SERPO: SELF-EVOLVING RUBRIC POLICY OPTIMIZATION FOR OPEN-ENDED TEST-TIME REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reinforcement learning (TTRL) enables language models to self-evolve at inference time without labeled feedback. Existing methods rely on answer voting and therefore do not extend naturally to open-ended generation, where valid responses cannot be mapped to a shared canonical answer. Without external reward models or stronger judges, adaptation must instead construct reliable rewards from the model's own outputs. We introduce SERPO (Self-Evolving Rubric Policy Optimization), which replaces answer voting with a closed loop that co-evolves response evidence, query-specific rubrics, and policy parameters. Good–Normal–Bad (G-N-B) response evolution organizes maximally separated rollouts into ordered archives; rubric evolution retains criteria that distinguish archived responses; probabilistic criterion scoring converts verdict-token likelihoods into reward signals; and policy evolution optimizes the actor with the resulting signals. New actor rollouts then refresh both the archives and rubrics, closing the three-way evolution loop. Across two model configurations, two in-domain benchmarks, and four OOD benchmarks, SERPO improves HealthBench and ResearchQA by up to **20.63** and **20.31** points over the corresponding base models, raises the six-benchmark macro-average by up to **8.06** points, and supports OOD transfer and continued cross-benchmark evolution.

1 INTRODUCTION

Test-time reinforcement learning (TTRL) enables a deployed language model to self-evolve on the test distribution through rewards constructed at inference time. We study open-ended generation under a transductive setting: the model may use the original test prompts and its own sampled responses, but receives no reference answers, human feedback, external reward models, or stronger judges. It repeatedly samples responses, constructs rewards, and updates its parameters on the same prompt set. TTRL therefore depends on a reward signal that remains informative as the policy evolves.

Existing TTRL methods construct this reward by sampling several solutions, selecting a pseudo-label through majority voting or self-consistency-style aggregation, and rewarding responses that agree with it (Wang et al., 2023; Zuo et al., 2025). Subsequent work improves pseudo-label reliability through self-reflection, entropy-based confidence, or explicit mitigation of majority-vote failures (Wu et al., 2026; Agarwal et al., 2025; Lin et al., 2026). These methods still assume that responses can be reduced to comparable answers. The assumption holds for multiple-choice, symbolic, and canonically normalized tasks, but it fails for open-ended generation. Two medical responses may recommend the same action yet differ in contraindications, uncertainty, and escalation guidance; equally safe answers may also use different wording or levels of detail. Answer voting can therefore reward shared omissions and reject valid alternatives. Current TTRL objectives remain centered on tasks with extractable answers and do not provide a reliable voting target for open-ended responses.

Rubric rewards offer a natural alternative for open-ended tasks because they score separate requirements such as factuality, completeness, instruction compliance, and safety (Liu et al., 2023; Kim et al., 2024; Ye et al., 2024; Liu et al., 2024; Gunjal et al., 2025). Recent work has established

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.

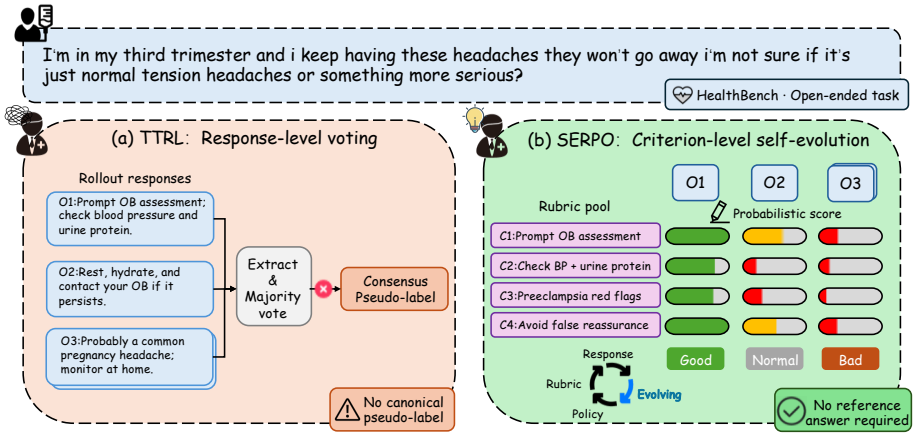

Figure 1: **Criterion-level evidence for open-ended TTRL.** Claim consensus can retain frequent but incomplete advice, whereas SERPO forms G-N-B evidence using evolving query-specific criteria without reference answers.

rubric–policy co-evolution as an approach to open-ended learning, using adversarial updates, temporal contrasts, meta-verification, or response-set conditioning (Ding et al., 2026; Li et al., 2026a; Guan et al., 2026; Wang et al., 2026). We study adaptation on a fixed set of unlabeled test prompts, where the original prompts and self-generated responses are the only task-specific information available. SERPO keeps the judge and rubric-generator parameters frozen, while adapting their evaluation context through rolling query-local response archives and an evolving criterion pool. Only the actor receives parameter updates.

We address this problem with SERPO, which co-evolves query-local G-N-B response archives, query-specific rubrics, and shared actor parameters. Figure 1 contrasts this criterion-verification loop with answer voting. For each prompt, SERPO stores an ordered G-N-B response archive and derives atomic criteria from contrasts among these responses. SERPO aggregates post-reasoning Pass/Fail probabilities into continuous rewards under the evolving rubric; LLM-as-a-Verifier studies a related fixed-criterion interface (Kwok et al., 2026). SERPO ranks criteria by their score dispersion across responses, consolidates duplicates, and removes persistently weak criteria. The resulting rubric supplies group-relative rewards for actor updates. New actor responses then refresh the archives and rubric, closing the response–rubric–policy loop.

Across two model configurations, two in-domain benchmarks, and four OOD benchmarks, SERPO improves HealthBench and ResearchQA by up to 20.63 and 20.31 points and raises the six-benchmark macro-average by up to 8.06 points. The evolved policies also transfer to unseen benchmarks and continue improving under sequential cross-benchmark evolution.

Contributions.

- We formulate TTRL for open-ended generation under a fixed, label-free information budget and introduce judge-based response voting and a stronger claim-consensus objective.
- We introduce SERPO, which uses rolling query-local response archives to adapt evaluation rubrics as the policy evolves, converting criterion-level feedback into rewards for open-ended TTRL.
- We introduce a probabilistic reward interface for evolving rubrics, aggregating criterion-satisfaction probabilities derived from post-reasoning Pass/Fail distributions.
- Experiments across two model configurations and six benchmarks establish strong in-domain gains, OOD transfer, and continued cross-benchmark evolution.

2 Related Work

Test-Time Reinforcement Learning. TTRL turns unlabeled test prompts into online reinforcement-learning data by sampling solutions and using the majority answer as a pseudo-label, following self-consistency aggregation (Wang et al., 2023; Zuo et al., 2025). Self-reflection, entropy minimization, and extinction-window mitigation improve pseudo-label selection or confidence (Wu et al., 2026; Agarwal et al., 2025; Lin et al., 2026), but still require answer extraction and task-specific equivalence. SERPO instead constructs rewards through query-specific criterion verification, which applies when open-ended responses have no canonical answer.

Rubric Rewards and Policy–Rubric Co-Evolution. Rubrics turn open-ended criteria into rewards for self-distillation or policy training (Gunjal et al., 2025; Li et al., 2026b; Whitehouse et al., 2025; Chen et al., 2025; Xie et al., 2026; Rezaei et al., 2026; Fang et al., 2026). RLCER evolves CoT rubrics around outcome-centric RLVR; EvoRubrics adversarially trains a rubric generator; EvoLM alternates rubric and policy training using temporal checkpoint contrasts; EvoRubric shares one policy across roles with meta-verification and rubric memory; and DynamicRubric co-trains a response-set-conditioned evaluator (Sheng et al., 2026; Ding et al., 2026; Li et al., 2026a; Guan et al., 2026; Wang et al., 2026). Co-evolution can also proceed without external supervision, as in EvoLM and the self-supervised variant of EvoRubrics (Li et al., 2026a; Ding et al., 2026). SERPO adapts the response evidence and rubric text rather than the evaluator parameters. Its rolling query-local archives connect successive encounters with the same prompt, and criterion-level verdict probabilities provide continuous rewards for actor optimization. This reward interface builds on token-distribution-based evaluation and verification (Liu et al., 2023; Kwok et al., 2026).

