**50.3**	49.8	53.8	**46.2**	**48.7**	45.9	42.9	59.5	**48.3**	49.3
	MAF (Eq. 9)	33.4	28.4	53.4	46.8	50.9	**57.7**	45.1	42.9	42.9	**44.6**	52.4	44.3	45.7
	BA (Eq. 10)	35.5	30.0	55.4	47.1	50.1	56.2	45.7	45.5	**49.0**	42.9	**64.3**	**48.3**	**50.4**
Qwen3-8B	Prompted	37.4	33.6	51.5	49.9	45.5	20.3	39.7	48.1	33.7	53.6	61.9	46.6	49.3
	EvoLM (single judge)	49.9	**41.7**	74.3	55.3	**71.0**	82.2	62.4	57.1	**68.4**	**66.1**	61.9	**62.3**	**63.4**
	MF (Eq. 9)	51.0	36.0	74.2	**59.8**	69.7	76.5	61.2	57.1	62.2	64.3	66.7	60.9	62.6
	MAF (Eq. 9)	55.9	35.5	77.6	56.2	70.5	75.8	61.9	**57.8**	**71.4**	**66.1**	**69.0**	**64.3**	**66.1**
	BA (Eq. 10)	**57.6**	36.1	**78.5**	52.5	70.9	**85.6**	**63.5**	57.1	69.4	64.3	61.9	62.3	63.2
OLMo-3-7B	Prompted	36.2	24.7	**57.7**	48.7	**43.2**	32.5	40.5	40.9	32.7	37.5	40.5	38.0	37.9
	EvoLM (single judge)	37.9	26.4	54.0	48.0	39.9	**57.6**	43.9	**42.9**	31.6	26.8	**59.5**	**39.1**	**40.2**
	MF (Eq. 9)	32.3	22.3	49.8	51.8	35.4	38.0	38.3	40.3	32.6	**48.2**	30.9	38.3	38.0
	MAF (Eq. 9)	33.0	25.7	54.7	50.3	35.7	37.6	39.5	39.0	29.6	25.0	30.9	33.1	31.1
	BA (Eq. 10)	**38.7**	**27.6**	56.3	**52.7**	40.3	54.7	**45.1**	**53.2**	**41.8**	37.5	**52.4**	**47.4**	**46.2**

the judges of whether the accepted answer received a higher score than the rejected answer. We test two weight configurations: **Margin+Format** (MF) with $(w_m, w_f, w_\kappa) = (0.7, 0.3, 0)$, which omits the agreement penalty entirely, and **Margin+Agreement+Format** (MAF) with $(w_m, w_f, w_\kappa) = (0.5, 0.3, 0.2)$, which retains a moderate agreement incentive. Additionally, we consider the alternative reward design of using the binary vote and the agreement (BA) without the margin and format rewards:

$$R = \bar{v} - (1 - \hat{\kappa}), \quad \bar{v} = \frac{1}{J}\sum_{j=1}^{J} v_j, \quad \hat{\kappa} = \text{clip}(\kappa_F, 0, 1), \tag{10}$$

where $\bar{v}$ is the average vote (fraction of judges preferring the accepted answer). All multi-judge variants maintain comparable downstream policy quality to the single-judge baseline (Table 16).

The key advantage of multi-judge training appears at inference time. Table 15 evaluates all multi-judge rubric generators against the single-judge generator (EvoLM) and prompted rubrics, using three inference-time judges not in the training mix: Qwen3-1.7B (the single-judge training judge), Qwen3-8B, and OLMo-3-7B-Instruct. The multi-judge variants usually improve JudgeBench and remain competitive on RewardBench-2, with the best aggregation rule depending on the inference judge. On Qwen3-8B, MAF achieves 66.1% JudgeBench, compared to 63.4% for the single-judge generator, while remaining close on RewardBench-2 (61.9% vs. 62.4%). The improvement is even more pronounced on OLMo-3-7B-Instruct, where BA raises JudgeBench from 40.2% to 46.2% (+6.0). On the training judge Qwen3-1.7B, all multi-judge variants improve JudgeBench substantially (45.7–50.4% vs. 44.4%). These results suggest that training with diverse judges can produce rubrics that are more broadly interpretable, though the benefit is stronger for cross-judge robustness than for every static preference benchmark. Table 16 reports rubric quality and downstream policy quality for each strategy, evaluated with the Qwen3-1.7B training judge.

Table 16: Multi-judge aggregation strategies. Rubric quality (RB2 and JB) and downstream policy quality with Qwen3-1.7B as the evaluation judge. MF and MAF use Eq. 9 with weights $(0.7, 0.3, 0)$ and $(0.5, 0.3, 0.2)$, respectively. All scores are percentages.

