CoPE: Co-Evolving Policy and Experience via Context Distillation for Open-Ended Generation

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning with rubric-based rewards is widely used for open-ended generation tasks. However, scalar rewards discard the rich information contained in language feedback. Context distillation aims to internalize knowledge from external context, which we refer to as experience, into model parameters. Fixed, pre-defined context can become suboptimal as the policy improves during training. We propose CoPE, an on-policy context-distillation framework that jointly improves a single policy model's downstream-task performance and rubric-generation capability. The evolving rubrics are then used by a frozen feedback model to extract adaptive experience for subsequent policy training. CoPE includes two phases of context distillation: a) training the policy model with experience as additional context and b) training the policy model as the rubric generator to produce better rubrics for experience extraction. The two phases are performed iteratively and enhance each other during training. We evaluate CoPE on open-ended generation tasks and evaluation ability, where the policy needs to generate rubrics to choose the preferred response. Our results show that CoPE outperforms scalar reward RL and previous experience learning approaches and co-evolves the policy model on downstream tasks and rubric generation.

1 Introduction

Reinforcement learning (RL) has become a de facto part of post-training language models (Ouyang et al., 2022; Bai et al., 2022; Guo et al., 2025). Reward signals are critical to training policy models with RL. For verifiable domains with ground-truth answers, such as math and coding, reinforcement learning with verifiable rewards (Lambert et al., 2024) is effective because rewards are easy to compute. However, open-ended generation tasks such as creative writing (Wang et al., 2024) do not have answers that can be clearly defined and checked automatically.

To train policy models on open-ended tasks, rubric-based reward modeling (Gunjal et al., 2026) is widely used. It consists of two components: rubric generation and judging. Built on rubrics pre-generated or sampled from models, the reward calculation checks whether responses satisfy the rubrics. It provides more fine-grained response evaluation and greater interpretability. Recently, some work applied rubrics to RL for open-ended tasks (Gunjal et al., 2026; Shao et al., 2026).

However, standard RL reduces all evaluation criteria to a single scalar reward. Although many rubrics are prepared for judging responses from the policy, only a scalar reward is used for optimizing the policy, leading to sparse feedback. Experience learning (EL) (Ye et al., 2026a) learns directly from the *experience* extracted by the feedback model with pre-defined rubrics and on-policy sampled responses. Given a prompt, pre-defined rubrics, and corresponding on-policy responses, EL first extracts experience from these inputs with the external feedback model. After that, EL uses on-policy context distillation (Ye et al., 2026b) to internalize experience knowledge into the policy by minimizing the distribution divergence between the policy and the teacher with experience as context. EL converts sparse reward scores to dense language feedback and internalizes it into the policy model.

EL relies on *static* rubrics pre-generated before training. Such pre-defined rubrics, often generated by proprietary models such as GPT-4o, remain fixed as the policy evolves and may therefore fail to capture newly emerging behaviors or failure patterns in on-policy responses. To address this

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.

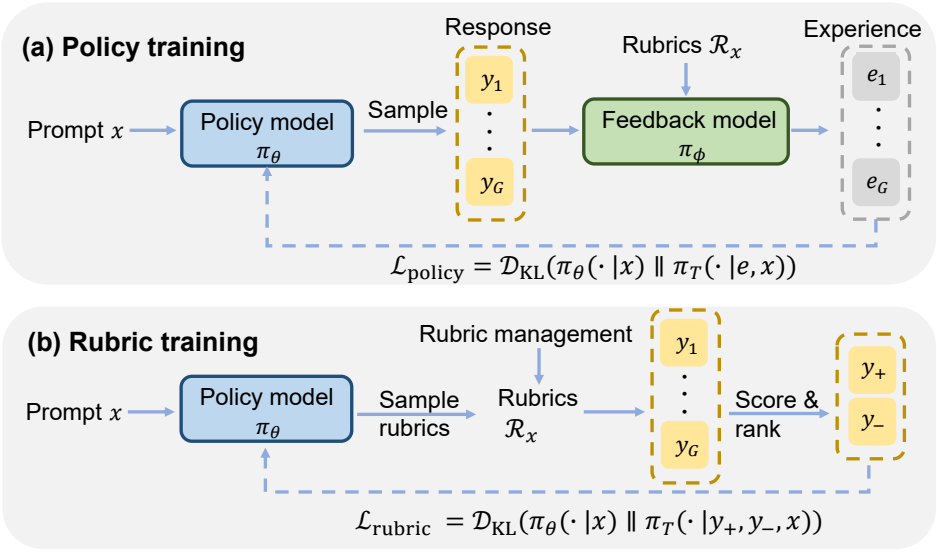

Figure 1: **The overview** of CoPE (α, β). CoPE alternates between two phases with α steps and β steps. **(a)** We first sample responses $\{y_i\}_{i=1}^G$ from the policy π_θ and extract experience e with rubrics $\mathcal{R}_x$ using the frozen feedback model π_ϕ. We use experience as context to train the policy model π_θ using the objective in Eq. (3). **(b)** The same policy model switches from generating downstream-task responses to serving as a rubric generator. The policy model generates new rubrics, which are then used to score all responses. We choose responses y_+ with the highest score and y_- with the lowest score and use them as additional context to train the rubric generator using Eq. (4).

limitation, we propose a new on-policy context distillation approach, CoPE, short for co-evolution with policy and experience. The key idea of CoPE is to make the experience adapt together with the evolving policy. Rather than directly optimizing free-form experience, which is difficult to structure and verify, we evolve the rubrics that guide experience extraction. Specifically, a single policy model learns both to generate responses and to produce increasingly informative rubrics. These evolving rubrics are then used by the feedback model to extract experience from current on-policy responses, allowing the supervision provided to the policy to adapt throughout training. By alternating between learning from experience and improving the rubrics used to extract it, CoPE forms a co-evolutionary loop between the policy and its experience. The overview of CoPE is shown in Figure 1.