Contextual Self-Evolution. Other methods improve a frozen model through test-time sampling or search (Wang et al., 2023; Yao et al., 2023; Bay & Yearick, 2026), iterative self-feedback (Madaan et al., 2023; Shinn et al., 2023), or accumulated external state. ACE stores reusable context, RGSD supplies teacher rubrics, ARBOR maintains a rubric buffer, TTCS creates curriculum variants, and recursive agents select improved configurations across rounds (Zhang et al., 2025b; Rezaei et al., 2026; Liu et al., 2026; Yang et al., 2026; Nguyen et al., 2026). Their progress resides mainly in context, memory, or external rubric pools and may depend on auxiliary task construction. SERPO retains evolving response and rubric states but converts their feedback into persistent actor updates on the fixed test set.

3 Preliminaries

3.1 Test-Time Reinforcement Learning

We consider fixed-set transductive TTRL. Let $\mathcal{D}_{\text{adapt}} = \{x_n\}_{n=1}^{N}$ denote the unlabeled test prompts observed before adaptation. During adaptation, a method may use only these prompts, its sampled responses, and derived statistics; it receives no references, external task rewards, human or external evaluator feedback, auxiliary corpus, or grader-guided prompt selection. Auxiliary models may be queried only in a frozen, evaluation-blind manner: they are not updated during adaptation and cannot access evaluation resources. After adaptation, a hidden grader evaluates fresh responses on $\mathcal{D}_{\text{adapt}}$ for reporting only; we refer to this transductive evaluation on the adaptation prompts as in-domain (ID) evaluation. A disjoint $\mathcal{D}_{\text{eval}}$ measures held-out transfer.

At adaptation step t, the policy processes a minibatch $\mathcal{X}_t \subset \mathcal{D}_{\text{adapt}}$ and samples G responses $o_{t,i} \sim \pi_{\theta_t}(\cdot \mid x)$ for each prompt. The pseudo-reward is fixed within each actor update and may evolve between updates. Appendix G gives the formal objective and answer-level voting reward; SERPO replaces answer agreement with criterion verification.

3.2 Rubric Criteria for Open-Ended Responses

A rubric decomposes open-ended quality into atomic natural-language criteria. We write $\mathcal{R} = \{(c_m, \rho_m, w_m)\}_{m=1}^{M}$, where c_m is a criterion, $\rho_m \in \{+1, -1\}$ is its polarity, and $w_m \geq 0$ is its weight. A judge provides a criterion-satisfaction probability $q_J(x, o, c_m) \in [0, 1]$ for response o and criterion c_m; positive criteria reward desirable properties, while negative criteria mark failures.

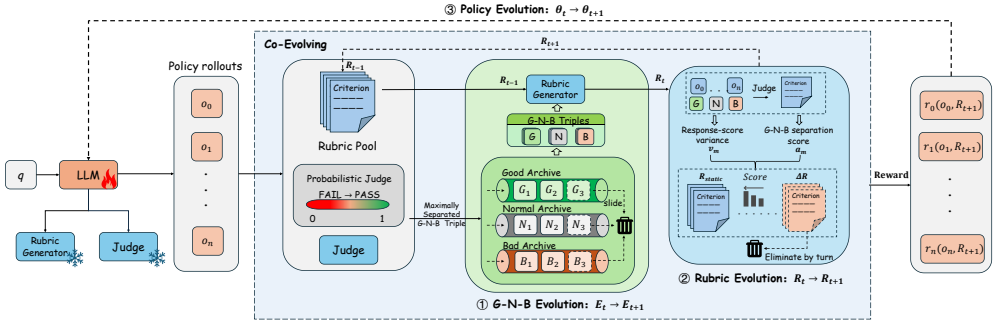

Figure 2: **Overview of SERPO.** Here q is the input query, while o_i and r_i are the i-th policy rollout and its scalar reward. $\mathcal{E}_t$, $\mathcal{R}_t$, and θ_t denote the query-local G-N-B archive state, query-specific rubric state, and shared actor parameters at encounter t, respectively. d_m denotes criterion utility; Δ_R denotes the low-utility elimination region. Snowflakes mark fixed initial-weight copies used as the rubric generator and judge.

Section 4 shows how SERPO orients and calibrates these probabilities, then aggregates the resulting scores into the scalar reward used by GRPO.

3.3 Group-Relative Policy Optimization

SERPO uses GRPO (Shao et al., 2024), a PPO-style group-relative optimizer (Schulman et al., 2017), as the actor optimizer; only the reward source changes. For a rollout group on prompt x, GRPO converts scalar rewards $\{r_i\}_{i=1}^{G}$ into critic-free advantages, $\hat{A}_i = (r_i - \bar{r})/(\sigma_r + \varepsilon_A)$, and updates the actor with the standard clipped objective and KL penalty. For answer-verifiable tasks, rewards can be defined by agreement with a selected pseudo-label (Appendix G); SERPO instead supplies r_i from the evolving rubric reward in Section 4.4.

4 Method: SERPO

SERPO alternates between updating response archives, revising evaluation criteria, and updating the actor. For each prompt, it maintains a rubric and three response archives, called Good, Normal, and Bad. These labels indicate relative rubric scores at the time of selection, not externally verified correctness. All prompts share one actor. Frozen copies of the initial model generate criteria and judge responses; only the actor receives parameter updates.

At encounter t, the archives for prompt x are $\mathcal{E}_t(x) = (\mathcal{A}_t^G(x), \mathcal{A}_t^N(x), \mathcal{A}_t^B(x))$, and its rubric is $\mathcal{R}_t(x) = \{(c_m, \rho_m, w_{m,t})\}_{m \in \mathcal{M}_t(x)}$. Here $\mathcal{M}_t(x)$ indexes the active criteria; we omit x when unambiguous. SERPO first samples responses and computes scores for archive ordering. It then updates the archives, proposes and merges criteria when scheduled, refreshes criterion utilities and the active rubric, and computes rewards for the actor update. Figure 2 summarizes these steps; the supplement gives the complete algorithm.

4.1 Criterion Scoring and Reward Interface

Under the fixed-set budget, each judge call must preserve reward resolution. Hard verdicts create ties and shrink GRPO advantages, so SERPO converts post-reasoning Boolean verdict-token probabilities into criterion-satisfaction probabilities. If $\ell_{\mathrm{T}}^{(m)}$ and $\ell_{\mathrm{F}}^{(m)}$ are the log probabilities assigned to the positive and negative verdicts, define

$$q_J(x, o, c_m) := \frac{\exp \ell_{\mathrm{T}}^{(m)}}{\exp \ell_{\mathrm{T}}^{(m)} + \exp \ell_{\mathrm{F}}^{(m)}}. \tag{1}$$

Here $q_J(x, o, c_m)$ is the criterion-satisfaction probability for response o on criterion c_m. This $[0, 1]$ probability distinguishes confident from borderline satisfaction. SERPO converts positive and neg-

ative criteria into an oriented criterion score,

$$z_{m,t}(o) = \begin{cases} q_J(x, o, c_m), & \rho_m = +1, \\ 1 - q_J(x, o, c_m), & \rho_m = -1. \end{cases} \tag{2}$$

Here $z_{m,t}(o)$ is the polarity-oriented criterion score at encounter t. Larger $z_{m,t}(o)$ always indicates better behavior. SERPO caches each judged tuple and reuses it for archive ordering, utility estimation, calibration, and reward construction; GRPO receives only the final scalar reward computed after the rubric update in Section 4.4.

4.2 G-N-B Response Evolution

For a new prompt, the fixed rubric generator first creates a nonempty initial set of query-specific criteria, from which SERPO initializes an equal-weight rubric. At each later encounter, the actor samples a rollout group and the judge assigns provisional archive scores for ordering,

$$r_{t,i}^{\mathrm{arc}} = \frac{\sum_{m \in \mathcal{M}_t(x)} w_{m,t} z_{m,t}(o_{t,i})}{\sum_{m \in \mathcal{M}_t(x)} w_{m,t}}. \tag{3}$$

This score is used to order responses for the archive update. For the current rollout group, write $s_i = r_{t,i}^{\mathrm{arc}}$. SERPO jointly selects high-, intermediate-, and low-scoring responses to maximize their separation. Let $\mathcal{I}_t = \{(i, j, k) \in [G]^3 : i, j, k \text{ distinct}, \ s_i \geq s_j \geq s_k\}$, where $[G] = \{1, \ldots, G\}$. SERPO selects

$$\begin{aligned} D_t(i, j, k) &:= (s_i - s_j)(s_j - s_k)(s_i - s_k), \\ (i_t^G, i_t^N, i_t^B) &= \arg \max_{(i,j,k) \in \mathcal{I}_t} D_t(i, j, k). \end{aligned} \tag{4}$$

Here $D_t(i, j, k)$ is the separation score of a candidate ordered triple, and (i_t^G, i_t^N, i_t^B) are the selected Good, Normal, and Bad rollout indices. If the maximum D_t is zero, the archive is left unchanged. Otherwise, the selected responses are appended to bounded FIFO *Good*, *Normal*, and *Bad* archives, producing $\mathcal{E}_{t+1}(x)$. This keeps query-local evidence both ordered and separated for rubric refresh.