Aggregation	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
Single judge (baseline)	37.2	**30.6**	54.0	43.5	**54.4**	56.1	46.0	**45.5**	39.8	30.4	**61.9**	43.4	44.4
Score averaging	**37.5**	28.0	50.7	38.4	43.1	30.0	37.9	29.2	27.6	25.0	35.7	28.9	29.4
Majority voting	34.2	26.0	50.2	46.2	49.4	50.3	42.7	46.8	35.7	42.9	50.0	43.4	43.8
BA (Eq. 10)	35.5	30.0	55.4	47.1	50.1	56.2	45.7	45.5	**49.0**	42.9	**64.3**	**48.3**	**50.4**
MF (Eq. 9)	37.4	29.4	**56.5**	**50.3**	49.8	53.8	**46.2**	**48.7**	45.9	42.9	59.5	**48.3**	49.3
MAF (Eq. 9)	33.4	28.4	53.4	46.8	50.9	**57.7**	45.1	42.9	42.9	**44.6**	52.4	44.3	45.7

	Downstream Policy Quality (OLMo3-Adapt)												
Aggregation	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
Single judge (baseline)	95.8	92.2	**86.2**	**68.5**	**67.6**	85.3	37.7	32.1	54.2	82.3	86.4	42.2	**69.2**
Score averaging	95.2	**93.4**	84.5	59.8	59.5	82.2	**39.4**	31.3	54.2	80.5	85.0	**42.3**	67.3
Majority voting	**95.9**	92.4	74.0	63.5	57.0	**85.5**	36.4	32.3	54.0	82.4	87.2	33.4	66.2
BA (Eq. 10)	95.4	93.1	77.1	67.3	55.9	85.2	37.2	32.2	**57.1**	82.7	87.6	41.3	67.7
MF (Eq. 9)	95.3	92.3	79.0	64.7	57.2	84.6	36.0	32.3	53.1	82.8	87.6	31.6	66.4
MAF (Eq. 9)	95.8	93.3	79.6	51.2	59.2	83.3	36.2	**33.8**	56.5	**83.0**	**88.3**	33.7	66.2

The single-judge baseline produces the strongest downstream policy (69.2%), outperforming all multi-judge strategies (66.2–67.7%). Adding judges does not improve training signal quality, likely

Table 17: Summary of ablation results. Each row varies one design dimension while holding others at the main configuration (⋆). Rubric = RewardBench-2 average. Policy = OLMo3-Adapt 12-benchmark average (%).

Dimension	Variant	RB2	Policy
Reward design	5–10 criteria, no-dealbreaker prompt	38.9	67.6
	main prompt	37.9	**69.5**
	main prompt + margin	37.2	66.9
	main prompt + margin + format⋆	**46.0**	69.2
Alt. freq. K	2	46.4	67.9
	10	44.5	68.5
	20	46.9	68.6
	50⋆	46.0	**69.2**
	100	**48.5**	68.7
Model config.	Single-model⋆	46.0	**69.2**
	Two-model	**48.4**	68.9
Pref. signal	Inferred question	48.1	68.5
	Rubric-conditioned	48.4	68.5
	Temporal contrast⋆	46.0	**69.2**
	Combined (IQ + RC)	45.6	68.9
	Combined (all 3)	**49.3**	67.8
Judge size	0.6B	22.1	67.9
	1.7B⋆	46.0	**69.2**
	4B	61.6	66.5
	8B	59.5	67.1
	14B	**67.6**	69.0
	32B	66.5	67.6
Step gap	$[5, 20]$	**48.8**	68.6
	$[20, 100]$⋆	46.0	**69.2**
	$[100, 300]$	47.1	68.9
Cross-arch.	Qwen3-8B⋆	**46.0**	**69.2**
	OLMo-3-7B	45.3	63.9
	Llama-3.1-8B	34.0	42.2

because the additional judges introduce calibration mismatches that dilute the discriminative signal. This is most visible with score averaging (37.9% RB2), where averaging across differently calibrated judges destroys rank ordering. Among multi-judge variants, the Binary Agreement reward (BA) achieves the highest JudgeBench accuracy (50.4%) and the best downstream quality (67.7%), while MF achieves the highest RB2 (46.2%). The cross-judge transfer benefits of multi-judge generators are analyzed in Section 3.3 (Table 15).

D Detailed Ablation Results

This appendix provides full per-benchmark breakdowns for each ablation summarized in Section 3.4. All experiments use the co-evolving training setting unless otherwise noted. Rubric quality is measured by RewardBench 2 (RB2) and JudgeBench (JB) discriminative accuracy. Downstream policy quality is measured on the twelve-benchmark OLMo3-Adapt suite. All scores are percentages.

D.1 Reward Shaping

The rubric reward design has three dimensions: the rubric generation prompt (5–10 criteria, no-dealbreaker prompt vs. main prompt, which adds dealbreaker criteria and flexible scoring levels), the reward type (binary 0/1 vs. continuous margin), and the format reward (absent vs. 0.3 weight). We ablate these cumulatively, using the co-evolving setting with temporal contrast and $K = 50$. Format% denotes the fraction of generated rubrics satisfying the R_{format} schema (Appendix A.3), measured post-hoc at training step 1000. The exact texts of the 5–10 criteria, no-dealbreaker prompt and the main rubric-generation prompt are given in Appendix E.3 and Appendix E.1, respectively.