We evaluate CoPE on five open-ended generation tasks. CoPE achieves the best performance on them and consistently outperforms EL and RL baselines by a large margin. Specifically, CoPE improves Qwen3-8B by 13.3 points in win rate on AlpacaEval-v2 and by about 3–5 points on other benchmarks. We ablate different design choices for context and training strategy. Furthermore, we show in Sec. 3.3 that both the quality of rubrics generated by the policy and downstream task performance improve during training.

2 Method

We propose CoPE, short for co-evolution with policy and experience, an on-policy context distillation approach for training a language model on open-ended generation tasks, where no ground-truth answers or preference labels are required. Given a prompt x, CoPE trains models with two phases iteratively. In the first phase, we train the policy model with language feedback (i.e. experience) as additional context to improve the policy model for downstream tasks. In the second phase, we optimize the same policy model to generate better rubrics used for extracting experience. The two phases enhance each other and improve the policy model for open-ended tasks and rubric generation during the training.

2.1 Preliminaries

Reinforcement learning (RL) with rubric-based rewards. Given a prompt x, the policy model π_θ generates a response $y \sim \pi_\theta(\cdot \mid x)$. For open-ended generation tasks lacking verifiable answers, such as creative writing (Wang et al., 2024), we define a reward function using K instance-specific

Algorithm 1 CoPE (α, β): Co-Evolution with Policy and Experience

Require: Dataset $\mathcal{S}$, policy π_θ, fixed feedback model π_ϕ, teacher π_T, initial rubrics $\{\mathcal{R}_x\}$, rollout size G, training steps T

for each step $t = 1, \ldots, \lfloor T/(\alpha+\beta) \rfloor$ **do**
  Sample $(x, \mathcal{R}_x) \sim \mathcal{S}$
  // Policy training
  for $a = 1, \ldots, \alpha$ **do**
    Sample responses $\{y_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid x)$
    Extract experiences $e_i \sim \pi_\phi(\cdot \mid x, y_i, \mathcal{R}_x)$
    Update π_θ with $\mathcal{L}_{\text{policy}}$ in Eq. (3)
  end for
  // Rubric training for experience
  for $b = 1, \ldots, \beta$ **do**
    Generate evolving rubrics $\mathcal{R}_x^{\text{new}} \sim \pi_\theta(\cdot \mid x)$
    Update $\mathcal{R}_x$ by deduplication and discrimination-based pruning
    Select highest- and lowest-score response y_+ and y_-
    Update π_θ with $\mathcal{L}_{\text{rubric}}$ in Eq. (4)
  end for
end for
return π_θ

rubric criteria $\mathcal{R}_x = \{(c_k, w_k)\}_{k=1}^{K}$ (Gunjal et al., 2026), where $w_k \in [0, 1]$ and $c_k \in [0, 1]$ denote the weight of criterion k and a correctness function indicating how the response y satisfies criterion k, respectively. The total reward $r(x, y)$ is calculated as

$$r(x, y) = \frac{\sum_{k=1}^{K} w_k c_k}{\sum_{k=1}^{K} w_k}. \tag{1}$$

RL approaches train the policy model π_θ by maximizing the expected reward $r(x, y)$ as

$$\max_\theta \mathbb{E}_{x, y \sim \pi_\theta(\cdot|x)}[r(x, y)]. \tag{2}$$

Compared with RLHF (Ouyang et al., 2022) using a parameterized reward model to assign rewards, rubric-based rewards provide fine-grained and interpretable rules to calculate rewards.

Experience learning (EL) with on-policy context distillation. Instead of only using a scalar reward, EL (Ye et al., 2026a) addresses the problem of training a language model on open-ended tasks by internalizing experience from self-generated rollouts. Formally, given another frozen language model π_ϕ, it extracts language feedback e given the prompt x, response y, and rubrics $\mathcal{R}_x$, i.e., $e \sim \pi_\phi(\cdot \mid x, y, \mathcal{R}_x)$. EL trains the policy model in an on-policy context distillation manner (Ye et al., 2026b) that uses experience as additional context. The training objective is

$$\mathcal{L}_{\text{policy}} = \mathbb{E}_{(x, \mathcal{R}_x) \sim \mathcal{S}, \{y_i\}_{i=1}^{G} \sim \pi_\theta(\cdot|x)} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \mathcal{D}_{\text{KL}}(\pi_\theta(\cdot \mid x, y_{i,<t}) | \pi_T(\cdot \mid e, x, y_{i,<t})) \right], \tag{3}$$

where π_T denotes the teacher model without gradient propagation and $y_{i,<t}$ denotes the prefix up to the t-th token of response y_i. When the teacher π_T is initialized from the same checkpoint as the policy π_θ and is subsequently kept frozen, or maintained as an exponential moving average of π_θ, we refer to the setting as self-distillation. In this paper, we primarily focus on self-distillation where π_T is frozen from the initial checkpoint of π_θ. $\mathcal{D}_{\text{KL}}$ denotes Kullback–Leibler (KL) divergence. Compared with scalar reward RL, EL directly learns from rich language feedback and provides more informative feedback for the policy model.

2.2 CoPE: Co-Evolution with Policy and Experience

Although EL provides more informative feedback, automating and scaling experience generation for training remains unclear. EL uses pre-defined rubrics generated by a static LM. As the policy π_θ

is trained, experience from fixed rubrics becomes outdated and susceptible to reward hacking. To tackle these challenges, we propose *co-evolution with policy and experience* (CoPE) using experience that co-evolves with the policy model for open-ended generation tasks.

The intuition behind CoPE is that improving rubric quality improves the extracted experience because the experience e is extracted from π_ϕ given rubrics $\mathcal{R}_x$. We choose to train the policy as the rubric generator rather than directly generating experience since rubrics are easier to verify and provide clearer structure, whereas experience lacks structure. We ablate different design choices in Sec. 3.3. Behind this intuition, two questions remain: a) how to improve the quality of rubrics and manage them and b) how to improve response y and rubrics $\mathcal{R}_x$ simultaneously with a single policy model. The overview of CoPE is presented in Figure 1 and we detail CoPE as follows.

Given on-policy rollouts $\{y_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid x)$ where G denotes the number of rollouts, we first follow standard EL to train the policy model in an on-policy context distillation manner as in Eq. (3). The difference is that we replace pre-generated rubrics $\mathcal{R}_x$ with dynamic rubrics $\mathcal{R}_x$ during training.