4.3 Rubric Evolution

When a prompt reaches its refresh interval, the rubric generator proposes atomic criteria from the prompt, current rubric $\mathcal{R}_t(x)$, and archives $\mathcal{E}_t(x)$ from preceding encounters. We then evaluate the candidate pool on the updated archive, incorporating responses from the current encounter. SERPO merges candidates with the existing pool: duplicate requirements map to one canonical statement and stable identifier, related but distinct rules remain separate, retained criteria keep their counters, and new criteria start from zero.

Let $\widetilde{\mathcal{M}}_{t+1}(x)$ denote the resulting criterion pool after carrying over existing criteria and applying any scheduled merge. Criterion utility is the normalized variance of oriented scores over archived responses; it does not depend on their G-N-B labels. After each encounter and any scheduled merge, SERPO scores only unseen archive–criterion pairs and converts them to Eq. (2). Let $\mathcal{H}_{t+1}(x) = \mathcal{A}_{t+1}^G(x) \cup \mathcal{A}_{t+1}^N(x) \cup \mathcal{A}_{t+1}^B(x)$ denote the updated archive entries for the current prompt. For each $m \in \widetilde{\mathcal{M}}_{t+1}(x)$, SERPO assigns a discrimination utility

$$d_{m,t+1} = 4 \operatorname{Var}_{o \in \mathcal{H}_{t+1}(x)} \left[z_{m,t+1}(o) \right]. \tag{5}$$

The factor 4 normalizes the maximum variance of a $[0, 1]$ score to one. Constant scores yield zero utility, while greater score dispersion yields higher utility. Criterion polarity remains defined by Eq. (2).

SERPO removes criteria that consistently rank near the bottom of the pool. If $|\widetilde{\mathcal{M}}_{t+1}(x)| \geq 4$, the bottom $K_{t+1} = \max\{1, \lfloor \zeta |\widetilde{\mathcal{M}}_{t+1}(x)| \rfloor\}$ criteria enter the elimination region Δ_R in Figure 2, where ζ is the elimination fraction. If the pool has fewer than four criteria, the elimination region is empty. A criterion's counter increments only while it remains in this region, resets once it leaves, and triggers deletion after three consecutive at-risk rounds. If the surviving pool exceeds its capacity, SERPO retains the highest-utility criteria up to the cap. Survivors form $\mathcal{M}_{t+1}(x)$.

4.4 POLICY EVOLUTION

SERPO computes the GRPO reward after updating the rubric. It weights the surviving criteria by utility and calibrates their scores using the Good and Bad archives. All criterion quantities below refer to the updated rubric $\mathcal{R}_{t+1}(x)$. We suppress prompt and time indices, writing $o_i = o_{t,i}$, $r_i = r_{t,i}$, and $\mathcal{M} = \mathcal{M}_{t+1}(x)$; w_m denotes the final-reward weight.

$$\tilde{z}_m(o) = \begin{cases} \mathrm{clip}_{[0,1]}\left(\dfrac{z_m(o) - \mu_m^B}{\mu_m^G - \mu_m^B}\right), & \mu_m^G - \mu_m^B \geq \delta, \\ z_m(o), & \text{otherwise}, \end{cases} \tag{6}$$

$$r_i = \frac{\sum_{m \in \mathcal{M}} w_m \tilde{z}_m(o_i)}{\sum_{m \in \mathcal{M}} w_m}, \qquad w_m = \max(\epsilon_u, d_m).$$

Here μ_m^G and μ_m^B are the mean oriented scores in the Good and Bad archives. Calibration is applied when both means are available and their signed difference is at least δ; otherwise SERPO uses $z_m(o)$. GRPO normalizes the resulting rewards within each rollout group. The updated actor supplies responses for subsequent archive and rubric updates; the rubric generator and judge remain fixed.

5 EXPERIMENTS

Our experiments answer five questions. **RQ1**: how much does SERPO improve in-domain open-ended performance across benchmarks and model configurations? **RQ2**: do the gains from evolution transfer to held-out OOD benchmarks? **RQ3**: how do rubric evolution and policy evolution complement each other? **RQ4**: which components and evaluator design choices drive SERPO's gains? **RQ5**: can SERPO sustain improvement over longer horizons and after switching to a new benchmark?

5.1 EXPERIMENTAL SETUP

Benchmarks and evaluation. We evaluate two Qwen-family models, `Qwen3-4B-Instruct-2507` (Qwen Team, 2025) and `Qwen3.5-9B` (Qwen Team, 2026a), in non-thinking mode. We use official evaluation or validation splits, including HealthBench-500 (Arora et al., 2025) (denoted HealthBench), unless otherwise noted. Actor-updating methods evolve independently on HealthBench or ResearchQA (Li S. Yifei et al., 2026). ID evaluation uses fresh responses to adaptation prompts. For actor-updating methods, OOD evaluation applies the HealthBench-evolved policy to MedQA (Jin et al., 2021) and LLMEval-Med (Zhang et al., 2025a), and the ResearchQA-evolved policy to GPQA-Diamond (Rein et al., 2023) and RaR-Science (Gunjal et al., 2025) without further adaptation. Frozen-actor baselines initialize rubrics per OOD prompt; the evolution variant updates them locally (Appendix C.2). MedQA and GPQA-Diamond are multiple-choice benchmarks scored by direct answer extraction; GPT-5.1 (OpenAI, 2025) evaluates HealthBench, ResearchQA, LLMEval-Med, and RaR-Science using their official rubrics. Except for this privileged reference, no external judge or official rubric is available during adaptation. In Table 1, subscripts denote changes from Base, light-blue rows extend the method above, bold marks the best label-free result, and Avg. is the macro-average over all six benchmark columns.

Evolution protocol. On the ID benchmarks, adapted methods in Table 1 follow a common 30-epoch schedule, and reported scores are averaged over three evaluation runs. The sequential experiment evolves the same actor for 30 epochs on HealthBench and then 30 epochs on ResearchQA, evaluating both benchmarks during the second stage. Figure 3 alone extends SERPO to 45 epochs to examine long-horizon evolution. The supplement reports full decoding, optimization, compute, and reproducibility details, plus mean$\pm$standard deviation for all main-table entries.

Baselines. *Base* performs no adaptation. To extend voting-based TTRL to open-ended generation, both voting baselines use a frozen copy of the initial actor as a self-judge and rebuild their pseudo-labels from the current rollout group at every adaptation step. We strengthen both baselines with $G = 16$ rollouts per prompt, twice the group size used by SERPO. *Response vote* compresses each

Table 1: Main ID/OOD results. Frozen-actor OOD entries use query-local TTS. *Claim-consensus TTRL; †external judge and official rubric (not label-free).

