# ARRCE: LEARNING PROCESS RUBRICS FROM STATIC ANCHORS FOR OPEN-DOMAIN REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We introduce Asymmetrically Anchored Rubric–Response Co-Evolution (ARRCE), a reinforcement learning framework that learns process rubrics from automatically synthesized static supervision for open-domain tasks. The central difficulty is to adapt evaluation criteria to the policy's changing behavior while preserving the task requirements that define a useful answer. ARRCE decouples rubric synthesis into an offline stage that constructs and freezes task and evidence-backed fact criteria, and an online stage that proposes process criteria from sampled trajectories whose relative strengths are determined by static-rubric scores. The static criteria also supervise candidate quality: a proposed process rubric is rewarded for separating held-out trajectories in a direction consistent with static performance, subject to observability and scope checks, before entering a query-specific memory for subsequent response rewards. A shared policy learns response generation and rubric proposal through separate objectives in one parameter update, yielding a synthesis loop that requires no additional human annotation of trajectory preferences or process rubrics. We instantiate this framework on dataset-derived tasks with synthesized static rubrics across healthcare, scientific question answering, and long-form writing. The experimental results show that full ARRCE improves over the response-SFT baseline by $0.057$ on ResearchQA ($+7.8\%$), $2.99$ OP points on LLMEval-Med ($+4.4\%$), $0.248$ on WritingBench ($+3.3\%$), and $0.040$ on FollowBench ($+8.1\%$), and the gains are positive across the four benchmarks.[1]

## 1 INTRODUCTION

Open-domain reinforcement learning requires a reward that distinguishes useful answers when no single executable test captures task success. A healthcare response must respect the conversation and avoid unsafe advice; a scientific answer must cover the question and support its claims; a long-form response must satisfy constraints while remaining coherent. Rubrics make these requirements explicit through criteria that an evaluator can apply to generated answers. Recent work uses instance-specific and evidence-grounded rubrics to extend reinforcement learning beyond tasks with exact-match or programmatic verifiers (Gunjal et al., 2026; Huang et al., 2025; Mei et al., 2026).

The difficulty is deciding which criteria should remain fixed and which should adapt. Task requirements and externally supported facts define the target of optimization, but sampled trajectories also expose behavioral differences that are difficult to enumerate in advance. These differences can concern evidence selection, citation use, organization, or other visible actions. A fixed set of criteria may provide little guidance about such differences as the policy changes. Allowing every criterion to change with the policy creates a different problem: an easier evaluation standard can appear to be an improvement in the answer. A useful adaptive reward therefore needs both a source of adaptation and a reference that constrains what adaptation means.

Evolving-rubric methods already address parts of this problem. DR Tulu updates rubrics during deep-research training; EvoRubric and EvoLM train response and rubric generation within one

[1] Anonymous code repository: `https://anonymous.4open.science/r/ARRCE-5C5A`.



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.

model; EvoRubrics rewards generated rubrics using discrimination, alignment, and improvement under golden criteria (Shao et al., 2026; Guan et al., 2026; Li et al., 2026; Ding et al., 2026). These precedents motivate a more specific question: *can a fixed, automatically synthesized rubric supply both the trajectory contrasts and the learning signal needed to synthesize useful process rubrics without additional instance-level human annotation?*

We propose ARRCE, which decouples static and process rubric synthesis. Before reinforcement learning, a synthesis model extracts an answer contract from each task, gathers evidence where external verification is needed, and produces task and fact criteria. The resulting static rubric is frozen. During training, the current policy samples response trajectories, and static-rubric scores identify the stronger and weaker member of each proposal pair, with ties represented explicitly. Conditioned on these visible trajectories and their scores, the policy proposes an observable process criterion. An evaluator then applies that candidate to a disjoint set of trajectories from the same task. The candidate receives a reward for useful discrimination in the direction defined by the static rubric, together with structural and semantic checks. This creates two concrete uses for the same static supervision: it labels the examples from which a process rubric is synthesized and evaluates the quality of the synthesized rule.

Accepted process rubrics enter a delayed, query-specific memory and influence subsequent response rewards within static feasibility constraints. Response generation and rubric proposal share parameters, but retain separate sampled actions, rewards, and optimization objectives.

Our contributions are three connected elements of this supervision pipeline:

- We decouple offline static-rubric synthesis from online process-rubric synthesis, preserving a fixed reference while the adaptive criteria change.
- We train the process-rubric proposer with static-rubric supervision on held-out trajectories: global discrimination together with the constant-rule mask excludes degenerate candidates, while a static-tie residual rewards refinement within groups that the aggregate static score leaves unresolved.
- We construct trajectory comparisons automatically from static scores, enabling process-rubric synthesis without manually labeled strong/weak examples or process criteria.

We ground the method and its evaluation in a complete training procedure. The training data contain 3,696 prepared task records with synthesized static rubrics across healthcare, scientific question answering, and long-form writing. The experimental results show that full ARRCE is higher than the response-SFT baseline on all four reported benchmarks. The ablation experiment section analyzes the three contribution ablations and the training-protocol control through direct comparisons.

## 2 Related Work

**Rubric-based rewards for open-ended reinforcement learning.** RLHF and Constitutional AI provide feedback for open-ended responses through human preferences or written principles, while DPO, SimPO, Self-Rewarding Language Models, and RLAIF reduce dependence on instance-level human preferences through preference optimization, simpler objectives, self-evaluation, or AI feedback (Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023; Meng et al., 2024; Yuan et al., 2024; Lee et al., 2024). These approaches usually compress feedback into response-level preferences or scalar rewards, which makes policy optimization stable but leaves unclear which observable behavior improved and how to handle a single query that simultaneously imposes factual, formatting, and coverage requirements. Text2Reward and Eureka generate rewards with language models or program search, automated process verifiers refine supervision to reasoning steps, and robust reward-model training studies how to reduce reward distortion under distribution shift (Xie et al., 2024; Ma et al., 2024; Setlur et al., 2025; Liu et al., 2025a; Lightman et al., 2024; Wang et al., 2024). For open-domain tasks, the policy gradually exposes behavioral differences in evidence selection, citation, organization, and tool use that are not fully enumerated in the initial task description, so a purely pre-specified rubric can miss useful process signals. ARRCE addresses this gap with decoupled reward construction: an offline synthesizer creates and freezes an answer contract and fact rubrics from task requirements and verifiable sources, while the online policy proposes process rubrics from its own observable trajectories; the static rubrics define the direction of trajectory com-

parisons and constrain candidate rewards, so process adaptation remains anchored to a stable task objective.

**Evolving rubrics and policy–rubric co-evolution.** Dynamic-rubric methods allow evaluation criteria to change with a task or policy behavior; Rubicon, DR Tulu, EvoRubric, EvoLM, EvoRubrics, and related self-improvement mechanisms explore retrieval-augmented criterion generation, rubric updates in deep research, joint training of responses and criteria, and rubric optimization by discrimination, alignment, and improvement (Gunjal et al., 2026; Huang et al., 2025; Mei et al., 2026; Shao et al., 2026; Guan et al., 2026; Li et al., 2026; Ding et al., 2026; Wang et al., 2026; Sheng et al., 2026; Tian et al., 2026; Gandhi et al., 2026). Co-evolution can discover behaviors absent from the initial rubric but also causes evaluator drift, so rubric quality cannot be judged from self-scores on the generating trajectories alone and requires an independent reference, held-out evaluation, and structural constraints. LLM-Rubric, Prometheus, and Prometheus 2 study multidimensional rubrics, interpretable feedback, and judge calibration, while RM-Bench, RewardBench, RMB, MT-Bench, JudgeBench, and WildBench test automatic evaluators through reward-model robustness, out-of-distribution preferences, downstream alignment correlation, dialogue capability, judging consistency, and real-task coverage (Hashemi et al., 2024; Kim et al., 2024a;b; Liu et al., 2025b; Lambert et al., 2025; Zhou et al., 2025; Zheng et al., 2023; Tan et al., 2025; Lin et al., 2025). ARRCE makes proposal and evaluation trajectories explicitly disjoint and uses frozen static scores to determine a candidate's direction; only criteria that pass held-out evaluation, static-boundary checks, and a non-degeneracy mask enter query-specific delayed memory, preventing a criterion from certifying itself on the same samples.

**Process supervision and rubric-based evaluation.** Process supervision extends reward beyond the final answer to intermediate steps; Let's Verify Step by Step and Math-Shepherd show how step-level verification or process rewards can improve difficult reasoning, often relying on mathematical correctness, executable checks, or domain-specific annotations (Lightman et al., 2024; Wang et al., 2024). For open-domain agents, ToolLLM, AgentBench, WebArena, and tau-bench extend evaluation to tool choice, argument passing, web interaction, long-horizon state, and task completion, while Search-R1 incorporates retrieval decisions into reinforcement-learning trajectories (Qin et al., 2024; Liu et al., 2024; Zhou et al., 2024; Yao et al., 2025; Jin et al., 2025). ResearchQA, HealthBench, WritingBench, FollowBench, LLMEval-Med, and LongWriter-Zero cover scientific question answering, healthcare safety, long-form writing, instruction following, clinical evaluation, and ultra-long generation, providing independent test criteria for open-domain responses (Yifei et al., 2025; Arora et al., 2025; Wu et al., 2025; Jiang et al., 2024; Zhang et al., 2025; Wu et al., 2026). ARRCE turns this gap into a learnable process-rubric problem: candidates describe only observable actions or answer properties in a trajectory and do not rely on hidden chain-of-thought; each candidate receives static supervision on independent trajectories and affects response rewards only after all hard requirements are satisfied.

# 3 METHOD

ARRCE separates the specification of task success from the discovery of useful response behavior (Figure 1). For each task $x$, an offline synthesizer constructs a frozen static rubric set $\mathcal{S}_x$. During reinforcement learning, the policy proposes process rubrics from its own trajectories; the static rubric set supplies both the trajectory comparison signal and the reward used to select and learn these proposals. A query-specific buffer $\mathcal{B}_x^t$ stores previously accepted process rubrics. The response policy $\pi_\theta$ and rubric proposer $G_\theta$ are two prompted modes of the same trainable model. The synthesis model and training judge are separate from these trainable parameters.