Rubric evolution. In order to improve the quality of experience, we aim to improve the quality of rubrics because the experience is generated from $\pi_\phi(\cdot \mid x, y, \mathcal{R}_x)$. Pre-defined rubrics generated from the static LM limit the quality of experience. To improve rubric quality during training, in each step, we generate rubrics $\mathcal{R}_x^{\text{new}}$ with the current policy π_θ given a prompt x. We prompt the policy model to generate two types of evolving rubrics following (Shao et al., 2026): a) positive rubrics that capture desirable behaviors and relevant knowledge and b) negative rubrics which analyze and summarize undesirable behaviors.

Rubric management. After getting new rubrics $\mathcal{R}_x^{\text{new}}$, to prevent the number of rubrics from growing linearly during training and to improve the coverage and discriminativeness of rubrics, we need to manage the rubrics. We denote the current rubric set as $\mathcal{R}_x^{\text{old}}$. **First**, we deduplicate highly similar rubrics within $\mathcal{R}_x^{\text{old}} \cup \mathcal{R}_x^{\text{new}}$ using n-grams. We set $n = 8$ and the threshold to 0.8 to filter similar rubrics. **Second**, after deduplication, to improve the discriminativeness of rubrics, we remove any rubrics that all responses in the same group either satisfy ($c_k = 1$) or fail to satisfy ($c_k = 0$). **Third**, to control the evaluation cost, we set the maximum number of rubrics to K_{max}. If the number of rubrics is more than K_{max}, we only keep K_{max} rubrics with the highest score variance across all responses. After three steps, we get the rubrics $\mathcal{R}_x$ for training. The rubric management strategy leads to slightly "off-policy" training because we use a mixture of newly generated on-policy rubrics and old rubrics. We empirically find that rubric management leads to better performance and that training is still stable. Therefore, we use this rubric management strategy rather than using only rubrics sampled from the current policy. We then update $\mathcal{R}_x^{\text{old}}$ with $\mathcal{R}_x$ for the next training step.

Build context for rubric training. After acquiring updated rubrics, we train π_θ to generate better rubrics in a context distillation manner. The key to context distillation is to select proper additional context. We consider using rollout responses as additional context because they already exist when we optimize the policy with Eq. (3). Reusing these existing responses incurs no additional rollout cost. We use the response with the highest score y_+ and the response with the lowest score y_- from $\{y_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid x)$ as additional context because they provide contrastive signals for evaluating rubric quality. Together they form a contrastive pair that helps analyze whether newly generated rubrics can distinguish them. For newly sampled rubrics $\mathcal{R}_x^{\text{new}}$ and rubrics $\mathcal{R}_x$ with the rubric management described above, the training objective of context distillation is formulated as

$$\mathcal{L}_{\text{rubric}} = \mathbb{E}_{x \sim \mathcal{S}, \mathcal{R}_x^{\text{new}} \sim \pi_\theta(\cdot|x)} \frac{1}{|\mathcal{R}_x|} \left[\sum_{j=1}^{|\mathcal{R}_x|} \mathcal{D}_{\text{KL}}(\pi_\theta(\cdot \mid x, \mathcal{R}_{x,<j}) \| \pi_T(\cdot \mid x, y_+, y_-, \mathcal{R}_{x,<j})) \right]. \quad (4)$$

Eq. (3) and Eq. (4) define different objectives to optimize responses and rubrics. To optimize both objectives in a single policy model, we alternate between Eq. (3) and Eq. (4). This enables the policy to generate better responses for open-ended tasks and better rubrics during training.

Joint optimization of policy and experience. One key question is how to balance downstream task ability and rubric generation under a limited computation budget. To study this, we first extend CoPE to a more general and flexible formulation, CoPE (α, β), where α and β denote the number of updates for Eq. (3) and Eq. (4), respectively. During training, we first update the policy π_θ α steps

Table 1: Evaluation results of RL, EL and CoPE on five benchmarks. "Policy" denotes the policy model π_θ and "feedback" denotes the feedback model π_ϕ. CW-v3 denotes CreativeWritingV3.

Policy + Feedback	**Method**	**WildChat** Score	**AlpacaEval-v2** Win Rate (%)	**WildBench** Score	**ArenaHard-v2** Win Rate (%)	**CW-v3** Score
Qwen3-8B + Qwen3-8B	Base Model	78.1	34.5	17.6	29.1	73.8
	RL	79.2	37.3	18.4	30.5	74.3
	EL	80.0	40.0	21.8	31.0	76.0
	CoPE (ours)	**83.4**	**47.8**	**23.2**	**32.9**	**79.1**
Qwen3-8B + GPT-4o	RL	80.4	38.2	19.2	31.2	74.4
	EL	80.7	37.4	22.0	31.9	75.3
	CoPE (ours)	**84.5**	**43.4**	**24.6**	**32.5**	**76.0**
OLMo-3-7B + OLMo-3-7B	Base Model	75.7	43.6	39.0	21.8	78.7
	RL	76.0	45.9	42.1	23.3	78.8
	EL	77.1	50.8	47.5	26.2	78.8
	CoPE (ours)	**79.2**	**54.3**	**51.2**	**29.7**	**80.4**
OLMo-3-7B + GPT-4o	RL	76.8	44.7	40.3	22.5	77.8
	EL	78.2	48.5	46.2	25.1	78.0
	CoPE (ours)	**79.0**	**50.6**	**52.8**	**27.0**	**79.4**