Table 18: Reward shaping ablation (full results). All rows use Qwen3-8B single-model, Qwen3-1.7B judge, temporal contrast gap $[20, 100]$.

	RewardBench2							JudgeBench						
Variant	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg	Format%
5–10 criteria, no-dealbreaker prompt	30.1	30.3	48.9	42.0	45.2	37.0	38.9	29.2	34.7	16.1	31.0	28.9	27.7	30
main prompt	32.8	30.3	44.8	41.9	46.0	31.7	37.9	37.0	40.8	**46.4**	**61.9**	42.6	**46.5**	1
main prompt + margin	30.3	24.7	50.1	30.4	50.9	36.8	37.2	42.9	**41.8**	26.8	50.0	40.9	40.4	63
main prompt + margin + format	**37.2**	**30.6**	**54.0**	**43.5**	**54.4**	**56.1**	**46.0**	**45.5**	39.8	30.4	**61.9**	**43.4**	44.4	**100**

	Downstream Policy Quality (OLMo3-Adapt)												
Variant	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
5–10 criteria, no-dealbreaker prompt	95.7	**92.9**	71.9	68.1	66.6	84.2	36.6	32.4	**55.6**	82.1	**87.4**	38.2	67.6
main prompt	**95.8**	92.5	**88.8**	**68.7**	**71.0**	85.1	**38.6**	31.2	52.5	81.8	86.1	42.0	**69.5**
main prompt + margin	**95.8**	92.5	74.8	68.6	61.5	**85.4**	37.9	**32.5**	50.7	81.4	87.0	34.5	66.9
main prompt + margin + format	**95.8**	92.2	86.2	68.5	67.6	85.3	37.7	32.1	54.2	**82.3**	86.4	**42.2**	69.2

On the matched 100-prompt audit at step 1000, format validity is 30% with the 5–10-criterion prompt, 1% with the main prompt and binary reward, 63% after adding the margin reward, and 100% after adding the format reward. The margin reward alone retains valid structure more often than the binary main-prompt objective, but does not make formatting reliable. Adding the margin reward alone also lowers downstream performance on AlpacaEval (34.5) and BBH (61.5); adding the format reward (weight 0.3) recovers both (42.2 and 67.6, respectively) while achieving the highest rubric quality (46.0% RB2). The main run's training log reports 99.41% valid rubrics at step 1000, consistent with the 100/100 post-hoc audit.

D.2 Alternating Frequency

In co-evolving training, the alternating frequency K controls how many steps each component trains before switching. Frequent switching (K small) tightly couples the models, allowing each to adapt to the other's changes, but may introduce noise if neither converges before switching. Infrequent switching (K large) allows each component to converge but causes distribution shift at transitions. We test $K \in \{2, 10, 20, 50, 100\}$ with temporal contrast preference signal.

Table 19: Effect of alternating frequency K in co-evolving training (full results).

	RewardBench2							JudgeBench					
K	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
2	34.6	30.7	49.3	48.1	53.5	61.9	46.4	37.7	**51.0**	33.9	54.8	42.9	44.3
10	32.8	**31.3**	50.9	41.5	48.4	61.9	44.5	42.2	46.9	**44.6**	47.6	**44.6**	**45.4**
20	33.9	29.9	55.0	**49.7**	50.8	62.3	46.9	**46.8**	36.7	42.9	54.8	44.3	45.3
50 (main)	**37.2**	30.6	54.0	43.5	**54.4**	56.1	46.0	45.5	39.8	30.4	**61.9**	43.4	44.4
100	37.0	29.2	**57.3**	46.3	50.6	**70.5**	**48.5**	41.6	31.6	35.7	**61.9**	40.3	42.7

	Downstream Policy Quality (OLMo3-Adapt)												
K	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
2	95.5	92.6	77.9	63.8	65.1	84.3	37.7	31.8	56.5	84.4	88.0	37.7	67.9
10	95.4	91.1	81.3	68.0	67.4	86.1	36.4	**34.1**	53.3	81.3	**88.0**	40.2	68.5
20	**95.8**	92.1	84.6	68.6	65.1	84.9	36.8	32.3	**56.9**	82.5	87.6	35.6	68.6
50 (main)	**95.8**	92.2	**86.2**	68.5	67.6	85.3	**37.7**	32.1	54.2	82.3	86.4	**42.2**	**69.2**
100	**95.8**	**93.6**	83.5	**68.8**	65.5	85.2	37.0	32.3	53.8	**84.5**	86.7	37.3	68.7

K=50 achieves the best downstream average (69.2%), with performance declining at both extremes. Very tight coupling (K=2, avg 67.9%) provides insufficient convergence time within each phase, while very loose coupling (K=100, avg 68.7%) allows the rubric generator to train on an increasingly stale policy distribution. Notably, K=100 achieves the highest RB2 (48.5%) despite lower downstream performance, consistent with the disconnect between held-out discriminative accuracy and training signal effectiveness observed throughout our experiments. The rubric generator converges more fully at large K, producing rubrics that generalize better to held-out benchmarks, but the resulting reward landscape is less effective for on-policy training because the rubrics do not adapt to the current policy distribution.