Method	Evolve on HealthBench			Evolve on ResearchQA			Avg.
	HealthBench (ID)	MedQA (OOD)	LLMEval-Med (OOD)	ResearchQA (ID)	GPQA-Diamond (OOD)	RaR-Science (OOD)	
Qwen3-4B							
Base	32.30	53.81	59.82	57.29	53.03	67.29	53.92
TTRL (response vote)	$34.27_{+1.97}$	$53.74_{-0.07}$	$59.92_{+0.10}$	$57.91_{+0.62}$	$54.54_{+1.51}$	$66.46_{-0.83}$	$54.47_{+0.55}$
+ claim consensus*	$43.71_{+11.41}$	$53.65_{-0.16}$	$55.92_{-3.90}$	$69.39_{+12.10}$	$57.07_{+4.04}$	$\mathbf{68.15}_{+0.86}$	$57.98_{+4.06}$
RGSD (TTS; static rubric)	$27.13_{-5.17}$	$50.96_{-2.85}$	$54.32_{-5.50}$	$58.20_{+0.91}$	$52.18_{-0.85}$	$65.13_{-2.16}$	$51.32_{-2.60}$
+ rubric self-evolution	$32.06_{-0.24}$	$52.36_{-1.45}$	$60.36_{+0.54}$	$65.51_{+8.22}$	$54.92_{+1.89}$	$67.36_{+0.07}$	$55.43_{+1.51}$
SERPO (ours)	$\mathbf{49.83}_{+17.53}$	$\mathbf{54.73}_{+0.92}$	$\mathbf{62.50}_{+2.68}$	$\mathbf{77.60}_{+20.31}$	$\mathbf{59.09}_{+6.06}$	$68.12_{+0.83}$	$\mathbf{61.98}_{+8.06}$
Ext. judge + official rubric†	$56.14_{+23.84}$	$53.62_{-0.19}$	$60.18_{+0.36}$	$82.73_{+25.44}$	$57.32_{+4.29}$	$67.32_{+0.03}$	$62.89_{+8.96}$
Qwen3.5-9B							
Base	44.68	69.84	69.27	68.62	70.20	77.87	66.75
TTRL (response vote)	$53.75_{+9.07}$	$68.03_{-1.81}$	$69.75_{+0.48}$	$67.43_{-1.19}$	$68.69_{-1.51}$	$78.13_{+0.26}$	$67.63_{+0.88}$
+ claim consensus*	$60.88_{+16.20}$	$69.60_{-0.24}$	$70.32_{+1.05}$	$79.45_{+10.83}$	$68.54_{-1.66}$	$77.47_{-0.40}$	$71.04_{+4.30}$
RGSD (TTS; static rubric)	$39.42_{-5.26}$	$64.60_{-5.24}$	$65.73_{-3.54}$	$63.77_{-4.85}$	$69.61_{-0.59}$	$71.22_{-6.65}$	$62.39_{-4.36}$
+ rubric self-evolution	$44.18_{-0.50}$	$70.24_{+0.40}$	$69.24_{-0.03}$	$70.77_{+2.15}$	$71.03_{+0.83}$	$76.94_{-0.93}$	$67.07_{+0.32}$
SERPO (ours)	$\mathbf{65.31}_{+20.63}$	$\mathbf{71.32}_{+1.48}$	$\mathbf{74.18}_{+4.91}$	$\mathbf{83.18}_{+14.56}$	$\mathbf{72.22}_{+2.02}$	$\mathbf{79.93}_{+2.06}$	$\mathbf{74.36}_{+7.61}$
Ext. judge + official rubric†	$68.41_{+23.73}$	$70.38_{+0.54}$	$72.86_{+3.59}$	$90.21_{+21.59}$	$69.65_{-0.55}$	$78.43_{+0.56}$	$74.99_{+8.24}$

response into its principal recommendation and rewards membership in the largest word-set Jaccard cluster. Our stronger *claim consensus* baseline extracts atomic claims, constructs a response–claim support matrix A, and rewards coverage of claims supported by at least a fraction κ of the group:

$$p_j = \frac{1}{G}\sum_{\ell=1}^{G} A_{\ell j}, \qquad \mathcal{T}_\kappa = \{j : p_j \geq \kappa\}, \qquad r_i^{\text{claim}} = \frac{\sum_{j\in\mathcal{T}_\kappa} A_{ij}}{|\mathcal{T}_\kappa|}, \qquad \kappa = 0.5. \tag{7}$$

This claim-level objective follows the atomic decomposition used in long-form factual evaluation (Min et al., 2023) and preserves partial agreement without requiring complete-response equivalence. The supplementary material details the voting baselines, including reward definitions, decoding settings, thresholds, canonicalization, and fallback rules.

We instantiate *RGSD (TTS)* (Rezaei et al., 2026), originally a rubric-guided self-distillation method, as rubric-guided test-time scaling, where the rubric acts as privileged guidance and the actor remains frozen. We then add our rubric-evolution mechanism while still freezing the actor, isolating gains without parameter evolution. SERPO additionally updates the actor with GRPO. The privileged reference uses `Qwen/Qwen3.6-27B` (Qwen Team, 2026b) as a strong fixed external judge with the official rubric during adaptation.

5.2 Main Results

Open-Ended Test-Time Evolution (RQ1). Table 1 shows that claim consensus improves response voting by 13–28%, as claim decomposition recovers semantic agreement missed by whole-

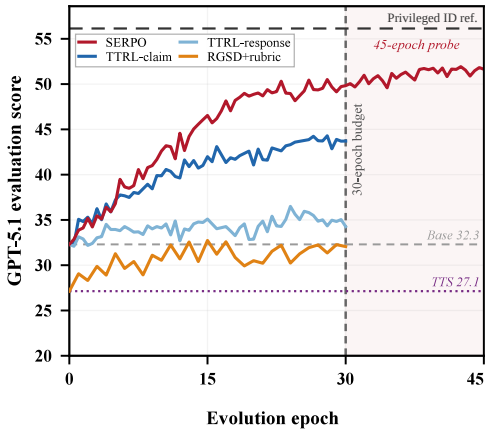

Figure 3: Qwen3-4B evolution on HealthBench (HB). Observed trajectories are shown through epoch 30 for the baselines and epoch 45 for SERPO.

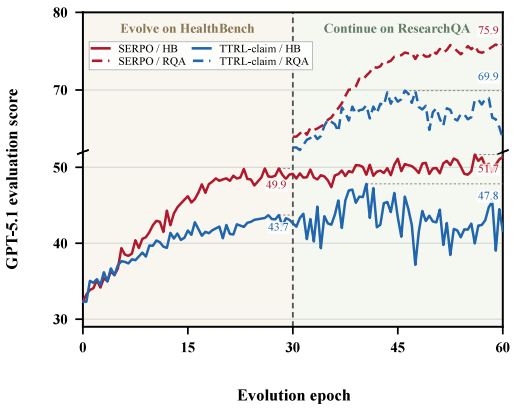

Figure 4: Qwen3-4B evolution from HealthBench (HB) to ResearchQA (RQA); labels mark stage maxima.

response voting. This finer pseudo-label is still frequency-based: highly supported claims can remain incomplete when the rollout group shares the same omission. SERPO is the strongest label-free method in every in-domain setting, improving Base by 21–54% and claim consensus by 5–14%. The remaining gap supports the benefit of evolving quality criteria beyond agreement-based rewards. Without external supervision, Qwen3-4B recovers roughly three quarters of the in-domain improvement achieved by the external-judge and official-rubric reference.

OOD Transfer (RQ2). All eight SERPO OOD results improve over Base, by up to 11.4%. Across both model configurations, HealthBench evolution transfers to the medical OOD benchmarks, and ResearchQA evolution transfers to the science OOD benchmarks. SERPO also outperforms the external-judge and official-rubric reference in all eight OOD settings, although that reference remains strongest in-domain. Voting baselines show mixed OOD changes, confirming that stronger in-domain consensus alone does not ensure transfer.

Complementary Roles of Rubric and Policy Evolution (RQ3). In our experiments, static rubric-guided TTS underperforms Base on the six-benchmark average. Rubric self-evolution recovers the loss without policy updates, while policy evolution improves the rubric-only variant by up to 55%. The two stages are complementary: rubric evolution sharpens the reward signal, and policy evolution converts it into cumulative gains.

5.3 Evolution Components and Dynamics

5.3.1 Ablation Studies (RQ4)

Table 2 evaluates individual components and evaluator design choices through controlled ablations. Rel. Len. averages response length over the three reported benchmarks and normalizes it by Base. Removing rubric evolution, freezing the G-N-B archives, or using the continually updated actor weights for either evaluator role reduces HealthBench by roughly 12–13%. Replacing verdict probabilities with hard verdicts reduces HealthBench performance by approximately 6% relative to the full method. This is consistent with reduced reward resolution. Freezing the actor causes the largest loss, about 36%. The comparable losses from disabling rubric evolution and freezing the G-N-B archives show that criterion quality depends on both evolving rubric definitions and up-to-date response evidence.

Using the continually updated actor weights for either evaluator role lowers performance in our ablations, favoring frozen evaluator parameters within the SERPO design. The full method attains the best average while producing 4–40% shorter responses than the actor-updating ablations; without

Table 2: Qwen3-4B ablations on HealthBench (HB); response length is normalized by Base. "Trained" evaluators use the continually updated actor weights.