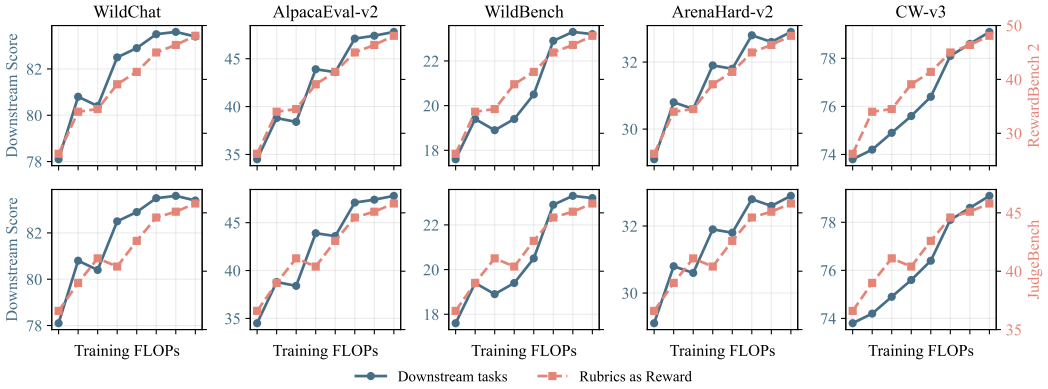

Figure 2: Performance of downstream tasks and rubrics as reward.

with Eq. (3) to train the policy model for downstream tasks and update the same policy π_θ with Eq. (4) β steps to optimize rubrics. Therefore, in the next iteration, the policy will generate better rubrics, which in turn yield better experience.

As the policy evolves, the rubric generator will provide more discriminative rubrics and thus better experience. The policy will benefit from evolving experience and improve further. Different values of (α, β) control different preferences for training the policy for downstream tasks or as the rubric generator. A higher value of α/β denotes a stronger preference to optimize the policy for downstream tasks and inversely for rubric generation. We study the impact of (α, β) in Sec. 3.3. Note that EL is a special case of CoPE (α, β) when we set $\alpha = 1$ and $\beta = 0$. The full algorithm of CoPE (α, β) is shown in Algorithm 1.

3 Experiments

3.1 Setup

Training Data and Models. We train Qwen3-8B (non-thinking mode) (Yang et al., 2025) and OLMo-3-7B-Instruct (Olmo et al., 2025) as policy models. For each policy model, we use the same frozen policy model or GPT-4o as the feedback model π_ϕ. We use 7485 prompts from WildChat-IF (Zhao et al., 2024) as the training set. Following (Ye et al., 2026a), we mix prompts from Tulu3 (Lambert et al., 2024) with a 4:1 ratio to maintain general ability. The same training data mixture and base model ensure a fair comparison among different algorithms.

Training. For the RL baseline, we use rubric-as-reward (Gunjal et al., 2026) with GRPO (Guo et al., 2025) as the implementation. For all prompts used in training, we use rubrics generated by GPT-4o following (Ye et al., 2026a). For our CoPE, we set $\alpha = 2$ and $\beta = 1$ to balance

Table 2: Evaluation on reward benchmarks to assess the rubric generator's ability to rank preferred responses over dispreferred ones. Note that public baselines are only for reference because the training set is different. †: only for verifiable tasks. ‡: requires ground-truth preference labels.

Policy	Feedback	RewardBench 2	JudgeBench
Public baselines for reference			
RLCER†	-	45.0	49.2
Rubric-ARM‡	-	56.1	51.5
EvoLM	-	46.0	44.4
Llama-3.1-8B	-	36.4	46.0
Models trained with CoPE			
Qwen3-8B	-	26.2	36.6
Qwen3-8B	Qwen3-8B	48.1	45.8
Qwen3-8B	GPT-4o	48.4	49.0
OLMo-3-7B	-	43.5	41.1
OLMo-3-7B	OLMo-3-7B	49.7	46.8
OLMo-3-7B	GPT-4o	48.4	47.2

policy training and rubric generation. We keep the compute cost of EL and CoPE similar for a fair comparison, although exactly matching computation is impossible. We set the batch size to 256 and group size $G = 8$. The number of training epochs is set to 3. We set the temperature to 1.0 and max generation length to 4096. The learning rate is set to 1e-6. Reverse KL in Eq. (3) and Eq. (4) is computed over the top 256 tokens of the student model's probabilities. We set the maximum number of rubrics per prompt K_{max} to 15. For a fair comparison, we use the same initial rubrics (generated by GPT-4o), the same group size G, the same feedback model as EL (Ye et al., 2026a).

Evaluation. Our evaluation includes two parts. *First*, we evaluate downstream task performance. We use 250 prompts (not in the training set) in WildChat-IF (Zhao et al., 2024) with 4 responses sampled per prompt as in-domain evaluation. Furthermore, we evaluate on four out-of-domain benchmarks with open-ended questions: WildBench (Lin et al., 2025), AlpacaEval v2.0 (Li et al., 2023), ArenaHard v2.0 (Li et al., 2025), and CreativeWritingV3 (Paech, 2025). GPT-4o serves as the evaluator for all benchmarks. For CreativeWritingV3 and WildChat-IF, we use direct rubric-based scoring ranging from 0 to 100. For AlpacaEval v2.0 and ArenaHard v2.0, we report pairwise win rates against GPT-4-Turbo references. We report preference scores in the range [-100,100] for WildBench. We set temperature to 0.7, top-p to 0.95, and maximum generation length to 4096 and report the average of three runs with different random seeds.

Second, we evaluate CoPE on reward benchmarks. We use RewardBench 2 (Malik et al., 2026) and JudgeBench (Tan et al., 2025) to evaluate whether rubric generators trained with CoPE can correctly judge preferred or dispreferred responses. In addition to the base model as a baseline, we include several public baselines for reference, including EvoLM (Li et al., 2026), RLCER (Sheng et al., 2026), and Rubric-ARM (Xu et al., 2026). We additionally provide a qualitative comparison with related rubric-based methods in Table 7, highlighting their differences in training requirements, applicability, and learning signals.

3.2 Main Results

Table 1 shows the results of different approaches on five open-ended generation tasks. All approaches consistently improve the base model. EL and CoPE provide more informative feedback compared with RL and consistently achieve better results. Compared with EL using experience from *static* rubrics, CoPE with evolving rubrics and experience achieves the best performance across all benchmarks. Specifically, CoPE improves Qwen3-8B by 13.3 percentage points in win rate on AlpacaEval-v2 with Qwen3-8B as the feedback model and improves OLMo3-7B by 12.5 points on WildBench. Results in Table 1 show the effectiveness of CoPE across different tasks and various policy and feedback model combinations.