## 3.1 DECOUPLED STATIC AND PROCESS RUBRIC SYNTHESIS

The offline stage takes a task and its available documents. A language model first extracts an *answer contract*: atomic requirements, their strength, and whether they require external evidence. Query-derived requirements produce *task rubrics*, such as requested scope or format. Requirements needing evidence produce *fact rubrics* after retrieval. Each fact rubric includes a quotation that matches its cited source, while each task rubric specifies the requested behavior directly. Deterministic checks enforce contract membership, requirement strength, evidence validity, and exact

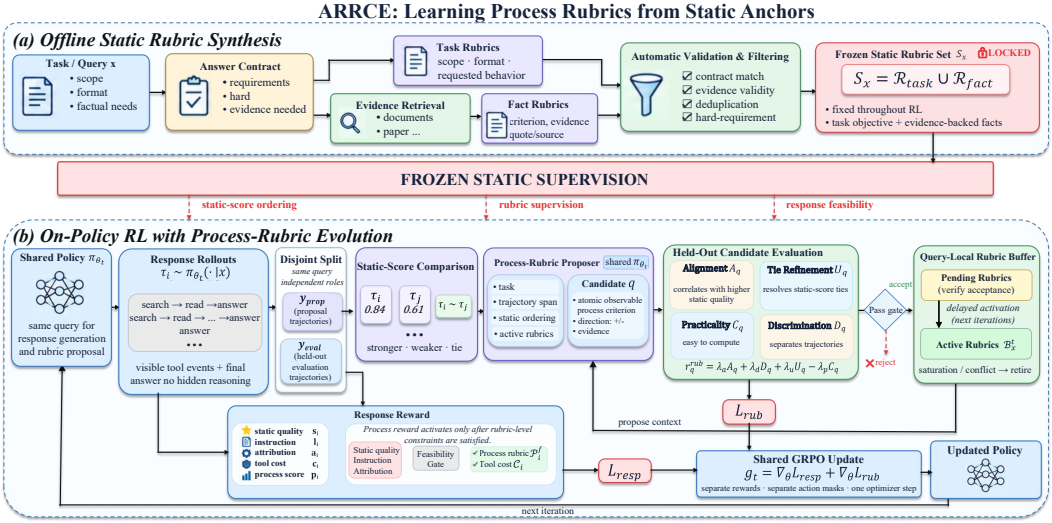



Figure 1: The ARRCE pipeline. The upper panel synthesizes task and fact rubrics offline and freezes their union as a static anchor. The lower panel samples response trajectories, forms disjoint proposal and evaluation subsets, and uses static scores to define strong, weak, and tie relations. The shared proposer generates process candidates; static-supervised candidate rewards include alignment, global discrimination, and a residual term conditioned on equal static aggregate scores. Accepted criteria enter a query-local pending buffer and become active on a later visit. Response and proposer losses accumulate at the same policy parameters and produce one shared optimizer step.

criterion deduplication. The training data select high-quality records with at least three rubrics, coverage of all hard requirements, and a required fact rubric when facts are applicable. Appendix A.1 gives the full contract, and Appendix A.6 lists the corresponding prompt template.

The resulting static set is frozen throughout RL. Process synthesis runs online and depends on current policy behavior, while the static set remains the task-specific reference for evaluating each proposed criterion. This separation lets process criteria adapt while preserving a stable task objective. With this method, we require no manual rubric writing for individual questions, trajectory-preference annotation, or candidate approval once the tasks, models, prompts, and filters are specified.

### 3.2 Constructing Trajectory Comparisons Without Preference Labels

At iteration $t$, sample $G$ trajectories $\tau_i \sim \pi_{\theta_t}(\cdot \mid x)$, including visible tool actions and a final answer. A frozen judge returns binary satisfaction scores $X_{ik}^S$ for static rubric $s_k \in \mathcal{S}_x$. The static score is

$$s_i = \frac{\sum_k w_k X_{ik}^S}{\sum_k w_k}. \tag{1}$$

With equal default weights, $s_i$ is the mean satisfaction over the task and fact criteria associated with $x$. This scalar provides a common scale for comparing trajectories from the same query and for defining the direction of each proposal pair.

The valid rollout group is deterministically divided into disjoint proposal and evaluation subsets, $\mathcal{Y}_t^{\text{prop}}$ and $\mathcal{Y}_t^{\text{eval}}$. For every unordered pair in the proposal subset, $G_{\theta_t}$ generates one atomic process rubric. The context contains the task, projected trajectories, their static scores and relative ordering, together with the text of all currently used static and process rubrics. A higher-scoring trajectory is designated stronger under the static signal; equal scores remain ties. The procedure uses all pairs, including ties, and the usual $G = 8$ and $|\mathcal{Y}_t^{\text{prop}}| = 4$ yield six proposals and four evaluation trajectories per task.

The proposer observes the trajectory fields specified by its prompt and receives aggregate static scores with their ordering. A candidate specifies an atomic criterion and a positive or negative

direction. Candidate evaluation uses the disjoint evaluation subset, and the exact proposal, judging, and validation contracts are listed in Appendix A.6.

### 3.3 Rewarding Process Rubrics With Static Rubrics

For candidate $q$, the judge evaluates the static criteria and $q$ jointly on each of the $n$ evaluation trajectories. Let $v_i(q) \in \{0,1\}$ denote satisfaction of $q$, and let $\widetilde{s}_i$ be the weighted static score from this evaluation call. In addition to global non-degeneracy, we compute a residual term only on pairs with equal static aggregate scores; this rewards process criteria that refine distinctions left unresolved by the static score. Define

$$p_q = \frac{1}{n}\sum_i v_i(q), \qquad \Delta_q = \mathrm{mean}_{v_i=1}\,\widetilde{s}_i - \mathrm{mean}_{v_i=0}\,\widetilde{s}_i, \tag{2}$$

$$D_q = 4p_q(1-p_q), \qquad \mathcal{T}_q = \{(i,j) : i<j,\ \widetilde{s}_i = \widetilde{s}_j\}, \tag{3}$$

$$U_q = \begin{cases} \dfrac{1}{|\mathcal{T}_q|}\displaystyle\sum_{(i,j)\in\mathcal{T}_q} |v_i(q) - v_j(q)|, & |\mathcal{T}_q| > 0, \\ 0, & |\mathcal{T}_q| = 0. \end{cases} \tag{4}$$

An empty conditional-group mean is assigned zero. If no equal-score pair exists, $U_q = 0$ and the candidate is still scored. The alignment gap measures whether trajectories satisfying the process rule have higher static quality. $D_q$ assigns zero global discrimination to a constant verdict, whereas $U_q$ measures refinement within static-score ties. Equality uses the weighted scores computed in the candidate-evaluation call. Appendix A.3 derives the exact behavior of constant rules and proves proposer and response invariance under the mask used by the algorithm.

A validator checks five properties: a valid atomic rule, observable behavior, respect for the static boundary, a useful response principle, and avoidance of proxies such as verbosity or hidden reasoning. Their conjunction is $V_q$. With duplication penalty $P_q$, the default reward is

$$r_q^{\mathrm{rub}} = \begin{cases} \lambda_\Delta\,\mathrm{clip}(\Delta_q, 0, \Delta_{\max}) + \lambda_D D_q + \lambda_U U_q - \lambda_{\mathrm{dup}} P_q + b_q, & V_q = 1,\ \Delta_q \geq -\epsilon_S, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Here $b_q$ is a format bonus for successfully parsed candidates. The default coefficients and optional validator-based terms appear in Appendix A.2. A candidate enters the pending buffer only when it passes the gates, reaches the activation threshold, and has $a_q = 1$.

To make constant verdicts inert in both learning branches, we apply a non-degeneracy mask after candidate evaluation. A constant candidate is still evaluated and its diagnostics are retained, but a zero mask sets its learning reward to zero and prevents it from entering the proposer batch, pending buffer, or active buffer. Define

$$\chi_q = \mathbf{1}[D_q + U_q > 0], \qquad r_{q,\mathrm{mask}}^{\mathrm{rub}} = \chi_q r_q^{\mathrm{rub}}, \qquad a_q = \chi_q \mathbf{1}[r_q^{\mathrm{rub}} \geq r_{\min}], \tag{6}$$

where $a_q$ is the activation indicator and $r_{\min}$ is the configured activation threshold. The mask is applied after all candidate-reward terms, including validator-based terms, have been combined.

### 3.4 Response Learning and Delayed Rubric Activation

Responses receive a static score $s_i$, instruction score $I_i$, attribution score $A_i$ when citations are required, and normalized tool-cost score $C_i$. Let $\mathcal{K}_i$ index the enabled hard dimensions, with $u_i = (s_i, I_i, A_i)$, weights $a_k$, and thresholds $\tau_k$. Define

$$H_i = \frac{\sum_{k\in\mathcal{K}_i} a_k u_{ik}}{\sum_{k\in\mathcal{K}_i} a_k}, \qquad z_i = \prod_{k\in\mathcal{K}_i} \mathbf{1}[u_{ik} \geq \tau_k], \tag{7}$$

$$\bar{H}_i = \mathrm{clip}\left(\frac{H_i - \bar{\tau}_i}{1 - \bar{\tau}_i}, 0, 1\right). \tag{8}$$

where $\bar{\tau}_i$ is the correspondingly weighted threshold. Only active process rubrics that pass the non-degeneracy mask contribute to the process score; when none is available, $P_i^t$ is set to zero. With

trajectory-format indicator $f_i$, the response reward is

$$P_i^t = \begin{cases} \dfrac{\sum_{q \in \mathcal{B}_x^t} \chi_q w_q v_i(q)}{\sum_{q \in \mathcal{B}_x^t} \chi_q w_q}, & \sum_{q \in \mathcal{B}_x^t} \chi_q w_q > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

and the response reward is

$$r_i^{\text{resp}} = \begin{cases} H_i - 1 + b_{\text{resp}} f_i, & z_i = 0, \\ 1 + \alpha \bar{H}_i + \beta P_i^t - \lambda_C C_i + b_{\text{resp}} f_i, & z_i = 1. \end{cases} \tag{10}$$

Process scores are applied after all enabled hard dimensions meet their thresholds. Attribution is removed and the remaining weights are renormalized when the judge identifies a task as not requiring citations.