D.3 Model Configuration

Single-model configuration shares parameters between policy and rubric generator, reducing memory but potentially introducing self-evaluation bias. Two-model configuration uses separate Qwen3-

8B instances. The two-model configuration is evaluated at step 950 (the closest policy checkpoint available) due to the different training schedule under parameter separation.

Table 20: Single-model versus two-model configuration (full results).

	RewardBench2							JudgeBench					
Configuration	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
Single-model	37.2	**30.6**	54.0	43.5	54.4	56.1	46.0	**45.5**	**39.8**	30.4	**61.9**	43.4	44.4
Two-model	**40.3**	29.9	**56.7**	**50.9**	**56.2**	**56.4**	**48.4**	42.2	38.8	**44.6**	57.1	43.4	**45.7**

	Downstream Policy Quality (OLMo3-Adapt)												
Configuration	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
Single-model	**95.8**	92.2	**86.2**	68.5	67.6	85.3	**37.7**	**32.1**	54.2	82.3	86.4	**42.2**	**69.2**
Two-model	95.5	**93.3**	79.5	**68.9**	**67.8**	**85.8**	37.5	31.6	**54.9**	**84.1**	**86.9**	40.6	68.9

Two-model configuration improves RB2 by 2.4 points (48.4% vs. 46.0%), suggesting that single-model mode introduces a mild self-evaluation bias: when policy and rubric generator share parameters, the model may produce rubrics whose criteria its own responses naturally satisfy, inflating apparent rubric quality on the training distribution without improving generalization to held-out benchmarks. However, downstream policy quality is similar, with the single-model configuration ahead by 0.3 points (69.2% vs. 68.9%). The per-benchmark breakdown shows complementary strengths: single-model leads on HumanEval+ (86.2% vs. 79.5%) while two-model leads on Zebra (84.1% vs. 82.3%) and GPQA (54.9% vs. 54.2%). In practice, the single-model configuration is preferred because it halves memory requirements while slightly improving the downstream average.

D.4 Preference Signal Source

The three preference pair construction methods capture different aspects of quality. Temporal contrast captures general improvement over training, inferred question captures question-addressing, and rubric-conditioned captures rubric utility. We test each signal alone and in combination. Combined configurations mix signals equally during training.

Table 21: Comparison of preference signal sources (full results).

	RewardBench2							JudgeBench					
Method	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
Inferred question	**37.4**	25.4	**66.4**	46.1	57.9	55.2	48.1	**53.9**	40.8	39.3	40.5	**46.3**	43.6
Rubric-conditioned	36.4	**35.3**	52.4	**48.2**	54.4	**63.4**	48.4	42.9	40.8	39.3	54.8	43.1	44.4
Temporal contrast (main)	37.2	30.6	54.0	43.5	54.4	56.1	46.0	45.5	39.8	30.4	**61.9**	43.4	44.4
Combined (IQ + RC)	33.2	32.0	54.3	44.5	54.4	55.1	45.6	41.6	**48.0**	37.5	**61.9**	45.1	**47.2**
Combined (all 3)	36.5	27.8	64.7	46.5	**58.7**	61.8	**49.3**	40.9	31.6	**46.4**	50.0	40.3	42.2

	Downstream Policy Quality (OLMo3-Adapt)												
Method	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
Inferred question	95.1	**93.5**	80.3	68.9	66.0	**85.8**	36.6	**32.7**	**56.5**	**84.0**	87.2	35.5	68.5
Rubric-conditioned	95.6	92.7	82.8	68.0	65.0	85.4	36.8	32.6	54.0	83.4	86.5	39.0	68.5
Temporal contrast (main)	**95.8**	92.2	**86.2**	68.5	**67.6**	85.3	**37.7**	32.1	54.2	82.3	86.4	**42.2**	**69.2**
Combined (IQ + RC)	**95.8**	93.4	83.2	**69.1**	64.9	85.7	36.6	31.5	55.1	82.9	86.9	41.3	68.9
Combined (all 3)	95.7	93.3	80.7	67.6	63.0	85.1	35.1	31.8	54.9	81.9	**87.5**	37.4	67.8

Temporal contrast achieves the lowest RB2 (46.0%) but the highest downstream average (69.2%) among single-signal variants, while the combined-all-3 configuration achieves the highest RB2 (49.3%) but the lowest downstream (67.8%). This again illustrates the disconnect between held-out discriminative accuracy and training signal effectiveness. Temporal contrast's downstream advantage is concentrated on code generation (HumanEval+ 86.2% vs. 80.3–82.8% for other single signals) and open-ended generation (AlpacaEval 42.2% vs. 35.5–39.0%). Its advantage likely stems from its temporal structure, which provides a natural curriculum of gradually harder preference pairs as the policy improves, with both responses generated by the same model at different capability levels. Inferred question and rubric-conditioned signals produce identical downstream averages (68.5%) but with different per-benchmark profiles, suggesting they capture complementary aspects of quality. Overall, temporal contrast alone works best for downstream policy training; combining multiple signal types does not improve over individual signals.