Table 2 reports results on RewardBench 2 and JudgeBench. Table 2 shows that CoPE improves the base model on both benchmarks across different policy and feedback models. Notably, CoPE

Table 3: Ablation of context design. We use Qwen3-8B as the policy model and a fixed Qwen3-8B as the feedback model. We report the win rate on AlpacaEval-v2. **(a)** Context ablation of $\mathcal{L}_{\text{policy}}$ (c.f. Eq. (3)). Pos. and Neg. denote positive rubrics and negative rubrics, respectively. **(b)** Context ablation of $\mathcal{L}_{\text{rubric}}$ (c.f. Eq. (4)). **(c)** Ablation of evolving experience or rubrics during training. We highlight the default setting.

(a) $\mathcal{L}_{\text{policy}}$ context

Context	Win Rate (%)
-	34.5
Pos. rubrics	40.6
Neg. rubrics	39.5
Rubrics $\mathcal{R}_x$	45.4
Experience e	47.8

(b) $\mathcal{L}_{\text{rubric}}$ context

Context	Win Rate (%)
-	34.5
$\{y_i\}_{i=1}^{G}$	47.1
y_+, y_-	47.8
Random pair	47.4
y_+	38.6
y_-	36.1

(c) Evolving context

Context	Win Rate (%)
-	34.5
Experience e	41.5
Rubrics $\mathcal{R}_x$	47.8

Table 4: Ablation study on initial rubric generator, rubric management strategy, and the maximum number of rubrics $K_{\max}$. Default settings are highlighted.

Setting	**AlpacaEval-v2**	**WildBench**
Initial rubric generator		
Qwen3-8B	47.4	22.5
GPT-4o	47.8	23.2
Rubric management		
Fixed	40.0	21.8
Only $\mathcal{R}_x^{\text{new}}$	43.8	22.1
w/o pruning	44.3	22.5
w/o deduplication	46.1	23.1
Full	47.8	23.2
$K_{\max}$		
$K_{\max} = 5$	45.1	22.7
$K_{\max} = 10$	46.9	23.1
$K_{\max} = 15$	47.8	23.2
$K_{\max} = 20$	47.6	22.9

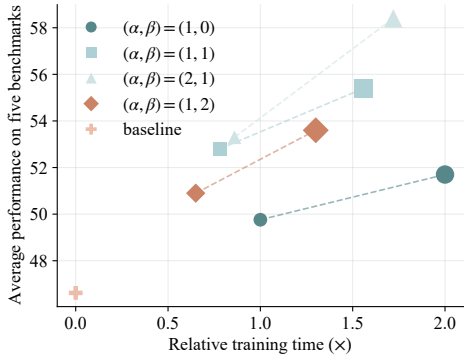

Figure 3: Ablation study on different (α, β). $(\alpha, \beta) = (0, 0)$ denotes the Qwen3-8B baseline and $(\alpha, \beta) = (1, 0)$ denotes EL. Settings with $\beta \neq 0$ correspond to CoPE variants. We present results under two computation budgets to test the scalability of different (α, β).

improves Qwen3-8B by 22.2 points on RewardBench 2 and 12.4 points on JudgeBench with GPT-4o as the feedback model. The results validate the effectiveness of policy models trained with CoPE as rubric generators. Results in Table 1 and 2 show that CoPE can improve a single policy model on both downstream tasks and rubric generation for evaluation.

3.3 Analysis

In this section, we analyze design choices and hyperparameters in CoPE. We take Qwen3-8B as the policy model π_θ and use a frozen Qwen3-8B as the feedback model π_ϕ.

Ablation study on α and β. Figure 3 shows the results of CoPE with different (α, β) under different computation budgets. We have two observations. *First*, for all settings with $\beta \neq 0$, CoPE outperforms EL under a similar computation budget, which shows the robustness of CoPE to the value of (α, β). Furthermore, it validates the importance of co-evolving the policy and experience. *Second*, CoPE shows better scalability than EL. When computation is increased (2× in our experiments), the gains of CoPE are much larger than those of EL, while the improvement of EL is limited. Specifically, EL improves by 1.9 points over five benchmarks (49.8 to 51.7), whereas CoPE improves more substantially (5.1 points, 53.3 to 58.4) with our default setting $(\alpha, \beta) = (2, 1)$. Compared with EL, CoPE shows higher potential for scaling to larger training budgets.

Context ablation. A core question for context distillation is *what* context we distill. We ablate different context designs in CoPE, and the results are reported in Table 3. Table 3a shows that using experience extracted from the feedback model in Eq. (3) achieves better performance compared with using rubrics (positive, negative, or all). Table 3b shows that using all responses $\{y_i\}_{i=1}^{G}$ achieves

Table 5: Evaluation of rubric quality on deep research tasks.

Rubric generator	HealthBench	ResearchQA
Reported baselines		
GPT-4.1	52.5	51.0
Qwen3-8B	53.0	57.2
EvoLM	58.4	59.3
Qwen3-8B†	54.6	58.9
CoPE †	60.1	61.5

†: We roll out responses ourselves and evaluate on these responses. Reported baselines are from EvoLM (Li et al., 2026).

Table 6: Evaluation of the policy model on medical benchmarks.

Policy model	HealthBench	RaR-Medicine
Qwen3-4B	12.6	15.1
EvoRubrics	20.8	16.4
CoPE	23.4	18.6
Qwen3-8B	14.4	17.0
EvoRubrics	23.0	17.3
CoPE	28.9	20.5

lower performance than only using the best and the worst responses. The intuition behind this is that the most discriminative response pairs can provide better examples to help the model generate rubrics than all responses. Using random pairs also outperforms using all responses or a single response, which shows that simple but contrastive signals provide better context for rubric training. Instead of evolving rubrics during training, we also explore directly generating experience in Eq. (4). We report the results in Table 3c. Directly generating experience performs far worse than generating rubrics, although it still improves over the baseline. The intuition behind this is that it may be too difficult to directly optimize experience, and rubrics can guide the model to generate clearer and more structured knowledge with rules.