Both response and proposer updates use the clipped GRPO policy objective. For a proposer group, the reward mean and standard deviation are computed only over candidates with $\chi_q = 1$; if a proposer group has no non-degenerate candidate, its proposer loss is skipped. Response groups are formed per query, while proposer groups are formed per shared proposal context. Under the default setting, each proposal context samples one candidate, so the proposer advantage of a non-degenerate candidate is determined directly by its masked reward; when a context contains multiple candidates, group normalization still uses only non-degenerate candidates. For a group $x$, let $\mathcal{I}_x$ denote the outputs included in its objective: all response outputs, or only the proposer outputs with $\chi_q = 1$. GRPO (Shao et al., 2024) maximizes

$$\mathcal{J}_x(\theta) = \frac{1}{|\mathcal{I}_x|} \sum_{i \in \mathcal{I}_x} \min\Big(\rho_i(\theta)\hat{A}_i, \text{clip}(\rho_i(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\,\hat{A}_i\Big) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\,. \tag{11}$$

Here $\hat{A}_i = (R_i - \mu_x)/(\sigma_x + \epsilon_{\text{std}})$ and $\rho_i(\theta) = \pi_\theta(y_i \mid x)/\pi_{\text{old}}(y_i \mid x)$, where $\mu_x$ and $\sigma_x$ are computed over $\mathcal{I}_x$. The ratio is evaluated on generated action tokens under the corresponding mask. Response and proposer batches keep separate rewards and old-policy denominators, accumulate gradients at $\theta_t$, and make one shared optimizer step. Newly accepted rubrics are staged after this update and become eligible on a later visit to the same query. Appendix A.4 gives the iteration rules.

At evaluation time, the trained response policy generates answers from benchmark prompts and the configured tools. Benchmark evaluators then score these outputs separately from the training-time rubric proposal and reward process.

# 4 Experiments

The section follows four parts: experimental setup, experimental results, ablation studies, and case study. The $\pm$ terms denote standard deviations.

## 4.1 Experimental Setup

**Data and evaluation.** The completed static-rubric corpus contains 3,696 tasks from HealthBench, ResearchQA, and LongWriter-Zero (Arora et al., 2025; Yifei et al., 2025; Wu et al., 2026): 2,623 allocated to response SFT and 1,073 to RL. The RL allocation contains 294, 395, and 384 tasks from these sources, respectively. Static criteria and response trajectories are distinct artifacts. Following the DR-Rubric setup (Mei et al., 2026), training uses a mock-search tool; real Tavily search is reserved for the supplementary search-environment check in Appendix B.6. Appendix B.1 describes the corpus construction and verified counts.

**Models and inference settings.** Unless stated otherwise, each experimental profile fixes the policy model, initialization checkpoint, and tokenizer and uses a Qwen3.6-27B judge (Qwen Team, 2026; Yang et al., 2025) for both training and benchmark evaluation; extended reasoning is disabled for the training judge. The judge-sensitivity profile changes only the training judge to DeepSeek-V4-Flash (DeepSeek-AI et al., 2026) while holding the policy, data, search, and benchmark-evaluation protocol fixed. Training uses mock search with at most three returned documents per query and a 1,500-character limit per document.

Table 1: Experimental results and training-judge sensitivity. The V4-Flash row uses DeepSeek-V4-Flash as the training judge. LLMMED denotes LLMEval-Med and reports its overall performance (OP) percentage; RQA is a normalized rubric score, WB uses the 1–10 scale, and FB is the overall FollowBench score. Horizontal rules group the base/SFT baselines, static/dynamic RL variants, EvoRubric and DR Tulu/RLER variants, and ARRCE variants. The table reports experimental means; bold and underlined entries mark the highest and second-highest means for each benchmark, respectively. $\pm$ denotes standard deviation. Higher is better for every metric. Appendix 9 reports detailed results for the primary profiles, and Appendix 8 reports the mock/Tavily search comparison.

| Method | RQA ↑ | LLMMED ↑ | WB ↑ | FB ↑ |
|---|---|---|---|---|
| Base model | $0.697 \pm 0.0030$ | $67.02 \pm 0.0507$ | $7.252 \pm 0.0410$ | $0.443 \pm 0.0166$ |
| Response SFT | $0.734 \pm 0.0030$ | $68.37 \pm 0.0512$ | $7.452 \pm 0.0438$ | $0.495 \pm 0.0169$ |
| Static additive RL | $0.778 \pm 0.0028$ | $69.42 \pm 0.0512$ | $7.639 \pm 0.0445$ | $0.523 \pm 0.0169$ |
| Static gated RL | $0.764 \pm 0.0030$ | $71.21 \pm 0.0500$ | $7.646 \pm 0.0425$ | $0.528 \pm 0.0169$ |
| Dynamic additive RL | $0.779 \pm 0.0028$ | $70.76 \pm 0.0512$ | $7.682 \pm 0.0431$ | $0.526 \pm 0.0170$ |
| EvoRubric-adapted | $0.776 \pm 0.0029$ | $69.27 \pm 0.0506$ | $7.656 \pm 0.0436$ | $0.523 \pm 0.0169$ |
| DR Tulu/RLER | $0.780 \pm 0.0028$ | $70.91 \pm 0.0506$ | $7.632 \pm 0.0434$ | $0.518 \pm 0.0168$ |
| ARRCE | $\underline{0.791 \pm 0.0027}$ | $\underline{71.36 \pm 0.0507}$ | $\underline{7.700 \pm 0.0436}$ | $\underline{0.535 \pm 0.0169}$ |
| ARRCE(V4-Flash judge) | $\mathbf{0.796 \pm 0.0026}$ | $\mathbf{72.14 \pm 0.0502}$ | $\mathbf{7.717 \pm 0.0420}$ | $\mathbf{0.537 \pm 0.0165}$ |

**Evaluation protocol.** The benchmark experiments sample one generation per query, use temperature 1.0 and top-$p$ 1.0 for rollout, use judge temperature 1.0, permit up to 65,536 judge tokens, retry failed requests up to three times, and allow 16 concurrent judge requests.

**Benchmark scope.** Main metrics are normalized ResearchQA rubric coverage (Yifei et al., 2025), LLMEval-Med overall performance (OP) (Zhang et al., 2025), WritingBench writing quality (Wu et al., 2025), and FollowBench constraint satisfaction (Jiang et al., 2024). We abbreviate LLMEval-Med as LLMMED in tables; its OP is the overall usable-performance percentage. ResearchQA measures same-source generalization, and WritingBench measures cross-dataset writing transfer.

**Comparisons and controls.** Table 1 specifies the primary comparison. The comparison matrix includes a base model, response SFT, static additive RL, static gated RL, dynamic additive RL, full ARRCE, an EvoRubric-adapted profile, a DR Tulu/RLER profile, and a judge-sensitivity profile that changes only the training judge. Static additive and static gated RL isolate static supervision and response gating; the remaining profiles provide process-rubric, external-method, and judge-robustness reference points.

**Statistical reporting.** The main comparison reports benchmark results for decoupling, static supervision, and score-based trajectory controls. We report the experimental means and standard deviations for each profile; failed judgments and coverage accompany the displayed scores. Appendix B specifies configuration and failure reporting.

### 4.2 Experimental Results

**Observed results.** The response-SFT baseline is $0.734$ on ResearchQA, $68.37$ OP on LLMEval-Med, $7.452$ on WritingBench, and $0.495$ on FollowBench. Relative to this baseline, full ARRCE improves ResearchQA by $0.057$, LLMEval-Med by $2.99$ OP points, WritingBench by $0.248$, and FollowBench by $0.040$. All four benchmarks therefore show positive changes, although the scale and interpretation remain benchmark dependent: LLMEval-Med uses a percentage OP scale, while the WritingBench gain is measured on a 1–10 scale. The comparison profiles still show complementary strengths: static gated reaches the highest comparison-profile LLMEval-Med OP ($71.21$), the V4-Flash judge profile reaches the highest WritingBench mean ($7.717$), and static additive reaches $0.523$ on FollowBench, so the full ARRCE profile is not uniformly best on every metric. The appendices report detailed experimental results and the separate mock/Tavily search comparison for full ARRCE.

Table 2: Contribution ablations and a training-protocol control, grouped by full ARRCE reference, stage decoupling, static supervision, automatic trajectory comparison, candidate-evaluation split, and proposer-objective control. Variants from Coupled synthesis through Source-response evaluation are contribution ablations; No proposer objective is the training-protocol control. LLMMED abbreviates LLMEval-Med; parentheses report absolute differences from full ARRCE.

| Variant | RQA | LLMMED | WB | FB |
|---|---|---|---|---|
| Full ARRCE | 0.791 | 71.36 | 7.700 | 0.535 |
| Coupled synthesis | 0.774 (−0.017) | 70.57(−0.79) | 7.563 (−0.137) | 0.521 (−0.014) |
| No alignment | 0.766 (−0.025) | 70.02(−1.34) | 7.598 (−0.102) | 0.524 (−0.011) |
| No alignment + gate | 0.751 (−0.040) | 69.96(−1.40) | 7.524 (−0.176) | 0.510 (−0.025) |
| No tie residual | 0.749 (−0.042) | 69.82(−1.54) | 7.504 (−0.196) | 0.511 (−0.024) |
| Hidden score labels | 0.758 (−0.033) | 70.68(−0.68) | 7.556 (−0.144) | 0.515 (−0.020) |
| Shuffled score labels | 0.744 (−0.047) | 69.03(−2.33) | 7.500 (−0.200) | 0.508 (−0.027) |
| Source-response eval. | 0.782 (−0.009) | 70.76 (−0.60) | 7.601 (−0.099) | 0.531 (−0.004) |
| No proposer objective | 0.780 (−0.011) | 69.27 (−2.09) | 7.626 (−0.074) | 0.516 (−0.019) |

**Training-judge sensitivity.** Replacing the training judge with DeepSeek-V4-Flash improves full ARRCE with the default Qwen3.6-27B judge by $0.005$ on ResearchQA, $0.78$ OP points on LLMEval-Med, $0.017$ on WritingBench, and $0.002$ on FollowBench, corresponding to relative changes of approximately $0.6\%$, $1.1\%$, $0.2\%$, and $0.4\%$. This profile is highest among the listed configurations on all four benchmarks. Its reported standard deviations are also slightly smaller on all four benchmarks.