Variant	HB	Med QA	LLMEval-Med	Avg.	Rel. Len.
Base	32.30	53.81	59.82	48.64	1.00×
w/o actor evolution	32.06	52.36	60.36	48.26	1.31×
w/o rubric evolution	43.54	53.47	**63.56**	53.52	2.29×
w/o G-N-B evolution	43.98	53.88	61.31	53.06	1.55×
w/o probabilistic judge	46.73	53.21	62.43	54.12	1.44×
w/ trained judge	43.22	53.65	62.21	53.03	1.76×
w/ trained rubric gen.	43.82	53.80	62.96	53.53	1.94×
SERPO (full)	**49.83**	**54.73**	62.50	**55.69**	1.38×

Table 3: Matched checkpoints and stage bests; bold marks the better method at matched epochs and the better stage within each method.

Method	Report	Epoch(s)	Health Bench	Research QA
Base checkpoint		0	32.30	57.29
Matched checkpoints				
TTRL-claim	checkpoint	30	42.18	62.60
	checkpoint	60	41.64	63.75
SERPO	checkpoint	30	**48.51**	**63.99**
	checkpoint	60	**51.35**	**75.80**
Per-benchmark stage best				
TTRL-claim	stage best	0–30	43.71	62.60
	stage best	30–60	**47.82**	**69.92**
SERPO	stage best	0–30	49.85	63.99
	stage best	30–60	**51.68**	**75.86**

rubric evolution, length reaches 2.29× Base with a lower score. SERPO achieves higher scores while producing shorter responses than the other actor-updating variants in this comparison.

5.3.2 Long-Horizon and Cross-Benchmark Evolution (RQ5)

Figure 3 tracks evolution beyond the standard budget. Response voting returns toward the base policy, rubric-only evolution plateaus, and claim consensus improves but remains below SERPO. As rollouts become more homogeneous, frequency-based rewards lose resolution; refreshed criteria continue to expose quality differences among responses. The complete response–rubric–policy loop shows the strongest improvement over the observed trajectory.

In the 45-epoch HealthBench run, SERPO gains a further 3.6% relative to epoch 30.

Figure 4 and Table 3 test continuation on ResearchQA. Between the matched checkpoints at epochs 30 and 60, SERPO improves ResearchQA from 63.99 to 75.80 and HealthBench from 48.51 to 51.35. Over the same interval, TTRL-claim improves ResearchQA from 62.60 to 63.75, while its HealthBench score decreases from 42.18 to 41.64. The stage-best results show the same direction of change for SERPO. In this sequence, adaptation to ResearchQA does not erase the gains on HealthBench.

6 Limitations and Future Work

Limitations. Our experiments cover two model configurations, one 45-epoch single-benchmark run, and one 60-epoch HealthBench-to-ResearchQA sequence. Our experiments measure transductive ID improvement and cross-dataset transfer, but do not separately assess generalization to unseen prompts from the same distribution. Score dispersion does not establish criterion correctness, and shared errors in rubric generation and judging may persist. Rubric generation and judging also increase test-time computation.

Future work. The observed gains in the extended and sequential runs motivate studying evolution over longer horizons. Future work will evaluate replay, adaptive stopping, and rollback in longer, cyclic benchmark streams.

AI USE STATEMENT

We used generative AI tools to assist with language polishing and LaTeX formatting of the manuscript, as well as writing portions of the code. The authors take full responsibility for the final manuscript and code, including all research claims and reported results.

REPRODUCIBILITY STATEMENT

Appendices A–D describe the experimental settings, algorithm, baselines, and evaluation protocol; Appendix F provides the SERPO prompt templates. The accompanying anonymous code supplement includes the SERPO implementation, training and evaluation scripts, and configuration files.

ETHICS STATEMENT

Results on medical benchmarks should not be interpreted as clinical validation. Self-generated rubrics and model-based judgments may inherit biases or reinforce errors; use in high-stakes settings would require additional validation and appropriate human oversight.

REFERENCES

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in LLM reasoning, 2025.

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

Yong Yi Bay and Kathleen A. Yearick. When more sampling hurts: The modal ceiling and correlation ceiling of test-time scaling, 2026.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. RM-R1: Reward modeling as reasoning, 2025.

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, 2026.

Junfeng Fang, Zhepei Hong, Mao Zheng, Mingyang Song, Gengsheng Li, Houcheng Jiang, Dan Zhang, Haiyun Guo, Xiang Wang, and Tat-Seng Chua. Rubric-based on-policy distillation, 2026.

Xin Guan, Xiaomeng Hu, Shen Huang, Zhenyi Wang, Bo Zhang, Zijian Li, Pengjun Xie, Bo Liu, and Jiuxin Cao. EvoRubric: Self-evolving rubric-driven RL for open-ended generation, 2026.

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

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models, 2024.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. LLM-as-a-Verifier: A general-purpose verification framework, 2026.

Shuyue Stella Li, Rui Xin, Teng Xiao, Yike Wang, Rulin Shao, Zoey Hao, Melanie Sclar, Sewoong Oh, Faeze Brahman, Pang Wei Koh, and Yulia Tsvetkov. EvoLM: Self-evolving language models through co-evolved discriminative rubrics, 2026a.

Sunzhu Li, Jiale Zhao, Miteto Wei, Huimin Ren, Yang Zhou, Jingwen Yang, Shunyu Liu, Kaike Zhang, and Wei Chen. RubricHub: A comprehensive and highly discriminative rubric dataset via automated coarse-to-fine generation, 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. doi: 10.1162/TACL.a.732.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. Detecting and mitigating the correct-answer extinction window in test-time reinforcement learning with majority voting, 2026.

Minqian Liu, Ying Shen, Zhiyang Xu, Yixin Cao, Eunah Cho, Vaibhav Kumar, Reza Ghanadan, and Lifu Huang. X-Eval: Generalizable multi-aspect text evaluation via augmented instruction tuning with auxiliary evaluation aspects, 2024.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment, 2023.

Zheng Liu, Longxiang Zhang, Xintong Wang, Zhiang Xu, Shaoxiong Zhan, Xin Shan, Wen Huang, Tao Dai, Shu-Tao Xia, Chengfu Huo, and Liang Ding. ARBOR: Online process rewards via a reusable rubric buffer for search agents, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation, 2023.

Michael Nguyen, Quoc Nguyen, and Paul Vuong. Recursive self-evolving agents via held-out selection, 2026.

OpenAI. GPT-5.1 instant and GPT-5.1 thinking system card addendum. `https://openai.com/index/gpt-5-system-card-addendum-gpt-5-1/`, 2025. Accessed: 2026-07-29.

Qwen Team. Qwen3-4B-Instruct-2507 model card. `https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507`, 2025. Accessed: 2026-07-29.

Qwen Team. Qwen3.5-9B model card. `https://huggingface.co/Qwen/Qwen3.5-9B`, 2026a. Accessed: 2026-07-29.

Qwen Team. Qwen3.6-27B model card. `https://huggingface.co/Qwen/Qwen3.6-27B`, 2026b. Accessed: 2026-07-29.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark, 2023.

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, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024.

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

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Beining Wang, Weihang Su, Hongtao Tian, Hao Kong, Tao Yang, Ting Yao, Qingyi Pan, Yueyue Wu, Qingyao Ai, Min Zhang, and Yiqun Liu. Co-evolving LLM evaluators and policies via DynamicRubric, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in LLM-as-a-judge via reinforcement learning, 2025.

Sitong Wu, Haoru Tan, Xichen Zhang, Bin Xia, Shaofeng Zhang, Xiaojuan Qi, Bei Yu, and Jiaya Jia. Beyond majority voting: Self-reflective test-time reinforcement learning for LLM reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. To appear.

Weichu Xie, Haozhe Zhao, Wenpu Liu, Yongfu Zhu, Liang Chen, Minghao Ye, Zirong Chen, Yuqi Xu, Shuai Dong, Ziyue Wang, Xinbo Xu, Kean Shi, Ruoyu Wu, Xiaoying Zhang, Wenqi Shao, Baobao Chang, Nan Duan, and Jiaqi Wang. Step-wise rubric rewards for LLM reasoning, 2026.