Rubrics evolve with the policy. The main idea of CoPE is the co-evolution of policy and experience by evolving rubrics. We evaluate the ability of the policy as a rubric generator on RewardBench 2 (Malik et al., 2026) and JudgeBench (Tan et al., 2025). Figure 2 shows results on downstream tasks and two reward benchmarks during training. We take Qwen3-8B as the policy model π_θ and a frozen Qwen3-8B as the feedback model π_ϕ. Results show that the quality of rubrics generated by the trained policy and performance on downstream tasks consistently increase during training.

Ablation on rubrics design. We ablate different rubric designs, including initial rubric generation and rubric management. Results are listed in Table 4. We first test whether CoPE relies on frontier models to generate initial rubrics by using Qwen3-8B to generate rubrics. Results show that initial rubrics generated by Qwen3-8B are still effective and outperform EL significantly, which suggests that the method is not sensitive to the initial rubric generator. Furthermore, we ablate the importance of different rubric management strategies. Compared with the fixed rubrics (EL), using only new rubrics each step still outperforms EL by a large margin, which shows the importance of evolving rubrics. Without more detailed rubric processing including deduplication and pruning rubrics that are too easy or too hard, the performance is degraded compared with the full strategy. We also report performance with respect to the maximum number of rubrics $K_{\max}$. It is shown that for all values of $K_{\max}$, CoPE outperforms EL by a large margin. A higher $K_{\max}$ allows more rubrics and we find that increasing $K_{\max}$ from 5 to 15 leads to better performance; however, increasing $K_{\max}$ beyond 15 yields marginal performance gains. As a result, we set $K_{\max} = 15$ for the best performance.

3.4 Generalization to Other Domains

Evaluation on deep research rubrics. The training set of CoPE is a mixture of Tulu3 (Lambert et al., 2024) and WildChat-IF (Zhao et al., 2024), which mainly covers general-purpose tasks. We follow EvoLM (Li et al., 2026) to test whether the trained rubric generator can produce evaluation criteria usable for deep research tasks. We use HealthBench (Arora et al., 2025) and ResearchQA (Yifei et al., 2026) as the out-of-distribution test sets. For each prompt, we use DR-Tulu-8B-SFT and the corresponding RL checkpoints (Shao et al., 2026) to generate responses and use GPT-4.1 to grade all responses with human expert rubrics from the official benchmark.

We report pairwise ranking accuracy in Table 5. Results show that CoPE improves the base model by 5.5 points on HealthBench and 2.6 points on ResearchQA. This indicates that CoPE trained with WildChat-IF and Tulu3 can be a strong rubric generator for deep research evaluation.

Evaluation on the medical domain. We train CoPE with 4,000 standard-difficulty examples from the training set of HealthBench. We evaluate CoPE on two open-ended benchmarks in the medical domain: 1000 hard cases from HealthBench (Arora et al., 2025) and RaR-Medicine (Gunjal

et al., 2026). Prompts in RaR-Medicine are not seen during training and can be viewed as out-of-distribution tasks. We add EvoRubrics (Ding et al., 2026), a reinforcement learning approach with evolving rubrics, as a baseline.

Table 6 reports results on HealthBench and RaR-Medicine. CoPE improves both base models by a large margin. Specifically, Qwen3-8B achieves a 14.5-point gain on HealthBench. Furthermore, CoPE clearly outperforms EvoRubrics, a reinforcement learning approach with evolving rubrics. Notably, CoPE achieves a 5.9-point improvement over EvoRubrics on HealthBench with Qwen3-8B. Results show that CoPE is effective in medical open-ended tasks and demonstrate the superiority of language feedback over scalar-reward RL.

4 Related Work

Context Distillation. The goal of context distillation is to internalize knowledge into model parameters (Snell et al., 2022). Recently, to eliminate the exposure bias of off-policy training, on-policy context distillation (Ye et al., 2026b) distills context from a teacher with additional context using the sequence generated by itself. A similar line of work is on-policy self-distillation (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026), which typically uses itself as the teacher model. They are mainly designed for verifiable tasks and use ground-truth answers or environment feedback as context. The closest work to CoPE is experience learning (Ye et al., 2026a), which combines pre-defined rubrics and responses to generate experience with another feedback model. After that, it applies on-policy context distillation to train the policy model to learn from the experience for non-verifiable tasks. Unlike experience learning with pre-generated rubrics, we propose CoPE, which co-evolves the policy model and experience.

Learning from experience and language feedback. Compared with sparse rewards in RL, natural language feedback provides fine-grained textual critiques (Saunders et al., 2022; Chen et al., 2024). Most of them utilize supervised fine-tuning or reinforcement learning to learn from *static* critiques or critique-guided refinements (Wang et al., 2025; Chen et al., 2024; Feng et al., 2024; Hong et al., 2025; Luo et al., 2025; Huang et al., 2026). Critique-GRPO (Zhang et al., 2026) and EL (Ye et al., 2026a) sample language feedback from the policy being trained or an external feedback model, while they are only for task training and lack specific training on how to give better language feedback. CoPE enables a single policy model to learn open-ended generation tasks and generate better language feedback iteratively during training.

Rubrics as reward. Rubrics as rewards (Gunjal et al., 2026) are a promising alternative to human preferences for non-verifiable tasks such as open-ended question answering and creative writing. Each prompt is associated with a set of evaluation criteria, and the reward is computed from them. By factorizing the reward into multiple rules, rubrics as rewards provide fine-grained and transparent evaluation. Rubrics are usually written by human experts or generated by frontier models. They are widely used in reinforcement learning (Gunjal et al., 2026) and on-policy distillation (Rezaei et al., 2026). Recently, several works have explored dynamic rubrics for reinforcement learning (Shao et al., 2026; Li et al., 2026; Ding et al., 2026), while they still provide sparse rewards rather than language feedback. RCSD (Gu et al., 2026) trains a rubric generator and a reasoning model as a single policy. However, it differs from CoPE in two ways: i) it trains the rubric generator and reasoning model in two phases *separately*, while CoPE trains them *iteratively*, and ii) it needs reference answers to train the rubric generator, while CoPE does not.