D.5 JUDGE SIZE

Larger judges provide more accurate evaluation but increase compute cost. We ablate judge size from 0.6B to 32B parameters using Qwen3 models of the indicated size as the frozen judge, with Qwen3-8B as the policy and rubric generator in single-model configuration.

Table 22: Effect of judge model size (full results). All judges use Qwen3 models of the indicated size. The 1.7B judge is the main configuration.

	RewardBench2							JudgeBench					
Judge	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
0.6B	27.6	22.0	29.0	25.9	26.9	1.4	22.1	35.1	28.6	32.1	45.2	34.0	35.3
1.7B (main)	37.2	30.6	54.0	43.5	54.4	56.1	46.0	45.5	39.8	30.4	61.9	43.4	44.4
4B	46.4	39.8	77.6	**57.9**	69.6	78.4	61.6	50.6	57.1	57.1	64.3	55.1	57.3
8B	52.1	37.3	73.6	55.5	72.2	66.6	59.5	53.2	60.2	58.9	**69.0**	58.0	60.4
14B	55.7	**46.3**	81.6	55.8	**78.6**	**87.7**	**67.6**	**57.8**	58.2	**78.6**	66.7	**62.3**	**65.3**
32B	**57.8**	41.0	**83.0**	57.3	78.0	81.8	66.5	56.5	**63.3**	66.1	**69.0**	61.4	63.7

	Downstream Policy Quality (OLMo3-Adapt)												
Judge	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
0.6B	95.8	92.8	78.8	**69.1**	61.4	85.4	**38.8**	32.1	54.9	81.8	87.6	36.2	67.9
1.7B (main)	95.8	92.2	**86.2**	68.5	67.6	85.3	37.7	32.1	54.2	82.3	86.4	**42.2**	**69.2**
4B	**95.9**	91.8	72.3	58.1	67.1	82.8	34.8	33.3	53.8	82.8	87.7	37.0	66.5
8B	95.7	92.1	72.9	62.1	**68.8**	82.4	35.3	32.9	**56.7**	83.6	87.6	35.0	67.1
14B	**95.9**	**93.6**	85.1	66.7	68.2	**85.7**	35.7	32.1	56.0	84.1	88.1	36.4	69.0
32B	95.8	91.6	76.6	56.5	67.0	82.2	35.5	**34.0**	**56.9**	**84.4**	**88.8**	41.8	67.6

Larger judges improve held-out rubric accuracy substantially through 14B: RB2 rises from 46.0% with the 1.7B judge to 67.6% with the 14B judge, while the 32B judge remains similar at 66.5%. JudgeBench follows the same broad pattern, rising from 34.0% with the 0.6B judge to 62.3% with the 14B judge before a small decline at 32B. Yet the 1.7B judge produces the best downstream policy (69.2%), outperforming all larger judges (66.5–69.0%). Thus, stronger held-out discrimination from a larger judge does not imply a better training signal in these runs. One possible explanation is that larger judges are more tolerant of vague or underspecified criteria, weakening the selection pressure on the rubric generator to produce precise, verifiable rubrics. The 0.6B judge is too small to apply rubrics reliably (RB2 22.1%), but still achieves 67.9% downstream, only 1.3 points below the main configuration, further underscoring the robustness of the co-evolving framework.

D.6 TEMPORAL CONTRAST STEP GAP

The step gap $[g_{\min}, g_{\max}]$ specifies that rejected responses are sampled uniformly from policy rollouts $g_{\min}$ to $g_{\max}$ steps in the past. Small gaps pair similar responses from nearby training steps, providing noisy signal. Large gaps pair responses from substantially different policy versions, providing clearer quality differences but risking distribution shift.

Table 23: Effect of temporal contrast step gap (full results). The $[20, 100]$ gap is the main configuration.

	RewardBench2							JudgeBench					
Age Gap	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
$[5, 20]$	34.5	**36.1**	51.5	46.5	55.8	**68.4**	**48.8**	40.9	35.7	**44.6**	35.7	39.4	39.2
$[20, 100]$ (main)	**37.2**	30.6	54.0	43.5	54.4	56.1	46.0	**45.5**	**39.8**	30.4	61.9	**43.4**	44.4
$[100, 300]$	36.9	27.4	**56.8**	**47.3**	**56.5**	57.5	47.1	40.3	36.7	37.5	**64.3**	41.7	**44.7**

	Downstream Policy Quality (OLMo3-Adapt)												
Age Gap	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
$[5, 20]$	**96.1**	**92.5**	82.6	68.6	68.0	85.6	35.9	**32.8**	**56.0**	82.8	**87.3**	35.5	68.6
$[20, 100]$ (main)	95.8	92.2	**86.2**	68.5	67.6	85.3	**37.7**	32.1	54.2	82.3	86.4	**42.2**	**69.2**
$[100, 300]$	95.1	92.4	80.7	**69.2**	**68.8**	**85.7**	36.2	31.9	54.2	**85.3**	87.2	40.3	68.9

All three gap ranges produce competitive downstream policies (68.6–69.2%), confirming that temporal contrast is robust to this hyperparameter. The moderate $[20, 100]$ gap achieves the best downstream average, balancing contrast strength with distributional similarity between paired responses.