### 4.3 Ablation Studies

**Ablation design.** Table 2 separates contribution ablations from a training-protocol control. The contribution ablations test static/process synthesis decoupling, static-supervision signals including the proposal/evaluation trajectory split, and automatic trajectory comparison. The training-protocol control tests the proposer learning objective. The coupled control synthesizes static and process criteria together from the same trajectory context with matched synthesis and scoring budgets. The static-supervision controls remove the alignment reward and then the direction gate while keeping candidate-validity checks and discrimination terms fixed.

For automatic trajectory comparison, we hid the numerical static scores and stronger/weaker/tie labels from the proposer, or permuted those scores and recomputed consistent labels within each proposal subset. Both controls retained the same trajectories, pair enumeration, and true static scores for candidate evaluation.

The training-protocol control is included directly in Table 2. The source-response evaluation ablation uses the same responses for proposal and evaluation, whereas the proposer-objective control removes the proposer learning objective while retaining the online candidate and response-learning components.

**Stage decoupling.** Coupled static/process synthesis reduces the full scores of $0.791$ (ResearchQA), $71.36$ OP (LLMEval-Med), $7.700$ (WritingBench), and $0.535$ (FollowBench) to $0.774$, $70.57$ OP, $7.563$, and $0.521$. The corresponding drops are $0.017$, $0.79$ OP points, $0.137$, and $0.014$, indicating that stage separation supports both source-task quality and transfer across evaluation types.

**Static supervision.** Removing only the alignment reward gives $0.766$ on ResearchQA, $70.02$ OP on LLMEval-Med, $7.598$ on WritingBench, and $0.524$ on FollowBench. Removing the direction gate as well lowers these scores to $0.751$, $69.96$ OP, $7.524$, and $0.510$. Relative to full ARRCE, the WritingBench drops are $0.102$ and $0.176$, while the LLMEval-Med drops are $1.34$ and $1.40$ OP points; the additional gate intervention costs $0.074$ on WritingBench and $0.06$ OP points on

Table 3: Case-study comparison. Only the evidence-bearing differences between the response-SFT and full ARRCE trajectories are shown; the complete case appears in Appendix B.7.

| | **Response SFT** | **Full ARRCE** |
|---|---|---|
| Static coverage | 0.50 (S1, S4) | 1.00 (S1–S4) |
| Search difference | Broad association queries; one off-target meningococcal query. | Progresses from association to populations, regions, effect sizes, and confounders. |
| Missing / recovered evidence | No odds ratio or concrete adjusted covariates; generic positive-association summary. | Adjusted odds ratio $\approx$ 1.8, named behavioral factors, population/region comparisons, and design limitations. |

LLMEval-Med, suggesting complementary roles for continuous static consistency and discrete rejection.

**Static-tie residual.** Removing the static-tie residual gives $0.749$ on ResearchQA, $69.82$ OP on LLMEval-Med, $7.504$ on WritingBench, and $0.511$ on FollowBench. Relative to full ARRCE, the drops are $0.042$, $1.54$ OP points, $0.196$, and $0.024$, respectively. All four benchmarks decline, with larger changes on WritingBench and LLMEval-Med, indicating that the residual provides useful refinement when aggregate static scores are tied.

**Relative-quality supervision.** Hiding proposer scores and ordering labels gives $0.758$, $70.68$ OP, $7.556$, and $0.515$ on ResearchQA, LLMEval-Med, WritingBench, and FollowBench. Shuffling these labels lowers the scores to $0.744$, $69.03$ OP, $7.500$, and $0.508$. Relative to full ARRCE, the WritingBench drops are $0.144$ and $0.200$, while the LLMEval-Med drops are $0.68$ and $2.33$ OP points, indicating that reliable relative-quality information is especially important for the proposer.

**Held-out candidate evaluation.** Evaluating candidates on source responses lowers ResearchQA from $0.791$ to $0.782$, WritingBench from $7.700$ to $7.601$, and FollowBench from $0.535$ to $0.531$; LLMEval-Med OP falls from $71.36$ to $70.76$. The largest change is on WritingBench, supporting held-out evaluation as a source of candidate generalization beyond the trajectories used for proposal.

**Proposer objective.** Removing the proposer objective lowers ResearchQA from $0.791$ to $0.780$, LLMEval-Med OP from $71.36$ to $69.27$, WritingBench from $7.700$ to $7.626$, and FollowBench from $0.535$ to $0.516$. The $2.09$-point LLMEval-Med drop indicates that the explicit proposer objective contributes across the reported external metrics.

### 4.4 Case Study

We use a representative open-domain question-answering case to illustrate ARRCE's behavioral change: the pre-training response gives a broad positive association, whereas the trained response retrieves populations, regions, infection types, effect sizes, and behavioral factors, reports an adjusted estimate, and distinguishes association from causation. Static rubric coverage rises from $0.50$ to $1.00$; the complete case and audit appear in Appendix B.7.

## 5 Limitations and Conclusion

**Limitations.** Static supervision may be incomplete, and model-based evaluation may share errors or favor superficial properties. Shared parameters couple response and proposer updates, while synthetic and live retrieval define different environments.

**Conclusion.** ARRCE separates static anchors from evolving process rubrics, uses static scores to supervise proposal, and activates validated criteria as response rewards. It improves response-SFT by $0.057$ (RQA; $+7.8\%$), $2.99$ OP points (LLMMED; $+4.4\%$), $0.248$ (WB; $+3.3\%$), and $0.040$ (FB; $+8.1\%$), respectively, with task-dependent gains.

## AI USE STATEMENT

Generative models are components of the research method: they synthesize answer contracts, static rubrics, response trajectories, and process rubrics, and provide criterion-level evaluations. AI assistance was also used to inspect local research materials, organize the manuscript, draft text and equations, retrieve reference metadata, and develop experiment designs. These uses include synthetic-data generation and assistance with methodology and experiment design, which require disclosure under the ICLR author policy.

## ETHICS STATEMENT

Healthcare tasks are used to evaluate response quality, not to establish clinical safety or suitability for deployment. Generated criteria and automated judgments can retain errors, cultural biases, and limitations of their source data and models. Evidence retrieval does not ensure source reliability, and agreement with a synthetic rubric does not ensure factual correctness. Release and evaluation must respect the licenses, privacy constraints, and permitted uses of the underlying datasets and retrieved material. The proposed synthesis loop requires no new human annotation; any later human evaluation should be disclosed separately and conducted under the applicable institutional requirements.

## REPRODUCIBILITY STATEMENT

The method section and supplementary methodological description specify the data flow, candidate scoring, response reward, optimization, and rubric lifecycle. The experimental appendix records dataset counts, split statistics, experimental results, training configuration, and statistical procedures. An anonymized code repository for this submission is available at `https://anonymous.4open.science/r/ARRCE-5C5A`.

## REFERENCES

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, et al. HealthBench: Evaluating Large Language Models Towards Improved Human Health, 2025. URL https://arxiv.org/abs/2505.08775.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI Feedback, 2022. URL https://arxiv.org/abs/2212.08073.

DeepSeek-AI, Anyi Xu, Bangcai Lin, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Hongxin Ding, Baixiang Huang, Yue Fang, et al. EvoRubrics: Dynamic Rubrics as Rewards via Adversarial Co-Evolution for LLM Reinforcement Learning, 2026. URL https://arxiv.org/abs/2606.23038.

Shubham Gandhi, Saurabh Goyal, Kiran Kate, and Yara Rizk. DRACO: Fine-Grained Credit Assignment with Dynamic Rubrics for Long-Horizon Agent Training, 2026. URL https://arxiv.org/abs/2609.04094.

Xin Guan, Xiaomeng Hu, Shen Huang, et al. EvoRubric: Self-Evolving Rubric-Driven RL for Open-Ended Generation, 2026. URL https://arxiv.org/abs/2605.29847.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 127924–127945, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/cfd7bee7a651ee9af525098ef67a9e45-Paper-Conference.pdf.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language

texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.745. URL https://aclanthology.org/2024.acl-long.745/.

Zenan Huang, Yihong Zhuang, Guoshan Lu, et al. Reinforcement Learning with Rubric Anchors, 2025. URL https://arxiv.org/abs/2508.12790.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.257. URL https://aclanthology.org/2024.acl-long.257/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, et al. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Seungone Kim, Jay Shin, Yejin Cho, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/803485352e61e3ebf41221e4776c9fd4-Abstract-Conference.html.

Seungone Kim, Juyoung Suk, Shayne Longpre, et al. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, 2024b. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, et al. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.96. URL https://aclanthology.org/2025.findings-naacl.96/.

Harrison Lee, Samrat Phatale, Hassan Mansoor, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 26874–26901. PMLR, 2024. URL https://proceedings.mlr.press/v235/lee24t.html.

Shuyue Stella Li, Rui Xin, Teng Xiao, et al. EvoLM: Self-Evolving Language Models through Co-Evolved Discriminative Rubrics, 2026. URL https://arxiv.org/abs/2605.03871.

Hunter Lightman, Vineet Kosaraju, Yura Burda, et al. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, et al. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/771155abaae744e08576f1f3b4b7ac0d-Abstract-Conference.html.

Tianqi Liu, Wei Xiong, Jie Ren, et al. Rrm: Robust reward model training mitigates reward hacking. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/9d46574e5baae5121180228223a11836-Abstract-Conference.html.

Xiao Liu, Hao Yu, Hanchen Zhang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Yantao Liu, Zijun Yao, Rui Min, et al. Rm-bench: Benchmarking reward models of language models with subtlety and style. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=QEHrmQPBdd.

Yecheng Jason Ma, William Liang, Guanzhi Wang, et al. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/70c26937fbf3d4600b69a129031b66ec-Abstract-Conference.html.

Wangyi Mei, Zhouhong Gu, Zhenhan Bai, et al. Deep Research as Rubric for Reinforcement Learning, 2026. URL https://arxiv.org/abs/2606.01091.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e099c1c9699814af0be873a175361713-Abstract-Conference.html.