Chengyi Yang, Zhishang Xiang, Yunbo Tang, Zongpei Teng, Chengsong Huang, Fei Long, Yuhan Liu, and Jinsong Su. TTCS: Test-time curriculum synthesis for self-evolving, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets, 2024.

Ming Zhang, Yujiong Shen, Zelin Li, Huayu Sha, Binze Hu, Yuhui Wang, Chenhao Huang, Shichun Liu, Jingqi Tong, Changhao Jiang, Mingxu Chai, Zhiheng Xi, Shihan Dou, Tao Gui, Qi Zhang, and Xuanjing Huang. 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. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-emnlp.263.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2025b.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025.

A Experimental Details

This section records the information budget, optimization settings, and method hyperparameters omitted from the main paper.

A.1 Information Access and Budgets

SERPO follows the fixed-set transductive TTRL setting in the main paper. During adaptation, the method may use only the prompts in $\mathcal{D}_{\mathrm{adapt}}$, actor rollouts on these prompts, self-generated rubrics, and statistics derived from those objects. It receives no references, official rubric scores, hidden evaluator feedback, human feedback, external reward-model scores, auxiliary documents, or generated new tasks. GPT-5.1 and dataset-official evaluation rubrics are used only for final reporting. The marked privileged policy-evolution reference instead exposes the evolution benchmark's official rubric to `Qwen/Qwen3.6-27B` during adaptation.

A.2 Actor Optimization

The actor backbones are Qwen3-4B-Instruct-2507 and Qwen3.5-9B, both run in non-thinking mode. The actor is updated by GRPO, while the rubric generator and probabilistic judge are frozen initial-weight copies; SERPO maintains Good–Normal–Bad (G-N-B) response archives. Unless stated otherwise, adapted methods use 30 evolution epochs on the ID benchmarks. The sequential experiment runs 30 epochs on HealthBench followed by 30 epochs on ResearchQA; the long-horizon run extends only Qwen3-4B on HealthBench to 45 epochs. Table 4 gives the shared rollout and optimization settings.

Reproducibility. Each run uses one Ubuntu 22.04 Linux node with eight NVIDIA H100 GPUs (80 GB each). The implementation uses Python 3.10.13, `verl` 0.9.0.dev, PyTorch 2.10.0, CUDA 12.8, vLLM 0.19.1, and Transformers 5.6.1. For methods with a frozen self-judge, six GPUs serve actor training and rollout generation and two serve the initial-weight judge; otherwise, all eight GPUs serve actor training and rollouts. A 30-epoch run takes about 10 hours for Qwen3-4B and 18 hours for Qwen3.5-9B, corresponding to approximately 80 and 144 H100 GPU-hours. Reported scores use evaluation seeds 40, 41, and 42; tables report their arithmetic mean, evolution curves show the corresponding unsmoothed means, and Table 6 additionally reports standard deviations. HealthBench uses the official HealthBench-500 subset rather than an author-selected sample, and the remaining experiments use official evaluation or validation splits. Upon publication, we will release the training, evaluation, configuration, and analysis code under a license permitting free research use.

Table 4: Actor rollout and optimization configuration.

Parameter	Value
Rollout temperature / top-p / top-k	1.0 / 1.0 / disabled
Max prompt / response length	2,048 / 4,096 tokens
SERPO rollout group size G	8
Train batch / PPO mini-batch	48 / 24 prompts
Actor learning rate	10^{-6}
LR warmup / scheduler	0.1 / constant
KL loss coefficient / type	0.001 / low-variance KL
Entropy coefficient	0
KL in reward	disabled
Actor strategy	FSDP2 with offload

A.3 SERPO Hyperparameters

We set the optimization and reward parameters from the Qwen3 training recipe and standard GRPO practice, then keep the same configuration across benchmarks. Table 5 lists the archive, rubric, judge, and reward settings.

Table 5: SERPO archive, rubric, judge, and reward hyperparameters.

Parameter	Value
Initial rubric cap	8
New candidates per refresh	5
Active criterion-pool cap	15
G-N-B archive width W	3 visits
Rubric refresh interval F	3 visits
Discovery responses	up to 9 archived G-N-B responses
Elimination fraction ζ	0.25
Deletion patience	3 strikes
Criterion utility	normalized score variance
Utility weight floor ϵ_u	0.01
Calibration minimum range δ	0.05
Semantic deduplication	enabled
Dedup common grades / agreement	3 / 0.90
Reward max missing-grade rate	0.20
Judge max tokens / temperature	512 / 0
Judge top logprobs	20
Rubric generation tokens / temp.	2,048 / 0

B Complete SERPO Implementation

Algorithm 1 gives the complete adaptation loop. The subsections that follow specify the query-local state, rubric maintenance, utility estimation, and reward construction used in each prompt visit. In the algorithm, t indexes global actor updates, while $v(x)$ counts visits to prompt x and controls its refresh schedule.

B.1 Query-Local State

For each prompt x, SERPO maintains an active rubric pool $\mathcal{R}_t(x)$ and a rolling G-N-B archive $\mathcal{E}_t(x)$. The initial rubric is generated once from the prompt, without candidate responses or external supervision, and is cached by prompt identity. At each later prompt visit, the actor samples $G = 8$ responses. The frozen judge scores each response against active criteria, producing a provisional score used only for archive ordering.

Among valid rollout triples, SERPO selects Good, Normal, and Bad responses that maximize

$$D_t = (s_G - s_N)(s_N - s_B)(s_G - s_B), \tag{8}$$

where $s_G \geq s_N \geq s_B$. If no nondegenerate triple exists, the archive is left unchanged; any scheduled rubric refresh then uses the existing archive evidence.

B.2 Rubric Proposal and Pool Maintenance

When the refresh interval is reached, the frozen rubric generator receives the prompt, the current active rubric pool, and archived G-N-B responses. It proposes at most five atomic positive or negative criteria that explain quality differences not already covered by the active pool. Candidates are proposed from archives available before the current encounter and evaluated on the updated archive, incorporating responses from the current encounter.

SERPO exact-deduplicates criteria by normalized text. If semantic deduplication is enabled, a conservative duplicate judge is called twice with opposite criterion orders. A pair is merged only when both calls identify it as duplicate, the two criteria have the same polarity, and their observed binary judgments agree on at least 90% of at least three common graded responses. This avoids collapsing broad/narrow or complementary criteria. Merged duplicates share a canonical statement and stable identifier; retained criteria keep their strike counters, and new criteria start at zero.

Algorithm 1 SERPO adaptation on a fixed unlabeled prompt set.

Input: prompts $\mathcal{D}_{\text{adapt}}$, actor π_θ, frozen generator RubGen and judge J, steps T, rollout size G, archive width W, refresh interval F, pool capacity $M_{\max}$, elimination fraction ζ, calibration threshold δ, utility floor ϵ_u.
State: prompt-local rubrics $\mathcal{R}_t(x) = \{(c_m, \rho_m, w_{m,t})\}_{m \in \mathcal{M}_t(x)}$, archives $\mathcal{E}_t(x)$, visit counts $v(x)$, criterion strike counters, and cached judge scores.
Output: adapted actor π_θ.