D.7 Cross-Architecture Generalization

We test whether the co-evolving framework generalizes across model architectures by running the full pipeline with Llama-3.1-8B and OLMo-3-7B. For each architecture, we compare co-evolving training against a prompted baseline (policy trains with a frozen base-model rubric generator). All experiments use the main prompt + margin + format reward and Qwen3-1.7B as the frozen judge.

Table 24: Cross-architecture generalization (full results). Co-evolving and prompted baselines on Llama-3.1-8B and OLMo-3-7B, compared against the Qwen3-8B main result.

		RewardBench2							JudgeBench					
Architecture	Training	Factuality	Precise IF	Math	Safety	Focus	Ties	Avg	Knowledge	Reasoning	Math	Coding	Overall	Avg
Qwen3-8B	Co-evolving (main)	37.2	30.6	54.0	43.5	54.4	56.1	46.0	45.5	39.8	30.4	61.9	43.4	44.4
Llama-3.1-8B	Prompted	**30.2**	**29.7**	**50.1**	**29.4**	**44.5**	34.4	**36.4**	**47.4**	40.8	**48.2**	47.6	**45.7**	46.0
	Co-evolving	29.2	24.1	43.2	28.4	36.9	**42.0**	34.0	42.9	**41.8**	46.4	**57.1**	44.9	**47.1**
OLMo-3-7B	Prompted	32.6	26.8	55.6	**36.6**	49.0	**60.4**	43.5	37.0	**41.8**	35.7	**50.0**	39.7	41.1
	Co-evolving	**34.9**	**34.9**	**58.6**	33.0	**51.4**	58.8	**45.3**	**42.9**	40.8	**41.1**	45.2	**42.3**	**42.5**

		Downstream Policy Quality (OLMo3-Adapt)												
Architecture	Training	GSM8K	MATH	HE+	MBPP+	BBH	MMLU	IFEval	PopQA	GPQA	Zebra	AGI-E	AlpacaE	Avg
Qwen3-8B	Co-evolving (main)	95.8	92.2	86.2	68.5	67.6	85.3	37.7	32.1	54.2	82.3	86.4	42.2	69.2
Llama-3.1-8B	Prompted	**81.3**	**46.4**	48.9	36.1	**52.6**	47.7	**56.2**	31.7	**31.7**	**13.1**	**63.6**	**16.3**	**43.8**
	Co-evolving	71.8	40.8	**55.8**	**49.5**	50.5	**67.7**	31.1	**35.3**	29.0	12.0	62.6	0.2	42.2
OLMo-3-7B	Prompted	95.1	92.9	**89.3**	**63.8**	68.2	**78.8**	33.5	**32.1**	**47.1**	**64.5**	**79.8**	**23.3**	**64.0**
	Co-evolving	**95.3**	**93.2**	**89.3**	62.8	**73.8**	78.6	**34.8**	31.2	43.5	62.7	78.6	22.9	63.9

On OLMo-3-7B, co-evolving training improves rubric quality over the prompted baseline (45.3% vs. 43.5% RB2; 42.5% vs. 41.1% JudgeBench) while the downstream policy averages are nearly identical (63.9% vs. 64.0%). The policy-level effects vary by task: co-evolution improves BBH (+5.6) and IFEval (+1.3), but trails on GPQA (-3.6) and Zebra Logic (-1.8). On Llama-3.1-8B, co-evolution produces higher JudgeBench accuracy (47.1% vs. 46.0%) but lower RB2 (34.0% vs. 36.4%) and downstream policy quality (42.2% vs. 43.8%); its 0.2% AlpacaEval score indicates severe generation collapse. These mixed outcomes show that the co-evolving objective can be applied across architectures, but its policy-level benefit is architecture-dependent and may require architecture-specific optimization.

E Prompts

E.1 Main Training Prompts

The current main training configuration uses the main rubric-generation prompt together with the standard policy and judge prompts. Concretely, this corresponds to the main prompt + margin + format setup used for the main result and the main ablation reruns. In each block below, the angle-bracket message labels are editorial; the prompt text itself is reproduced verbatim, with the system message followed by the user message.