Long Ouyang, Jeff Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Yujia Qin, Shihao Liang, Yining Ye, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

Rafael Rafailov, Archit Sharma, Eric Mitchell, et al. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Amrith Setlur, Chirag Nagpal, Adam Fisch, et al. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, et al. DR Tulu: Reinforcement Learning with Evolving Rubrics for Deep Research, 2026. URL https://arxiv.org/abs/2511.19399.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

Leheng Sheng, Wenchang Ma, Ruixin Hong, et al. Reinforcing Chain-of-Thought Reasoning with Self-Evolving Rubrics, 2026. URL https://arxiv.org/abs/2602.10885.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, et al. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/9e720fce64f91114c49cfd640d821da3-Abstract-Conference.html.

Zihang Tian, Jingsen Zhang, Rui Li, et al. ARCO: Adaptive Rubrics with Co-Evolution for Multi-Step LLM-Based Agents, 2026. URL https://arxiv.org/abs/2606.21262.

Jianze Wang, Kunwang Zheng, Ying Liu, et al. SERPO: Self-Evolving Rubric Policy Optimization for Open-Ended Test-Time Reinforcement Learning, 2026. URL https://arxiv.org/abs/2607.26873.

Peiyi Wang, Lei Li, Zhihong Shao, et al. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yuhao Wu, Yushi Bai, Zhiqiang Hu, Roy Ka-Wei Lee, and Juanzi Li. LongWriter-Zero: Mastering Ultra-Long Text Generation via Reinforcement Learning, 2026. URL https://arxiv.org/abs/2506.18841.

Yuning Wu, Jiahao Mei, Ming Yan, et al. WritingBench: A Comprehensive Benchmark for Generative Writing, 2025. URL https://arxiv.org/abs/2503.05244.

Tianbao Xie, Siheng Zhao, Chen Wu, et al. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9941833e8327910ef25daeb9005e4748-Abstract-Conference.html.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. tau-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

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, 2025. URL https://arxiv.org/abs/2509.00496.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, et al. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 57905–57923. PMLR, 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Ming Zhang, Yujiong Shen, Zelin Li, et al. LLMEval-Med: A Real-world Clinical Benchmark for Medical LLMs with Physician Validation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 4888–4914, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.263. URL https://aclanthology.org/2025.findings-emnlp.263/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Enyu Zhou, Guodong Zheng, Binghai Wang, et al. Rmb: Comprehensively benchmarking reward models in llm alignment. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/427f20d90386fd27804f1831d6a3d48f-Abstract-Conference.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

# A METHOD DETAILS

## A.1 STATIC SYNTHESIS CONTRACT

The static synthesis stage starts from the task text and any available source documents. An answer contract contains a task goal, a Boolean indicating whether factual evidence is applicable, and a list of requirements. Each requirement has an identifier, a description, a strength in {hard, required, optional, conditional}, an indicator for retrieval, and a search query when appropriate. The synthesizer then produces separate task-rubric and fact-rubric lists, which are merged into the frozen static set. Both types store their requirement identifier, criterion, and strength. Fact rubrics additionally store source identifiers and quotations.

Validation checks that every rubric maps to an existing requirement and preserves its strength. Task rubrics must map to requirements that do not need retrieval and must not contain evidence. Fact rubrics must map to retrieval requirements and contain evidence from known documents. Quotations must be exact source substrings; a quotation differing only in whitespace can be restored to the original source span when the match is unique. Case-insensitive exact duplicate criteria are removed. These checks establish structural and quotation validity; they do not establish that a source is correct or that a quoted passage entails the criterion.

The automatic quality filter excludes records with an empty rubric set, fewer than three criteria, a missing hard requirement, or no hard/required fact rubric when factual evidence is applicable. Training uses records that pass this filter. The policy input contains the task, while rubrics and evidence are supplied as reward-side metadata; source documents are available through the search environment.

## A.2 REWARD DETAILS AND DEFAULT PARAMETERS

The alignment in Equation 2 first computes a conditional difference for every static column,

$$m_{qk} = \mathrm{mean}_{v_i(q)=1}\, X^S_{ik} - \mathrm{mean}_{v_i(q)=0}\, X^S_{ik}, \qquad \Delta_q = \frac{\sum_k w_k m_{qk}}{\sum_k w_k}. \tag{12}$$

This equals the difference of conditional weighted static means used in the main text. Both conditional means use zero for an empty group. Thus an always-passing rule has $\Delta_q = \mathrm{mean}_i\, \widetilde{s}_i$, whereas an always-failing rule has its negative. The raw score does not require both groups to be nonempty and does not reject a constant rule solely for zero discrimination. For a nonempty evaluation set, it computes a raw reward even when $n = 1$. Constant rules can consequently pass the raw acceptance test and receive format or alignment terms; the algorithmic mask then keeps them inert.

The optional general form adds $\lambda_{\mathrm{stab}} S_q + \lambda_{\mathrm{nov}} N_q$ inside the accepted branch of Equation 5; both coefficients are zero in the full configuration. The residual term is computed on the exact static-score ties only. Let

$$\mathcal{T}_q = \{(i,j) : i < j,\ \widetilde{s}_i = \widetilde{s}_j\}, \qquad D^{\mathrm{res}}_q = \begin{cases} \dfrac{1}{|\mathcal{T}_q|} \displaystyle\sum_{(i,j)\in\mathcal{T}_q} |v_i(q) - v_j(q)|, & |\mathcal{T}_q| > 0, \\ 0, & |\mathcal{T}_q| = 0. \end{cases} \tag{13}$$

We use $D^{\mathrm{res}}_q$ as the tie-conditioned component $U_q$ in Equation 5. Equal-score pairs are weighted equally, without first averaging within tie groups. The score uses all static task and fact columns with the configured static weights, and equality is exact after the weighted mean is computed. When no tie pair exists, the residual is zero. For example, static scores $[1, 1, 0, 0]$ give residual 0 to verdicts $[1, 1, 0, 0]$ and residual 1 to verdicts $[1, 0, 1, 0]$. The residual is invariant to flipping every process verdict, so direction still comes from $\Delta_q$ and the semantic validity gates.

## A.3 CONSTANT RULES: EXACT REWARD AND MASKED INVARIANCE

The scoring rule intentionally computes a reward for a constant candidate; it does not require both conditional groups to be nonempty. We first characterize this raw score, then prove the invariance induced by the algorithmic mask. A constant candidate is evaluated and retained for diagnostics, but

it is inert for activation and learning. Let $\bar{s} = n^{-1} \sum_{i=1}^{n} \widetilde{s}_i$ and let $q_c$ be a candidate with $v_i(q_c) = c$ for every evaluation trajectory, where $c \in \{0, 1\}$. With the empty-group convention used above,

$$p_{q_c} = c, \qquad D_{q_c} = 4c(1-c) = 0, \qquad U_{q_c} = 0, \tag{14}$$

$$\Delta_{q_c} = (2c-1)\bar{s}. \tag{15}$$

**Proof.** When $c = 1$, the passing group is the whole evaluation set and the failing group is empty, so $\Delta_{q_c} = \bar{s} - 0 = \bar{s}$. When $c = 0$, the passing group is empty and the failing group is the whole set, so $\Delta_{q_c} = 0 - \bar{s} = -\bar{s}$. The process verdict is identical for every trajectory, which gives $D_{q_c} = 0$ and zero disagreement on every static-score tie, hence $U_{q_c} = 0$.

Consequently, if

$$b_q^{\circ} = \lambda_{\text{stab}} S_q + \lambda_{\text{nov}} N_q - \lambda_{\text{dup}} P_q + b_q,$$

the unmasked candidate reward in Eq. 5 is exactly

$$r_{q_c}^{\text{rub}} = \mathbf{1}[V_{q_c} = 1,\ (c = 1)\ \vee\ (\bar{s} \le \epsilon_S)] \left[\lambda_\Delta\, \mathbf{1}[c = 1] \min(\bar{s}, \Delta_{\max}) + b_q^{\circ}\right]. \tag{16}$$

Thus a constant rule has no global discrimination or tie-refinement signal, but the unmasked raw score can still assign it a nonzero format, alignment, stability, or novelty reward. In particular, an always-passing rule is positively aligned whenever $\bar{s} > 0$, while an always-failing rule passes the direction gate only when $\bar{s} \le \epsilon_S$. Since the default proposer group contains one candidate per proposal context, its advantage is the scalar reward itself.

There is a separate, conditional invariance for response learning. Let $\mathcal{B}$ be the active process-rubric set before adding $q_c$, let $m = |\mathcal{B}|$, and let $P_{i,\text{raw}}^t$ denote the unmasked equal-weight process mean. Then

$$P_{i,\text{raw}}^t(\mathcal{B} \cup \{q_c\}) = \begin{cases} c, & m = 0, \\ \dfrac{m P_{i,\text{raw}}^t(\mathcal{B}) + c}{m+1}, & m > 0. \end{cases} \tag{17}$$

The induced raw response-reward change is therefore

$$r_{i,\text{raw}}^{\text{resp}}(\mathcal{B} \cup \{q_c\}) - r_{i,\text{raw}}^{\text{resp}}(\mathcal{B}) = \beta z_i \begin{cases} c, & m = 0, \\ \dfrac{c - P_{i,\text{raw}}^t(\mathcal{B})}{m+1}, & m > 0. \end{cases} \tag{18}$$

For a response group $\mathcal{G}$, GRPO is invariant to a common reward shift: if $R_i' = R_i + a$ for all $i \in \mathcal{G}$, then $\mu' = \mu + a$, $\sigma' = \sigma$, and $\hat{A}_i' = (R_i' - \mu')/(\sigma' + \epsilon_{\text{std}}) = \hat{A}_i$. Equation 18 is a common shift only in special cases, such as $m = 0$ with all trajectories hard-feasible, or $m > 0$ with both $z_i$ and $P_{i,\text{raw}}^t(\mathcal{B})$ constant over the response group. It is not generally a common shift when feasible and infeasible trajectories coexist, and with $m > 0$ even a constant verdict can change the relative rewards through the mean denominator. Thus the unmasked reward gives no unconditional response-invariance guarantee either. The GRPO calculation identifies the additional conditions needed for such a guarantee.