5 Conclusion

In this paper, we introduce CoPE, a post-training framework that co-evolves a single policy model to generate discriminative rubrics and solve downstream tasks. CoPE consists of two phases that are performed iteratively. In the first phase, CoPE conducts on-policy context distillation with experience extracted using rubrics. In the second phase, we train the policy as a rubric generator to provide discriminative rubrics using contrastive rollouts as additional context. These two phases are conducted iteratively and enhance each other. We evaluate CoPE on five open-ended generation tasks, and CoPE outperforms baselines and prior work by a large margin. Furthermore, we show that the two phases enhance each other during training and that CoPE also generalizes to other tasks.

AI USE STATEMENT

In this work, generative AI tools were used only for language refinement and coding support. Specifically, GPT6 assisted in the writing process by enhancing manuscript clarity and correcting grammatical issues, while Codex served as a coding agent during experiments by helping with code generation. These tools were not involved in generating research ideas, forming scientific claims or conclusions, or interpreting experimental results.

All content produced with AI assistance was thoroughly reviewed and verified by the authors. In particular, every piece of AI-generated code was manually inspected by the author team, and its correctness was confirmed by running it and examining the outputs. The full manuscript, including AI-assisted language edits, was manually checked by the authors to ensure technical accuracy and fidelity to the intended meaning. All experimental results and conclusions were independently verified by the author team.

We accept full responsibility for the final content of this work, including all text, claims, code, experimental results, and other artifacts produced with the help of generative AI.

REFERENCES

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

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Angelica Chen, Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. Learning from natural language feedback. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=xo3hI5MwvU.

Hongxin Ding, Baixiang Huang, Yue Fang, Weibin Liao, Zheng Li, Jinyang Zhang, Zhijing Wu, Junfeng Zhao, and Yasha Wang. Evorubrics: Dynamic rubrics as rewards via adversarial coevolution for llm reinforcement learning. *arXiv preprint arXiv:2606.23038*, 2026.

Xidong Feng, Bo Liu, Yan Song, Haotian Fu, Ziyu Wan, Girish A Koushik, Zhiyuan Hu, Mengyue Yang, Ying Wen, and Jun Wang. Natural language reinforcement learning. *arXiv preprint arXiv:2411.14251*, 2024.

Siyi Gu, Jialin Chen, Sophia Zhou, Arman Cohan, and Rex Ying. Rethinking reward supervision: Rubric-conditioned self-distillation. *arXiv preprint arXiv:2606.19327*, 2026.

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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Joey Hong, Kang Liu, Zhan Ling, Jiecao Chen, and Sergey Levine. Natural language actor-critic: scalable off-policy learning in language space. *arXiv preprint arXiv:2512.04601*, 2025.

Lei Huang, Xiang Cheng, Chenxiao Zhao, Guobin Shen, Junjie Yang, Xiaocheng Feng, Yuxuan Gu, Xing Yu, and Bing Qin. Bootstrapping exploration with group-level natural language feedback in reinforcement learning. *arXiv preprint arXiv:2603.04597*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Deen Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=QkfkxyRizZ.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Shuyue Stella Li, Rui Xin, Teng Xiao, Yike Wang, Rulin Shao, Zoey Hao, Melanie Sclar, Sewoong Oh, Faeze Brahman, Pang Wei Koh, et al. Evolm: Self-evolving language models through co-evolved discriminative rubrics. *arXiv preprint arXiv:2605.03871*, 2026.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KfTf9vFvSn.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking LLMs with challenging tasks from real users in the wild. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MKEHCx25xp.

Renjie Luo, Zichen Liu, Xiangyan Liu, Chao Du, Min Lin, Wenhu Chen, Wei Lu, and Tianyu Pang. Language models can learn from verbal feedback without scalar rewards. *arXiv preprint arXiv:2509.22638*, 2025.

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.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Samuel J Paech. Eq-bench creative writing benchmark v3. https://github.com/EQ-bench/creative-writing-bench, 2025.

MohammadHossein Rezaei, Anas Mahmoud, Zihao Wang, Utkarsh Tyagi, Advait Gosai, Razvan-Gabriel Dumitru, Aakash Sabharwal, Bing Liu, and Yunzhong He. Rubric-guided self-distillation: Post-training without rubric verifiers. *arXiv preprint arXiv:2606.12507*, 2026.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

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*, 2026. URL https://openreview.net/forum?id=97NEP1pyS3.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qA6FgH0nnZ.

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.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

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.

Tiannan Wang, Jiamin Chen, Qingrui Jia, Shuai Wang, Ruoyu Fang, Huilin Wang, Zhaowei Gao, Chunzhao Xie, Chuou Xu, Jihong Dai, et al. Weaver: Foundation models for creative writing. *arXiv preprint arXiv:2401.17268*, 2024.

Yubo Wang, Xiang Yue, and Wenhu Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025.

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, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tianzhu Ye, Li Dong, Guanheng Chen, He Zhu, Xun Wu, Shaohan Huang, and Furu Wei. Llm-as-a-coach: Experiential learning for non-verifiable tasks. *arXiv preprint arXiv:2607.18110*, 2026a.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026b.

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.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen M. Meng. Advancing LLM reasoning with natural language and numerical feedback. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gz7hVnrRWq.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

A Prompt Templates

We use prompt templates from EL (Ye et al., 2026a) and EvoLM (Li et al., 2026) and modify them slightly to fit our approach. We provide them here to make the paper self-contained and to facilitate reproducibility.

Prompt template for EL knowledge extraction. We parse the experience e from the <experience> tags and the score from the <score> tags.

```
You are an expert evaluator. Given a user prompt, a generated response,
and a list of quality rubrics, please rate the overall quality of the
response on a scale of 1 to 10 based on how well it satisfies the
rubrics.
Consider all rubrics holistically when determining your score. A
response that violates multiple rubrics should receive a lower score,
while a response that satisfies all rubrics should receive a higher
score.

<prompt>
{instruction}
</prompt>

<response>
{response}
</response>

<rubrics>
{rubric_list_string}
</rubrics>

First, analyze the response against each rubric item, discussing how
well the response meets or fails each criterion. Then, provide your
final score as an integer between 1 and 10, wrapped in <score> and
</score> tags.

After the score, distill your analysis into transferable experiential
knowledge which is general, high-level, widely applicable insights that
would help improve future responses to similar tasks. Focus on reusable
strategies and patterns rather than details specific to this particular
response. Output this knowledge wrapped in <experience> and
</experience> tags.

Example ending:
<score> your_integer_score_from_1_to_10 </score>

<experience>
some experiential knowledge...
</experience>

Your evaluation:
```

Prompt template for the teacher model on experience.

```
Here is some experiential knowledge:

<experience>
{experience}
</experience>

Given the above experiential knowledge, solve the following problem:
{prompt}
```

Policy prompt.