Policy.

```
<System prompt>
You are a helpful assistant.

<User prompt>
{question}
```

Rubric generation.

```
<System prompt>
You are an expert evaluator generating rubrics to assess answers to
questions.

Given a question, first analyze it to identify:
```

```
- First identify the most important aspect of the question that the
answer should satisfy, this will be used to form the Dealbreaker
criterion (explained later).
- Explicit requirements: directly stated constraints, formatting rules,
or content directives (e.g., "list three reasons", "write in Python",
"under 100 words")
- Implicit requirements: unstated but necessary qualities inferred from
context (e.g., explaining "blockchain to grandparents" implicitly
requires avoiding jargon, even if not explicitly forbidden)

Then generate a rubric of 2-5 criteria following these rules:
1. Structural atomicity: each criterion targets exactly one aspect. Do
not combine multiple conditions into one criterion.
2. Semantic objectivity: write criteria based only on the question,
without assuming any specific answer.
3. All weights must sum to exactly 1.0, reflecting each criterion's
importance, important criteria such as accuracy should have higher
weight.
4. Dealbreaker criterion: some criteria are so important that if the
answer does not satisfy them, the answer is just not good enough. For
example, for questions with verifiable short form answers or multiple
choice (such as math or factuality), the dealbreaker criterion should
be "the answer is equivalent to XXX (e.g. 100)". A Dealbreaker
criterion should have very high weight such as 0.8. But form
explanation based or questions requiring long form answers, accuracy
should not have such a high weight because it's too general and it will
be hard for the judge to assess accuracy. It's fine if there's no
dealbreaker.

For each criterion should be described in a sentence, define scoring
levels from 0.0 to 1.0. At minimum include 1.0 and 0.0, and add
intermediate levels (e.g., 0.5, 0.3, 0.8) wherever useful for
distinguishing answer quality. A judge will score each criterion and
multiply by the weight to produce a total score.

Output ONLY valid JSON in this format:
{"criteria": [{"criterion": "<a sentence of what this criterion
measures>", "weight": <float>, "scoring_levels": {"1.0":
"<description>", "0.5": "<description>", "0.0": "<description>"}},
{"criterion": ...}, ...]}

Example | question: "Explain how photosynthesis works in simple terms"
```

```
{"criteria": [{"criterion": "Explains that plants convert sunlight into
chemical energy", "weight": 0.3, "scoring_levels": {"1.0": "Clearly
explains the sunlight-to-energy conversion", "0.7": "Mentions sunlight
or energy but not the conversion process", "0.3": "Vague reference to
energy without clear connection to sunlight", "0.0": "No mention of the
energy conversion mechanism"}}, {"criterion": "Identifies CO2 and water
as inputs and oxygen and glucose as outputs", "weight": 0.25,
"scoring_levels": {"1.0": "All four substances correctly identified",
"0.6": "Three substances correctly identified", "0.3": "Some inputs or
outputs mentioned but incomplete", "0.0": "None identified or
incorrect"}}, {"criterion": "Avoids unnecessary jargon and is
understandable to a general audience", "weight": 0.2, "scoring_levels":
{"1.0": "Clear and jargon-free throughout", "0.8": "Mostly accessible
with minimal technical terms that are explained", "0.4": "Mostly
accessible but uses some unexplained technical terms", "0.0": "Dense
with jargon, inaccessible to a general reader"}}, {"criterion":
"Follows a coherent structure from inputs to process to outputs",
"weight": 0.15, "scoring_levels": {"1.0": "Well-organized with clear
progression", "0.6": "Generally structured but minor organizational
issues", "0.2": "Some structure but jumps between ideas", "0.0":
"Disorganized or incoherent"}}, {"criterion": "All stated facts about
photosynthesis are correct", "weight": 0.1, "scoring_levels": {"1.0":
"No factual errors", "0.8": "Minor inaccuracy that does not undermine
the explanation", "0.0": "Contains significant factual errors"}}]}

<User prompt>
{question}
```

Judge.

```
<System prompt>
You are an expert evaluator judging answers based on a rubric.

<User template>
Question: {question}

Rubric: {rubric}

Answer to evaluate: {answer}

Evaluate the answer against the rubric. For each criterion, decide how
well the answer satisfies it (0.0 = not at all, 1.0 = fully), then
multiply by the criterion's weight. Sum the weighted scores to get the
total (must be between 0.0 and 1.0).

Output ONLY valid JSON:
{"reasoning": "<evaluate each criterion, give satisfaction * weight,
then sum>", "score": <float 0.0-1.0>}

Example 1 (rubric: Factual Accuracy 0.4, Completeness 0.35, Clarity
0.25):
{"reasoning": "Factual Accuracy (weight 0.4): answer is fully correct,
1.0 * 0.4 = 0.4. Completeness (weight 0.35): covers main points but
misses edge cases, 0.6 * 0.35 = 0.21. Clarity (weight 0.25): well
organized and easy to follow, 1.0 * 0.25 = 0.25. Total = 0.86",
"score": 0.86}
Example 2 (rubric: Correctness of Solution 0.5, Use of Examples 0.3,
Appropriate Detail 0.2):
{"reasoning": "Correctness of Solution (weight 0.5): correct approach
but has an arithmetic error in the final step, 0.8 * 0.5 = 0.4. Use of
Examples (weight 0.3): no examples provided, 0.0 * 0.3 = 0.0.
Appropriate Detail (weight 0.2): gives a brief answer without
elaboration, 0.2 * 0.2 = 0.04. Total = 0.44", "score": 0.44}
```

```
Your evaluation:
```

E.2 Preference Pair Construction Prompts

Section 2.4 describes three methods for constructing preference pairs. The current main run uses temporal contrast; we include the inferred-question and rubric-conditioned prompts here as well because they appear in the preference-signal ablations in Appendix D.4.