The algorithm uses the following non-degeneracy mask after all candidate-reward terms have been combined, so that no later reward component can reactivate a constant candidate:

$$\chi_q = \mathbf{1}[D_q + U_q > 0], \qquad r_{q,\text{mask}}^{\text{rub}} = \chi_q r_q^{\text{rub}}, \qquad a_q = \chi_q \mathbf{1}[r_q^{\text{rub}} \ge r_{\min}], \tag{19}$$

where $a_q$ is the activation indicator. The candidate is eligible for neither the pending nor active buffer when $a_q = 0$. The response process score uses only masked rules:

$$P_{i,\text{mask}}^t = \begin{cases} \dfrac{\sum_{q \in \mathcal{B}_x^t} \chi_q w_q v_i(q)}{\sum_{q \in \mathcal{B}_x^t} \chi_q w_q}, & \sum_{q \in \mathcal{B}_x^t} \chi_q w_q > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

**Masked invariance proof.** For a nonempty binary verdict vector, every constant rule satisfies $D_q + U_q = 0$. Hence $\chi_{q_c} = 0$, $r_{q_c,\text{mask}}^{\text{rub}} = 0$, and $a_{q_c} = 0$.

For the proposer branch, let $\mathcal{I}_x^+ = \{j : \chi_{q_j} = 1\}$ be the non-degenerate candidate indices in a proposal context. The proposer reward mean, standard deviation, and policy objective are computed

Table 4: Method defaults used by the full experiment configuration.

| Component | Parameter | Value |
|---|---|---|
| Rollout | Trajectories per query $G$ | 8 |
| Split | Proposal/evaluation trajectories | $4/4$ |
| Proposal | Candidates per pair / per query | $1/6$ |
| Split | Seed | 0 |
| Static score | Default criterion weights | 1 |
| Hard thresholds | Static, instruction, attribution | $0.7, 0.7, 0.7$ |
| Hard aggregate | Static, instruction, attribution weights | $0.6, 0.2, 0.2$ |
| Response reward | $\alpha, \beta, \lambda_C$ | $0.25, 0.5, 0.1$ |
| Format bonuses | $b_{\text{resp}}, b_q$ | $0.5, 0.5$ |
| Alignment | $\epsilon_S, \Delta_{\max}$ | $0.05, 0.25$ |
| Rubric reward | $\lambda_\Delta, \lambda_D, \lambda_U, \lambda_{\text{dup}}$ | $1, 1, 1, 1$ |
| Optional terms | $\lambda_{\text{stab}}, \lambda_{\text{nov}}$ | $0, 0$ |
| Activation | Minimum rubric reward | 0 |
| Retirement | Saturation interval boundaries | $0.05, 0.95$ |
| Retirement | Saturation/conflict patience | $3/2$ query visits |

**Algorithm 1** One ARRCE training iteration

**Require:** query $x$, policy parameters $\theta_t$, query-local buffer $\mathcal{B}_x$

1: Activate previously staged non-degenerate criteria; snapshot active set $\mathcal{B}_x^t$.
2: Sample responses $\mathcal{Y}_x$ with $\pi_{\theta_t}$; score static and active process criteria; discard invalid trajectories.
3: Split $\mathcal{Y}_x$ into disjoint $\mathcal{Y}_x^{\text{prop}}$ and $\mathcal{Y}_x^{\text{eval}}$; compute static scores and strong/weak/tie relations.
4: Generate one process candidate $q$ for each unordered pair in $\mathcal{Y}_x^{\text{prop}}$; retain action log probabilities.
5: Validate each $q$ and score it on $\mathcal{Y}_x^{\text{eval}}$; compute raw $r_q^{\text{rub}}$ and mask $\chi_q$ using alignment, global discrimination, and tie-conditioned residual.
6: Compute response rewards from masked $\mathcal{B}_x^t$ and form separate response/proposer GRPO batches using only non-degenerate candidates.
7: Accumulate both masked gradients at $\theta_t$ and perform one shared optimizer step.
8: Stage only candidates with $a_q = 1$ for a later visit; update query-local statistics and retire saturated or conflicting criteria.

only over $\mathcal{I}_x^+$; when this set is empty, the proposer loss is skipped. Adding or removing $q_c$ does not change $\mathcal{I}_x^+$, so it does not change the advantage or proposer gradient of any non-degenerate candidate.

For the response branch, both the numerator and denominator of $P_{i,\text{mask}}^t$ contain only rules with $\chi_q = 1$. Adding or removing a constant rule contributes zero to both quantities. If no non-degenerate rule is active, the denominator remains zero and the process score remains zero. Therefore every trajectory keeps the same response reward.

Combining the two branches, consistent use of the mask in candidate rewards, proposer group normalization, buffer activation, and response process scores makes every constant rule inert: it cannot change the proposer loss, proposer gradient, or response reward.

The hard score in Equation 10 is normalized by the sum of enabled weights. Citation-free tasks use static and instruction weights renormalized from $0.6$ and $0.2$. The denominator in $\bar{H}_i$ has a numerical lower bound of $10^{-12}$. The response format check requires a nonempty final answer and validates the observable event contract: tool arguments must parse as an object, and tool-result entries must provide nonempty source identifiers, titles, URLs, and content. The format bonus applies to both feasible and infeasible responses. Cost is a model-estimated score of unnecessary or invalid tool use.

### A.4 Iteration Order and Optimization

Algorithm 1 summarizes one training iteration; routine GRPO details are given by Equation 11.

Both branches instantiate Equation 11 with separate rewards, action masks, and rollout likelihoods. The response group is formed per query and the proposer group per proposal context; proposer-group normalization includes only non-degenerate candidates, and a group with none is skipped. The optional KL term is controlled by the actor configuration and is zero in the default full-method profile.

Let $\mathcal{L}_{\mathrm{resp}}$ and $\mathcal{L}_{\mathrm{rub}}$ be the actor's masked policy losses computed using their respective advantages and old likelihoods. The shared gradient is

$$g_t = \nabla_\theta \mathcal{L}_{\mathrm{resp}}(\theta_t) + \nabla_\theta \mathcal{L}_{\mathrm{rub}}(\theta_t). \tag{21}$$

The candidate reward is never added to the response reward, and no parameter update intervenes between the two backward passes. A proposer failure permits a response-only update; if the proposer backward pass fails, the trainer clears gradients and recomputes the response gradient before stepping. An entirely invalid response batch skips the actor update.

An active rule is saturated when its current pass rate is at most $0.05$ or at least $0.95$. It is retired after three consecutive observed visits satisfying this condition. When both passing and failing trajectories exist, the trainer also tracks their static-score difference and retires a rule after two consecutive visits below $-0.05$. These counters advance on observations of the same query, not on every global iteration. The buffer is query-specific and persisted with training state. The method does not implement transfer of a stored rule between different queries, despite requesting generally useful wording in the proposer prompt.

### A.5 Scope of Automation

The training procedure automates contract generation, static rubric synthesis and validation, trajectory sampling and comparison, process proposal, candidate scoring, parameter updates, and buffer maintenance. The online loop uses task data, source documents, pretrained models, prompts, retrieval infrastructure, and selected reward coefficients; proposal/evaluation splits keep candidate proposal and candidate evaluation trajectories disjoint.

The no-proposer-loss ablation disables the direct proposer objective while retaining online synthesis and activation, with response and proposer updates sharing parameters. The source-evaluation ablation uses proposal trajectories as the candidate-evaluation subset. Both ablations retain the frozen static anchor.

### A.6 Prompt Templates

The following templates correspond to static synthesis, process-rubric proposal, rubric judging, process-rubric validation, and response judging. The output fields and decision boundaries are fixed by the experimental protocol; task text, criteria, trajectories, and evidence are supplied for each instance.

**Answer-contract extraction.**

```
Convert the user query into an Answer Contract. Extract scope, requested count,
format, completeness, conditions, core facts, and optional facts. Use hard for
explicit non-negotiable constraints, required for central content, optional for
useful details, and conditional for facts needed only when a stated condition
applies. Set requires_retrieval=true only for claims that must be verified from
external sources. Return raw JSON only:
{"fact_applicable":true,"query_goal":"...","requirements":[
 {"description":"...","strength":"hard|required|optional|conditional",
  "requires_retrieval":true,"search_query":"what must be verified"}]}
```

**Static-rubric synthesis.**

```
Generate Task Rubrics and Fact Rubrics in separate arrays. Every rubric maps to
exactly one Answer Contract requirement. A task_rubric maps only to a
requires_retrieval=false requirement. A fact_rubric maps only to a
requires_retrieval=true requirement and evaluates an externally verifiable
claim. For each supported hard or required retrieval requirement, generate at
least one fact_rubric. Copy evidence text character-for-character from the
cited document as one contiguous exact substring. Do not use model memory or
add unmapped facts. Return at least three rubrics in total and raw JSON only:
```

```
{"task_rubrics":[{"requirement_id":"C001","strength":"hard",
 "criterion":"..."}],"fact_rubrics":[{"requirement_id":"C002",
 "strength":"required","criterion":"...","evidence":[
 {"text":"exact copied source substring","citation":"D001"}]}]}
```

**Process-rubric proposal.**

```
You are the Rubric Proposer in an open-domain reinforcement-learning pipeline.
Propose exactly one observable, transferable process or preference rubric from
the task and sampled trajectories. Compare trajectories using visible events,
final answers, and aggregate static-rubric scores. A higher score is better
according to the frozen static signal; an equal score is a tie. Use this
ordering to discover a process difference, not as a complete preference label.
Do not introduce query-specific facts, hidden reasoning, verbosity, or token
count. Do not duplicate an active rubric. Return only one JSON object:
{"text":"one observable transferable rule","direction":"positive"}
```

**Rubric judge.**

```
Evaluate exactly one visible assistant trajectory against every supplied rubric
independently. Use only the task, visible trajectory events, rubric criterion,
rubric type, strength, direction, and supplied references. For direction=positive,
score=1 only when the trajectory clearly satisfies the criterion. For
direction=negative, score=1 only when the prohibited behavior does not occur.
Use score=0 when evidence is absent, ambiguous, contradicted, or insufficient.
Return exactly one JSON object with one top-level key, scores:
{"scores":[{"rubric_id":"R001","score":1},{"rubric_id":"R002",
"score":0}]}
```

**Process-rubric validator.**

```
Decide whether a proposed rubric is safe to use as an adaptive, transferable
training signal. Check five hard gates independently: schema, observable,
boundary, principle, and nonproxy. The rule must be atomic and verifiable from
visible actions, tool events, citations, or the final answer; it must not add
query-specific facts, rewrite the frozen static anchor, or reward hidden
reasoning, verbosity, or token count. Also return stability, novelty, and
duplicate_penalty in [0,1]. Return JSON with boolean gate fields and numeric
metric fields only.
```

**Response judge.**

```
Score one visible trajectory for the supplied open-domain task. Keep these
dimensions independent: instruction_score measures fulfillment of the user's
requested format, scope, and content; attribution_score measures whether
factual claims are linked to cited frozen source IDs; cost is a normalized
penalty for unnecessary, invalid, duplicated, or excessively broad tool use.
Return exactly this JSON object, with values in [0,1] and attribution_score
allowed to be null when citations are not required:
{"instruction_score":0.9,"attribution_score":0.8,"cost":0.1}
```

# B Experimental Protocol and Supplementary Results

This appendix summarizes data construction, training configuration, evaluation metrics, experimental results, and the ablations reported in the main paper.