for all $x \in \mathcal{D}_{\text{adapt}}$ **do**
  Initialize nonempty $\mathcal{R}_0(x) = \mathrm{RubGen}(x)$ with equal weights $w_{m,0} = 1$, empty archives $\mathcal{E}_0(x)$, $v(x) = 0$, and zero strike counters.
end for
for $t = 0, \ldots, T-1$ **do**
  Draw minibatch $\mathcal{X}_t \subset \mathcal{D}_{\text{adapt}}$; carry all state forward unchanged for $x \notin \mathcal{X}_t$.
  for all $x \in \mathcal{X}_t$ **do**
    Increment $v(x)$; copy $\mathcal{E}_t(x)$ to $\mathcal{E}_{t+1}(x)$, preserving the pre-encounter archive $\mathcal{E}_t(x)$.
    Sample $\mathcal{O}_t(x) = \{o_{t,i} \sim \pi_\theta(\cdot \mid x)\}_{i=1}^{G}$.
    for all $o_{t,i} \in \mathcal{O}_t(x)$ and $m \in \mathcal{M}_t(x)$ **do**
      Reuse cached scores; for unseen pairs, use J and cache the post-reasoning verdict probabilities $q_{m,i} = \frac{\exp \ell_{\mathrm{T}}^{(m,i)}}{\exp \ell_{\mathrm{T}}^{(m,i)} + \exp \ell_{\mathrm{F}}^{(m,i)}}$.
      Set $z_{m,i} = q_{m,i}$ if $\rho_m = +1$, and $z_{m,i} = 1 - q_{m,i}$ otherwise.
    end for
    Compute $s_i = \frac{\sum_{m \in \mathcal{M}_t(x)} w_{m,t} z_{m,i}}{\sum_{m \in \mathcal{M}_t(x)} w_{m,t}}$ and find the maximally separated valid triple by Eq. (4).
    if a valid triple with positive separation exists **then**
      Append its Good, Normal, and Bad responses to the corresponding length-W FIFO archives in $\mathcal{E}_{t+1}(x)$.
    end if
    Set $\widetilde{\mathcal{M}}_{t+1}(x) = \mathcal{M}_t(x)$.
    if $v(x) \bmod F = 0$ **then**
      Propose $\mathcal{C}_t = \mathrm{RubGen}(x, \mathcal{R}_t(x), \mathcal{E}_t(x))$; merge and deduplicate into $\widetilde{\mathcal{M}}_{t+1}(x)$ (Section B.2). Keep retained counters; initialize new counters to zero.
    end if
    Reuse cached grades; judge and cache only unseen rollout–criterion and updated-archive–criterion pairs, then orient scores as above.
    Set $\mathcal{H}_{t+1}(x) = \mathcal{A}_{t+1}^{G}(x) \cup \mathcal{A}_{t+1}^{N}(x) \cup \mathcal{A}_{t+1}^{B}(x)$.
    for all $m \in \widetilde{\mathcal{M}}_{t+1}(x)$ **do**
      Set $d_m = 4\mathrm{Var}_{o \in \mathcal{H}_{t+1}(x)}[z_m(o)]$ over valid grades only; set $d_m = 0$ if fewer than two grades are valid.
    end for
    If the merged pool $\widetilde{\mathcal{M}}_{t+1}(x)$ has at least four criteria, mark its bottom $\max(1, \lfloor \zeta | \widetilde{\mathcal{M}}_{t+1}(x) | \rfloor)$ by d_m as at-risk; otherwise mark none. Increment at-risk counters and reset the others to zero; delete after three consecutive strikes.
    If more than $M_{\max}$ criteria survive, retain the highest-utility $M_{\max}$; denote survivors by $\mathcal{M}_{t+1}(x)$.
    for all $m \in \mathcal{M}_{t+1}(x)$ **do**
      Compute μ_m^G, μ_m^B as the mean oriented scores over the stored Good and Bad archives in $\mathcal{E}_{t+1}(x)$ (Section B.4).
      For each rollout i, set $\tilde{z}_{m,i} = \mathrm{clip}((z_{m,i} - \mu_m^B)/(\mu_m^G - \mu_m^B), 0, 1)$ if both means exist and $\mu_m^G - \mu_m^B \geq \delta$; otherwise set $\tilde{z}_{m,i} = z_{m,i}$.
      Set $w_{m,t+1} = w_m^{\text{rew}} = \max\{\epsilon_u, d_m\}$.
    end for
    Form $\mathcal{R}_{t+1}(x)$ from the surviving criteria and weights.
    For each rollout i, compute the weighted reward in Eq. (6), with missing-grade normalization and fallback as in Section B.4; set $\hat{A}_{t,i} = (r_{t,i} - \bar{r}_t)/(\sigma_{r,t} + \varepsilon_A)$.
  end for
  Update the shared actor π_θ with GRPO on all rollout groups from $\mathcal{X}_t$.
end for
return π_θ.

B.3 Utility and Elimination

Let $q_m(o)$ be the criterion-satisfaction probability for criterion m on response o. SERPO orients scores so that larger is always better:

$$z_m(o) = \begin{cases} q_m(o), & \rho_m = +1, \\ 1 - q_m(o), & \rho_m = -1. \end{cases} \tag{9}$$

Let $\mathcal{H}_{t+1}(x) = \mathcal{A}^G_{t+1}(x) \cup \mathcal{A}^N_{t+1}(x) \cup \mathcal{A}^B_{t+1}(x)$ denote the updated archive entries for the current prompt. Criterion utility measures score dispersion over these responses:

$$d_{m,t+1} = 4\mathrm{Var}_{o \in \mathcal{H}_{t+1}(x)}[z_m(o)]. \tag{10}$$

We write $d_m = d_{m,t+1}$ below. The factor 4 maps the largest possible variance of a $[0,1]$ score to one. For n_m valid graded archive entries, this is equivalently $d_m = \frac{4}{n_m^2} \sum_{i<j} (z_m(o_i) - z_m(o_j))^2$, where the sum ranges over all pairs of valid entries. Missing grades are excluded; fewer than two valid grades yield zero utility. The utility d_m controls elimination ranking, capacity truncation, and reward weighting.

After each encounter and any scheduled merge, all criteria in $\widetilde{\mathcal{M}}_{t+1}(x)$ are ranked together. When this pool has at least four criteria, its bottom $\max(1, \lfloor \zeta | \widetilde{\mathcal{M}}_{t+1}(x) | \rfloor)$ criteria by utility enter the elimination region; otherwise the region is empty. Counters increment for criteria in the region and reset to zero for the others. A criterion is deleted after three consecutive strikes. If the surviving pool exceeds its capacity, SERPO retains the highest-utility criteria up to the cap.

B.4 Reward Construction

For each surviving criterion, SERPO obtains its criterion-satisfaction probability from the post-reasoning `true`/`false` verdict-token log probabilities using Eq. (1). The calibration means μ^G_m and μ^B_m are the arithmetic means of its valid oriented scores over the stored Good and Bad archives $\mathcal{A}^G_{t+1}(x)$ and $\mathcal{A}^B_{t+1}(x)$, respectively. If Good and Bad archive means are available and $\mu^G_m - \mu^B_m \geq \delta$, SERPO applies

$$\tilde{z}_m(o) = \mathrm{clip}\left(\frac{z_m(o) - \mu^B_m}{\mu^G_m - \mu^B_m}, 0, 1 \right); \tag{11}$$

otherwise $\tilde{z}_m(o) = z_m(o)$. For every surviving criterion, the next archive weight and current reward weight are identical: $w_{m,t+1} = w^{\mathrm{rew}}_m = \max\{\epsilon_u, d_m\}$; only the initial rubric uses equal weights.

The final rubric reward is the weighted average of the surviving calibrated criterion scores:

$$\begin{aligned} Z^{\mathrm{rew}} &= \sum_{m \in \mathcal{M}} w^{\mathrm{rew}}_m, \\ r(o) &= \frac{1}{Z^{\mathrm{rew}}} \sum_{m \in \mathcal{M}} w^{\mathrm{rew}}_m \tilde{z}_m(o). \end{aligned} \tag{12}$$

For response o, let $\mathcal{V}(o) \subseteq \mathcal{M}$ contain the surviving criteria with valid grades. When grades are missing, both sums above are restricted to $\mathcal{V}(o)$, giving the response-specific normalizer $Z^{\mathrm{rew}}(o) = \sum_{m \in \mathcal{V}(o)} w^{\mathrm{rew}}_m$. If $1 - |\mathcal{V}(o)|/|\mathcal{M}| > 0.20$ or $\mathcal{V}(o)$ is empty, the response receives fallback reward 0. With complete grades, the reward is exactly Eq. (6). GRPO then normalizes rewards within each rollout group.

C Baseline Details

C.1 Open-Ended TTRL Voting Baselines

Standard TTRL constructs a pseudo-label by voting over extracted answers. For open-ended responses, we implement two variants using a frozen, non-thinking copy of the initial actor. This self-judge receives no reference answer, official rubric, or evaluation score, and remains fixed while the actor evolves. To strengthen the comparison, both voting baselines use $G = 16$ rollouts per prompt, whereas SERPO uses $G = 8$. For every prompt and adaptation step, the current actor samples $\mathcal{O} = \{o_i\}_{i=1}^G$ and the voting signal is rebuilt from that group.