Temporal contrast. This method requires no special prompt. Current responses are generated using the standard policy system prompt ("You are a helpful assistant.") applied to the question. Earlier responses are retrieved from the stored rollout buffer indexed by training step.

Inferred question. The policy infers the intended question from a given response using the following prompt template:

```
<System prompt>
You are an expert at understanding what question someone was trying to
answer based on their response.

Given an answer, infer what question the person was likely trying to
answer. Be specific and output only the inferred question, nothing
else.

<User prompt>
Here is an answer that was written in response to some question:

{answer}

What question was this answer trying to respond to? Output only the
inferred question.
```

The dispreferred response is then generated by applying the standard policy prompt to the inferred question $\hat{q}$.

Rubric-conditioned. The preferred response is generated by prepending the rubric to the policy prompt, yielding the following template:

```
<System prompt>
You are a helpful assistant.

When answering, follow this rubric to ensure a high-quality response:

{rubric}

<User prompt>
{question}
```

The dispreferred response uses the standard system prompt without the rubric. The rubric is generated by the current rubric generator ρ_ϕ using the rubric-generation prompt corresponding to the training configuration: the main prompt is shown in Section E.1, and the 5–10 criteria, no-dealbreaker prompt used in the reward-shaping ablation is shown in Section E.3.

E.3 Generic Rubric Prompt Used by Prompted Baselines and Ablations

The GPT-4.1 and Qwen3-8B prompted baselines use the generic 5–10-criterion, no-dealbreaker prompt below. The reward-shaping ablation in Appendix D.1 also compares this prompt against the main prompt while keeping the policy and judge prompts fixed. The main prompt is shown in Section E.1.

Rubric generation (5–10 criteria, no-dealbreaker prompt).

```
<System prompt>
You are an expert evaluator generating rubrics to assess answers to
questions.

Given a question, first analyze it to identify:
- Explicit requirements: directly stated constraints, formatting rules,
or content directives (e.g., "list three reasons", "write in Python",
"under 100 words")
- Implicit requirements: unstated but necessary qualities inferred from
context (e.g., explaining "blockchain to grandparents" implicitly
requires avoiding jargon, even if not explicitly forbidden)

Then generate a rubric of 5-10 criteria following these rules:
1. Structural atomicity: each criterion targets exactly one aspect. Do
not combine multiple conditions into one criterion.
2. Semantic objectivity: write criteria based only on the question,
without assuming any specific answer.
3. All weights must sum to exactly 1.0, reflecting each criterion's
importance.

For each criterion should be described in a sentence, define scoring
levels from 0.0 to 1.0. At minimum include 1.0 and 0.0, and add
intermediate levels (e.g., 0.5, 0.3, 0.8) wherever useful for
distinguishing answer quality. A judge will score each criterion and
multiply by the weight to produce a total score.

Output ONLY valid JSON in this format:
{"criteria": [{"criterion": "<a sentence of what this criterion
measures>", "weight": <float>, "scoring_levels": {"1.0":
"<description>", "0.5": "<description>", "0.0": "<description>"}},
{"criterion": ...}, ...]}

Example | question: "Explain how photosynthesis works in simple terms"
{"criteria": [{"criterion": "Explains that plants convert sunlight into
chemical energy", "weight": 0.3, "scoring_levels": {"1.0": "Clearly
explains the sunlight-to-energy conversion", "0.5": "Mentions sunlight
or energy but not the conversion process", "0.0": "No mention of the
energy conversion mechanism"}}, {"criterion": "Identifies CO2 and water
as inputs and oxygen and glucose as outputs", "weight": 0.25,
"scoring_levels": {"1.0": "All four substances correctly identified",
"0.5": "Some inputs or outputs mentioned but incomplete", "0.0": "None
identified or incorrect"}}, {"criterion": "Avoids unnecessary jargon
and is understandable to a general audience", "weight": 0.2,
"scoring_levels": {"1.0": "Clear and jargon-free throughout", "0.5":
"Mostly accessible but uses some unexplained technical terms", "0.0":
"Dense with jargon, inaccessible to a general reader"}}, {"criterion":
"Follows a coherent structure from inputs to process to outputs",
"weight": 0.15, "scoring_levels": {"1.0": "Well-organized with clear
progression", "0.5": "Some structure but jumps between ideas", "0.0":
"Disorganized or incoherent"}}, {"criterion": "All stated facts about
photosynthesis are correct", "weight": 0.1, "scoring_levels": {"1.0":
"No factual errors", "0.5": "Minor inaccuracy that does not undermine
the explanation", "0.0": "Contains significant factual errors"}}]}

<User prompt>
{question}
```