## B.1 Data Construction and Split Boundaries

The static corpus is built from task/fact rubric synthesis and automated filtering. The split procedure verifies unique task identifiers, globally shuffles retained records with seed 42, assigns the first 2,623 to response SFT, and assigns the remainder to RL. Table 5 reports counts from the processed corpus.

The processed static corpus stores task prompts, source identifiers, answer contracts used by the reward, typed static criteria, and evidence documents. All retained records pass the automatic quality filter. The corpus contains 7,753 fact criteria and 6,154 task criteria. Conversation-based sources use the last user message for training conversion, while the complete source conversation is retained for auditing.

The static synthesis procedure uses Tavily retrieval while excluding benchmark references and rubrics from the evidence input. Documents retain titles, URLs, text, and local identifiers. Source metadata for HealthBench and ResearchQA retain the original benchmark rubrics.

Table 5: Static-rubric corpus. Criteria are individual static task/fact items. Evidence counts refer to RL tasks with at least one stored document. SFT counts here describe the task allocation.

| Source | SFT tasks | RL tasks | SFT criteria | RL criteria | RL evidence |
|---|---|---|---|---|---|
| HealthBench | 685 | 294 | 2,455 | 1,038 | 256 |
| ResearchQA | 998 | 395 | 3,509 | 1,379 | 394 |
| LongWriter-Zero | 940 | 384 | 3,925 | 1,601 | 84 |
| Total | 2,623 | 1,073 | 9,889 | 4,018 | 734 |

The response-SFT corpus contains 2,623 trajectories, with one distinct query identifier per allocated SFT task and the same domain counts as Table 5. All trajectories end in a nonempty assistant answer; 706 include a tool response. Of these trajectories, 927 are accepted and 1,696 are rejected. The strict subset contains the 927 accepted examples. Full-data and strict-subset SFT define separate response-training inputs for comparison.

### B.2 Training and Evaluation Configuration

Table 6 records thetraining and evaluation defaults. RL comparisons use a response-SFT initialization alongside base-model and SFT-only references.

Table 6: Training/evaluation configuration values. Method reward constants are specified separately in the method appendix.

| Setting | Experimental value |
|---|---|
| Response optimization | GRPO |
| Training epochs | 5 epochs |
| Task batch / response PPO minibatch | 48 / 48 |
| Response trajectories per task | 8 |
| Proposal/evaluation trajectories | 4 / 4 before invalid-response exclusions |
| Proposals per complete task group | 6, one per unordered proposal pair |
| Actor learning rate / warmup ratio | $5 \times 10^{-7}$ / 0.03 |
| Weight decay / gradient clipping | 0.01 / 1.0 |
| Response / proposer PPO epochs | 1 / 1 |
| Proposer minibatch size | 64 |
| Prompt / generation limits | 32,768 / 32,768 tokens |
| Total model context limit | 65,536 tokens |
| Policy sampling temperature / top-$p$ | 1.0 / 1.0 |
| Numerical precision | bfloat16 |
| Training runtime | 1 node, 8 GPUs; tensor/data/Ulysses parallelism 2 / 4 / 2 |
| Rollout mode | Asynchronous multi-turn tool interaction |
| Judge model | Qwen3.6-27B |
| Search environment | Mock search with Qwen3.6-27B; maximum top-$k$ 3; maximum document length 1,500 characters |
| Evaluation judge output | temperature 1.0; max 65,536 tokens |
| Evaluation rollout limits | temperature/top-$p$ 1.0 / 1.0; max new tokens 32,768; context 65,536 |
| Training seeds per profile | 3 |

The training protocol specifies the policy, tokenizer, optimizer, and eight-GPU parallelism. Training generates synthetic search responses, while real-retrieval comparisons use a fixed evidence corpus.

The table specifies the protocol parameters needed to reproduce the experiments. Reproducing the main result requires the policy model and checkpoint, training-seed IDs and per-seed scores, dataset manifest and split version, judge prompt/schema, search evidence snapshot, and checkpoint-selection rule.

### B.3 Contribution Ablations and Training-Protocol Control

The main paper reports full ARRCE, contribution ablations, and a training-protocol control. The contribution ablations cover static/process synthesis decoupling, static-supervision signals including

the response source used for candidate evaluation, and automatic trajectory comparison; the training-protocol control covers the proposer learning objective. The no-proposer-objective variant removes only the proposer learning objective while retaining response learning and the online candidate pipeline.

**Decoupled synthesis.** The coupled control receives the same task and evidence plus current proposal trajectories, and jointly regenerates static and process criteria. It uses a fixed generation schedule and matched synthesis-token budget. The response gate structure, candidate-count budget, and final task evaluator remain fixed.

**Static supervision of process rewards.** Setting the existing alignment coefficient to zero removes positive alignment reward while retaining the static direction gate. A second intervention also disables the direction gate, leaving the same validity gates, format bonus, discrimination terms, and duplication penalty. The corresponding benchmark results are reported in the ablation table.

**Automatic pair supervision.** The score-hidden control removes both numeric scores and better/worse/tie annotations from the proposer context. The shuffled control permutes score assignments within a proposal subset before all its pairs are enumerated and recomputes labels consistently. Both preserve the same trajectory text, pair membership, current criterion text, and true static supervision during candidate evaluation. Thus the intervention changes the information used to synthesize a process rule while leaving its evaluation signal intact.

**Training-protocol controls and sensitivity analyses.** The training-protocol control is proposer-objective removal. Training-judge substitution and search-environment changes are sensitivity analyses. Source-response candidate evaluation is reported as an ablation of the static-supervision contribution. The benchmark results are reported in the main comparison, ablation, and search-environment tables.

Table 7: Failure rates for contribution ablations and the training-protocol control. Rates are computed as failed query evaluations divided by the benchmark total.

| Variant | RQA | LLMMED | WB | FB |
|---|---|---|---|---|
| Coupled synthesis | 0.05% | 0.30% | 5.70% | 0.49% |
| No alignment | 0.05% | 0.15% | 5.50% | 0.37% |
| No alignment + gate | 0.08% | 0.15% | 5.90% | 0.73% |
| No tie residual | 0.08% | 0.30% | 5.70% | 0.24% |
| Hidden score labels | 0.05% | 0.30% | 5.80% | 0.49% |
| Shuffled score labels | 0.08% | 0.90% | 6.00% | 0.73% |
| Source-response evaluation | 0.03% | 0.60% | 5.70% | 0.49% |
| No proposer objective | 0.08% | 0.30% | 5.60% | 0.12% |

The table 7 reports failure rates for the contribution ablations and the training-protocol control. Failure rates remain low on ResearchQA, LLMEval-Med, and FollowBench, while WritingBench has higher rates of approximately 5.5–6.0%. The LLMEval-Med failure rate is highest for shuffled score labels, which also produces the largest LLMEval-Med drop in Table 2; the performance difference therefore cannot be attributed to coverage alone. The no-tie-residual control has failure rates of $0.08\%$, $0.30\%$, $5.70\%$, and $0.24\%$ across the four benchmarks, close to neighboring ablations, so its score changes can be interpreted together with the mechanism intervention.

### B.4 Metric Definitions and Failure Reporting

For ResearchQA, the local evaluator assigns each criterion an ordinal coverage level $\ell \in \{1, 2, 3, 4, 5\}$ and reports the within-query mean of $(\ell - 1)/4$. The labels distinguish uninferable, inferable but unmentioned, mentioned with important details absent, mentioned with some details absent, and sufficiently detailed coverage. The dataset score is a mean of query scores, rather than a pooled mean of all criteria or the fraction of fully correct answers.

LLMEval-Med, abbreviated as LLMMED in tables, reports five medical-ability categories, MK, MLU, MR, MSE, and MTG. The judge assigns a score from 1 to 5 to each sample; a score of at least 4 is counted as usable. The overall OP is the percentage of usable samples among all samples, and the experiments report its corresponding standard deviation.

WritingBench reports the query mean of 1–10 criterion scores; the optional leaderboard scale multiplies that mean by ten. FollowBench reports hard satisfaction, soft satisfaction, and consistent satisfaction across instruction levels. Each benchmark retains its native scale.

A query is marked as a judge failure when any required criterion verdict fails evaluation. Records preserve generation IDs, answers, tool transcripts, criterion verdicts, request/parse errors, and final statuses. Reports include valid-sample denominators, coverage, missing answers, invalid rollouts, invalid benchmark records, and judge failures. All methods use the same retry rule; LLMEval-Med reports usable/total counts and failure rates alongside OP.

### B.5 Evaluation Results and Uncertainty

The detailed experimental-results table covers nine primary profiles, including the base model and the V4-Flash judge profile, across four benchmarks. Table 9 records experimental means, standard deviations, valid/total counts, and failure rates for the primary profiles.

### B.6 Search-Environment Robustness

To compare search environments, we reevaluate the same full ARRCE checkpoint with mock search and real Tavily search on all four benchmarks. ResearchQA rises from $0.791$ to $0.822$, LLMEval-Med OP from $71.36$ to $74.59$, WritingBench from $7.700$ to $7.990$, and FollowBench from $0.535$ to $0.560$; the corresponding relative differences are $3.9\%$, $4.5\%$, $3.8\%$, and $4.7\%$.