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

<User prompt>
```

```
{question}
```

Prompt template for generating rubrics.

```
<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.
5. The criteria is an excellence indicators distinguishing superior
responses or critical flaws definitively degrading quality.
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}
```

Judging prompt.

```
<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}
```

Table 7: **Comparison of rubric-based methods**. "No external label" indicates that reference answers or preference pairs are not required.

Method	Trains rubric generator	No external labels	Support verifiable domains	Co-evolves with policy	Language feedback
Rubrics as reward	✘	✔	✔	✘	✘
RLCER	✔	✘	✘	✔	✘
Rubric-ARM	✔	✘	✔	✘	✘
EvoLM	✔	✔	✔	✔	✘
EL	✘	✔	✔	✘	✔
CoPE (ours)	✔	✔	✔	✔	✔

```
Your evaluation:
```

B Comparison with Related Work

We compare our method CoPE with previous rubric-based methods in Table 7, including rubrics as reward (Gunjal et al., 2026), EvoLM (Li et al., 2026), RLCER (Sheng et al., 2026), and Rubric-ARM (Xu et al., 2026) and EL (Ye et al., 2026a).

C Examples

We show one example of evolving rubrics (Table 8) and experience (Table 9) during the training.

Table 8: One example of rubric evolution during the training. Prompt: Write an engaging and a constructive article for my Morocco travel guide book on "Travel protection insurance" with humanlike style, non repetitive phrases and avoidance of unnatural sentences.

Epoch	Rubrics
1	The response should use a humanlike style that is engaging and avoids repetitive phrases; The response should avoid unnatural sentences and provide a fluid reading experience; The response must include key details about travel protection insurance relevant to Morocco travel; The response should include caveats about the importance of travel insurance in managing travel risks; The response should maintain a constructive tone that encourages informed travel decisions; The response must avoid giving personalized or definitive directives regarding insurance purchases; The response should be clear and complete, covering all necessary aspects of travel protection insurance; The response should maintain consistent formatting to enhance readability.
2	The article is engaging and constructive for a Morocco travel guidebook, capturing the reader's interest while providing useful guidance; The article uses a humanlike writing style, with natural, varied phrasing and avoidance of repetitive or robotic language; The article provides accurate and relevant information about travel protection insurance for Morocco, including practical tips or considerations; The article is well-structured and coherent, with a clear introduction, body, and conclusion that flow logically.
3	The article provides relevant and useful information about travel protection insurance in the context of travel to Morocco; The article is engaging and constructive, presenting the topic in a way that holds a travel-guide reader's interest while offering useful guidance; The article uses humanlike, natural language and maintains a smooth, fluid reading experience; The article avoids repetitive wording, sentence patterns, and redundant presentation of the same ideas; The article is coherent and appropriately structured for inclusion in a travel guidebook.

Table 9: One example of experience evolution during the training based on rubrics in Table 8.

Epoch	Experience
1	For informative travel writing, combine practical coverage with an engaging, human-like presentation. Include the major aspects readers need to understand the subject, explain how those considerations relate to travel risks, and keep the guidance constructive rather than merely listing possible problems. Maintain natural flow by varying phrasing and avoiding repeated rhetorical patterns, ideas, and expressions. In particular, avoid repeatedly emphasizing the same reassurance or benefit in slightly different words; once a point has been established, move the article forward with new information. When writing about consequential choices such as insurance or other forms of protection, explain benefits, risks, and relevant considerations without turning general information into definitive purchasing instructions. Prefer language that helps readers evaluate their needs and policy options over statements implying that one choice is universally necessary. Clarity and completeness should work together: cover the important dimensions of the topic, but organize them consistently so that additional detail contributes new value rather than restating earlier advice. Use formatting and paragraph structure to make practical information easy to follow.
2	For a destination-focused guidebook article, make practical information feel genuinely tailored to the setting rather than attaching destination references to otherwise generic advice. Concrete contextual examples can strengthen relevance, but the underlying guidance should also explain why particular considerations matter in that travel environment. Engagement should come from useful specificity and an appealing narrative voice, not from repeatedly using promotional or dramatic expressions. Avoid relying too heavily on familiar travel-writing formulas, superlatives, and recurring reassurance; they can make otherwise fluent prose feel templated. Humanlike writing benefits from variation in sentence construction, paragraph openings, transitions, and rhetorical devices. Watch for recurring structures that perform the same function, especially repeated statements about security, confidence, unexpected events, or enjoying the journey. Build the article as a progression rather than a sequence of loosely connected coverage categories. Establish the subject and its relevance, develop practical considerations in a logical order, and conclude by synthesizing the discussion rather than simply repeating the central recommendation. Each stage should advance the reader's understanding.
3	In guide-style informational writing, prioritize substantive relevance to the intended travel context. General information becomes more useful when connected to realistic destination-specific situations and considerations, while generic material should be limited unless it directly helps the reader understand the subject. Keep the article engaging and constructive by pairing practical guidance with readable prose. Avoid overusing promotional language or familiar travel clichés; sustained reader interest is better achieved through useful detail, purposeful examples, and an approachable voice. Naturalness and repetition should be managed separately. For natural prose, vary syntax, sentence length, transitions, and paragraph openings while favoring idiomatic expressions. For non-repetition, check whether the same message is being communicated several times under different wording and remove passages that add no meaningful information. Give every paragraph a distinct role in the article. Arrange those roles in a logical sequence and use transitions that reflect genuine relationships between ideas. A conclusion should provide closure or consolidate the reader's understanding rather than extensively restating points already established in the introduction and body.