Table 8: Search-environment diagnostic for full ARRCE. The checkpoint is held fixed while evaluation uses either the mock-search tool used during training and in the main table, or real Tavily search. Relative difference is $|\mathrm{Tavily} - \mathrm{mock}|/\mathrm{mock}$. These supplementary measurements are reported separately from the main experimental comparison.

| Benchmark | Mock search | Tavily search | Relative difference |
|---|---|---|---|
| ResearchQA | 0.791 | 0.822 | 3.9% |
| LLMMED | 71.36 | 74.59 | 4.5% |
| WritingBench | 7.700 | 7.990 | 3.8% |
| FollowBench | 0.535 | 0.560 | 4.7% |

Table 9: Experimental results. The table reports the experimental mean, standard deviation, valid/total count, and failure rate for each profile and benchmark. Scores use the main-table precision: three decimals for ResearchQA, WritingBench, and FollowBench, two decimals for LLMEval-Med, and four decimals for standard deviations.

| Profile | Benchmark | Score | SD | Valid/total | Failure rate |
|---|---|---|---|---|---|
| Base model | RQA | 0.697 | 0.0030 | 3750/3750 | 0.00% |
| Base model | LLMMED | 67.02 | 0.0507 | 665/667 | 0.30% |
| Base model | WB | 7.252 | 0.0410 | 965/1000 | 3.50% |
| Base model | FB | 0.443 | 0.0166 | 816 | 0.49% |
| Response SFT | RQA | 0.734 | 0.0030 | 3750/3750 | 0.00% |
| Response SFT | LLMMED | 68.37 | 0.0512 | 665/667 | 0.30% |
| Response SFT | WB | 7.452 | 0.0438 | 958/1000 | 4.20% |
| Response SFT | FB | 0.495 | 0.0169 | 817/820 | 0.37% |
| Static additive | RQA | 0.778 | 0.0028 | 3750/3750 | 0.00% |
| Static additive | LLMMED | 69.42 | 0.0512 | 662/667 | 0.75% |

| Profile | Benchmark | Score | SD | Valid/total | Failure rate |
|---|---|---|---|---|---|
| Static additive | WB | 7.639 | 0.0445 | 944/1000 | 5.60% |
| Static additive | FB | 0.523 | 0.0169 | 817/820 | 0.37% |
| Static gated | RQA | 0.764 | 0.0030 | 3750/3750 | 0.00% |
| Static gated | LLMMED | 71.21 | 0.0500 | 666/667 | 0.15% |
| Static gated | WB | 7.646 | 0.0425 | 943/1000 | 5.70% |
| Static gated | FB | 0.528 | 0.0169 | 818/820 | 0.24% |
| Dynamic additive | RQA | 0.779 | 0.0028 | 3749/3750 | 0.03% |
| Dynamic additive | LLMMED | 70.76 | 0.0512 | 663/667 | 0.60% |
| Dynamic additive | WB | 7.682 | 0.0431 | 944/1000 | 5.60% |
| Dynamic additive | FB | 0.526 | 0.0170 | 814/820 | 0.73% |
| EvoRubric-adapted | RQA | 0.776 | 0.0029 | 3749/3750 | 0.03% |
| EvoRubric-adapted | LLMMED | 69.27 | 0.0506 | 666/667 | 0.10% |
| EvoRubric-adapted | WB | 7.656 | 0.0436 | 942/1000 | 5.80% |
| EvoRubric-adapted | FB | 0.523 | 0.0169 | 817/820 | 0.37% |
| DR Tulu/RLER | RQA | 0.780 | 0.0028 | 3749/3750 | 0.03% |
| DR Tulu/RLER | LLMMED | 70.91 | 0.0506 | 661/667 | 0.90% |
| DR Tulu/RLER | WB | 7.632 | 0.0434 | 965/1000 | 3.50% |
| DR Tulu/RLER | FB | 0.518 | 0.0168 | 816/820 | 0.49% |
| Full ARRCE | RQA | 0.791 | 0.0027 | 3743/3750 | 0.19% |
| Full ARRCE | LLMMED | 71.36 | 0.0507 | 667/667 | 0.00% |
| Full ARRCE | WB | 7.700 | 0.0436 | 953/1000 | 4.70% |
| Full ARRCE | FB | 0.535 | 0.0169 | 817/820 | 0.37% |
| ARRCE (V4-Flash judge) | RQA | 0.796 | 0.0026 | 3750/3750 | 0.00% |
| ARRCE (V4-Flash judge) | LLMMED | 72.14 | 0.0502 | 667/667 | 0.00% |
| ARRCE (V4-Flash judge) | WB | 7.717 | 0.0420 | 958/1000 | 4.20% |
| ARRCE (V4-Flash judge) | FB | 0.537 | 0.0165 | 818/820 | 0.24% |

The full ARRCE profile exceeds Response SFT on ResearchQA (0.791 versus 0.734), LLMEval-Med OP (71.36 versus 68.37), WritingBench (7.700 versus 7.452), and FollowBench (0.535 versus 0.495). The evidence therefore supports positive changes across all four benchmarks. Among the comparison profiles, static gated has the highest LLMEval-Med OP (71.21), while the V4-Flash judge profile has the highest WritingBench mean (7.717), further indicating that the observed ranking depends on the benchmark.

### B.7 Case Study: From Static Anchors to Response Rewards

This case concerns an open-domain question asking what hospital-based case-control studies in China show about the association between alcohol use and sexually transmitted infections (STIs). The task requires evidence across multiple regions, a quantitative estimate of risk, discussion of behavioral and demographic confounders, and a calibrated assessment of evidence strength. The search environment is synthetic, so the case evaluates rubric coverage and evidence organization rather than independently verifying the cited literature.

**Static rubric.** The four criteria below define the static evaluation anchor used for both responses. A score of five is normalized to one and a score of one to zero in the coverage score.

**Frozen static criteria and verdicts for the case study.**

| ID | Criterion | ARRCE | SFT |
|---|---|---|---|
| S1 | Provide evidence of a significant positive association between alcohol use and STIs from multiple studies conducted in different regions of China. | 5 | 5 |
| S2 | Mention odds ratios or another measure quantifying the increased STI risk linked to alcohol use. | 5 | 1 |
| S3 | Address potential confounding factors considered in the studies, including sexual behavior and demographic variables. | 5 | 1 |
| S4 | Analyze the strength and consistency of the evidence linking alcohol use to STIs. | 5 | 5 |
| | Normalized coverage score | 1.00 | 0.50 |

**Process rubric.** For this task, a useful process criterion is: *Given search results for Chinese hospital-based case-control studies, organize the final synthesis around study population/region, at least one quantitative association, behavioral or demographic confounders, and study-design limitations; preserve the distinction between association and causation, and omit mechanisms that are not supported by the retrieved evidence.* We audit this single atomic criterion with four observable checks: (D1) issue targeted searches that move from the broad association to study populations, regions, effect sizes, and confounders; (D2) retain study-level details rather than only a generic consensus statement; (D3) connect the quantitative result and behavioral factors to the claim while distinguishing mediation from adjustment; and (D4) qualify hospital selection, self-report, urban concentration, and the absence of longitudinal evidence.

**Error trajectory: response SFT.** The pre-training response receives $0.50$. It correctly states that multiple Chinese hospital-based studies report a positive association and discusses HIV, syphilis, and gonorrhea, satisfying S1 and S4. Its retrieval remains broad and includes an off-target infection query, so it does not recover a study-level quantitative estimate. The answer contains no odds ratio, confidence interval, or other effect-size estimate (S2), and it does not identify concrete adjusted variables or explain how sexual behavior relates to confounding or mediation (S3). It also introduces a possible mucosal-barrier mechanism without tying that mechanism to a retrieved study. The resulting path moves from a general association claim directly to a generic synthesis and fails to make the missing rubric dimensions auditable.

**Rubric-aligned trajectory: full ARRCE.** The post-training ARRCE response receives $1.00$. Its retrieval progresses from the overall association to named STI types, MSM populations, Guangzhou/Shenzhen and Beijing settings, meta-analytic evidence, and quantitative risk. The final answer reports an adjusted odds ratio of approximately $1.8$ for frequent alcohol users in a Beijing study, identifies inconsistent condom use, number of partners, and impulsive encounters as behavioral factors, compares urban and vulnerable populations, and states that the evidence is concentrated in urban settings with limited longitudinal support. These details satisfy S2 and S3 while making the strength and consistency claim in S4 inspectable. The answer preserves the synthetic-search caveat and distinguishes association from causation.

**Why the aligned trajectory scores higher.** The score increase comes from recovering evidence that the error trajectory leaves unqualified. The aligned response searches for study populations and regions before settling on a single generalization, seeks a quantitative effect rather than relying on the phrase "positive association," and checks which behavioral and demographic variables are considered in adjustment. It then carries those distinctions into the answer: "One study among young adults in Beijing found that frequent alcohol users had an adjusted odds ratio of approximately 1.8 for STI acquisition, controlling for age, education, and socioeconomic status." The response further identifies "inconsistent or non-protective condom use," "increased number of sexual partners," and "impulsive or unpredictable sexual encounters" as behavioral factors, while stating the urban

concentration of the evidence and the limits of the study designs. These additions improve score coverage because they provide auditable support for S2–S4; the gain reflects evidence selection and qualification rather than answer length.

**Process-rubric interpretation.** The process rubric maps this improvement to four observable checks: targeted retrieval over populations, regions, effect sizes, and confounders; retention of study-level details; connection of quantitative and behavioral evidence to the claim while distinguishing adjustment from mediation; and explicit limitations without unsupported causal or mechanistic escalation. The error trajectory partially satisfies the first two checks but fails the quantitative and confounder checks. The aligned trajectory satisfies all four under the available evidence. This analysis explains why the static anchor and process rubric are complementary: the static rubric defines what the answer must contain, while the process rubric identifies the search and synthesis behaviors that make those dimensions recoverable.

**Trajectory-level audit of the process rubric.**

| Check | Observable requirement | Response SFT | Full ARRCE |
|---|---|---|---|
| D1 | Targeted search progression covering populations/regions, effect size, and confounders | Partial | Pass |
| D2 | Study-level details rather than a generic positive-association summary | Partial | Pass |
| D3 | Quantitative result plus behavioral/demographic explanation with association/mediation distinction | Fail | Pass |
| D4 | Explicit limitations and no unsupported causal/mechanistic escalation | Partial | Pass with caveat |