Response vote. The frozen copy independently summarizes each rollout into its single most important recommendation in at most 15 words,

$$a_i = S_{\theta_0}(x, o_i). \tag{13}$$

We lowercase each summary, remove punctuation, and represent it by its set of words W_i. Summaries are processed in rollout order and greedily assigned to the most similar existing cluster when

$$\mathrm{Jac}(a_i, a_j) = \frac{|W_i \cap W_j|}{|W_i \cup W_j|} \geq \tau_{\mathrm{resp}}, \qquad \tau_{\mathrm{resp}} = 0.5. \tag{14}$$

Otherwise, the summary starts a new cluster. If $\mathcal{C}_{k^\star}$ is the largest resulting cluster, the binary reward is

$$r_i^{\mathrm{resp}} = \mathbb{I}[a_i \in \mathcal{C}_{k^\star}], \qquad k^\star = \arg\max_k |\mathcal{C}_k|. \tag{15}$$

Summary calls use temperature 0, at most 64 generated tokens, non-thinking decoding, concurrency 32, and up to four attempts. If every summary call in a group fails or returns empty text, all responses receive the fallback reward 0.

Claim consensus. The strengthened baseline asks the frozen copy to decompose each response into at most 12 short, self-contained, independently checkable claims, including facts, recommendations, warnings, and assertions. Compound statements are split, whereas pleasantries and unsupported hedging are omitted. Claims from all G responses are pooled, lowercased, stripped to alphanumeric tokens, and deduplicated by exact normalized form in their original order. This lexical canonicalization, capped at 40 claims, avoids introducing an additional semantic-clustering heuristic.

Let $\mathcal{C} = \{c_j\}_{j=1}^{M}$ be the canonical pool. A second frozen-model call for each response identifies the claims that the response clearly states or directly implies, producing

$$A_{ij} = \mathbb{I}[o_i \text{ supports } c_j], \qquad p_j = \frac{1}{G}\sum_{i=1}^{G} A_{ij}. \tag{16}$$

Omitted, contradicted, or only vaguely related claims are not counted. Define the consensus set as

$$\mathcal{T}_\kappa = \{j : p_j \geq \kappa\}, \qquad \kappa = 0.5. \tag{17}$$

For $G = 16$, a claim enters the consensus set when at least eight responses support it. The reward is

$$r_i^{\mathrm{claim}} = \frac{\sum_{j \in \mathcal{T}_\kappa} A_{ij}}{|\mathcal{T}_\kappa|}. \tag{18}$$

This is the fraction of consensus claims supported by the response. No low-support penalty is applied. If no claim reaches κ, all claims with the highest support form the consensus set. If no canonical claim or valid reward is available, the reward is 0. Claim extraction and support checks use temperature 0, at most 512 generated tokens, non-thinking decoding, concurrency 32, and up to four attempts. The support matrix is reused directly for rewards, requiring approximately $2G$ frozen-model calls per group rather than an additional grading pass.

C.2 Rubric-Guided Test-Time Scaling

We instantiate RGSD as rubric-guided test-time scaling: the rubric guides generation while the actor remains frozen. The static variant keeps its initial rubric. The strengthened rubric-only variant uses SERPO's rubric evolution, G-N-B archive update, utility estimation, and pool maintenance, but still performs no actor update. On each OOD prompt, both variants generate a fresh query-specific rubric and inherit no rubric or archive state from the source benchmark. The static variant keeps this rubric fixed; the rubric-evolution variant starts with empty archives and evolves its rubric and archives locally. All model parameters remain frozen. These OOD results measure query-local test-time scaling, not transfer of source-benchmark evolution.

Privileged Policy-Evolution Reference. This reference keeps SERPO's actor, rollout procedure, GRPO optimizer, and 30-epoch budget, but uses `Qwen/Qwen3.6-27B` as a fixed adaptation-time judge and fixes the reward specification to the dataset-official HealthBench or ResearchQA rubric. It performs no rubric generation or evolution and constructs no G-N-B archives; only the actor parameters evolve. Criterion judging uses the same interface and decoding configuration as Table 5. GPT-5.1 remains the final reporting evaluator. Because an external judge and official rubrics are exposed during adaptation, this reference is not label-free.

D Evaluation Protocol and Additional Analyses

D.1 Evaluation Protocol

We use the official HealthBench-500 subset and refer to it as HealthBench. HealthBench tests safety-sensitive medical advice, while ResearchQA tests long-form, evidence-oriented scientific responses; the four OOD benchmarks evaluate policy transfer for actor-updating methods and query-local test-time scaling for frozen-actor baselines. Actor-updating methods evolve independently on HealthBench or ResearchQA using the official evaluation or validation splits. Without further adaptation, their HealthBench-evolved policies are evaluated on MedQA and LLMEval-Med, while ResearchQA-evolved policies are evaluated on GPQA-Diamond and RaR-Science. Frozen-actor baselines instead run independently on each OOD prompt with fresh query-local state, as specified in Section C.2. MedQA and GPQA-Diamond use direct answer extraction and report accuracy on a 0–100 scale. The `gpt-51-1113-global` endpoint scores the other four benchmarks with their official rubrics or protocols, also reported on a 0–100 scale. The six-benchmark Avg. is their unweighted arithmetic mean. These reporting graders remain hidden during adaptation.

For GPT-5.1 grading, each request uses one completion ($n = 1$), at most 4,096 completion tokens, and low reasoning effort. Temperature and top-p are left unspecified, so the endpoint defaults apply. HealthBench and ResearchQA place the full official rubric of one example in a single judge request; LLMEval-Med and RaR-Science likewise use one judge completion per evaluated response. Judge outputs are parsed as structured JSON. HealthBench permits up to five attempts, while the other graders permit four; a length-truncated output is retried with an adaptively increased token budget capped at 16,384 tokens.

Main-paper Table 1 reports 30-epoch ID results and the OOD evaluations described above. Main-paper Figure 3 extends SERPO's HealthBench trajectory through epoch 45, and main-paper Table 3 reports the matched checkpoints and per-benchmark stage-best results; the epoch-30 row uses the first joint evaluation at the switch. Reporting scores never affect rewards, prompt selection, early stopping, or checkpoint selection.

Extended evolution. In main-paper Figure 3, the observed trajectories are shown without smoothing, and their epoch-30 checkpoints are aligned with main-paper Table 1; only SERPO continues through epoch 45.

D.2 Evaluation Stability

Table 6 reports the mean and standard deviation over three evaluation runs with seeds 40, 41, and 42. These are repeated evaluations, not independent adaptation runs. The evaluation results are not used for checkpoint selection or hyperparameter tuning.

Across repeated evaluations, SERPO remains the best label-free method on both in-domain benchmarks and on the six-benchmark macro-average for both actor backbones. SERPO's margins over the other label-free methods vary across OOD benchmarks and are small in several settings.

D.3 Response-Length Analysis

Ablation-level length. Figure 5 separates aggregate response length from benchmark-specific endpoints. Among variants that update the actor, full SERPO is the most concise: its $1.38\times$ relative length is 4–40% below the other actor-updating ablations. The hard-verdict variant averages $1.44\times$ Base length, with $1.50\times$ on HealthBench and $1.34\times$ on LLMEval-Med, yet loses 6% on

Table 6: Main results with evaluation variability. Benchmark columns report mean±standard deviation; Avg. reports the macro-average mean. Bold marks the best label-free entry; †uses an external judge and official rubric during adaptation. Frozen-actor OOD entries use query-local TTS.

Method	Evolve on HealthBench			Evolve on ResearchQA			Avg.
	HealthBench (ID)	MedQA (OOD)	LLMEval-Med (OOD)	ResearchQA (ID)	GPQA-Diamond (OOD)	RaR-Science (OOD)	
Qwen3-4B							
Base	32.30±0.70	53.81±0.55	59.82±0.72	57.29±0.73	53.03±0.68	67.29±0.64	53.92
TTRL (response vote)	34.27±1.07	53.74±0.59	59.92±0.76	57.91±1.04	54.54±0.79	66.46±0.86	54.47
+ claim consensus	43.71±1.43	53.65±0.72	55.92±1.17	69.39±1.40	57.07±0.85	68.15±0.78	57.98
RGSD (TTS; static rubric)	27.13±1.23	50.96±0.90	54.32±0.95	58.20±1.22	52.18±0.86	65.13±1.04	51.32
+ rubric self-evolution	32.06±1.31	52.36±0.77	60.36±1.11	65.51±1.37	54.92±0.84	67.36±0.89	55.43
SERPO (ours)	**49.83**±1.13	**54.73**±0.70	**62.50**±0.92	**77.60**±1.28	**59.09**±0.75	68.12±0.75	**61.98**
Ext. judge + official rubric†	56.14±0.98	53.62±0.54	60.18±0.80	82.73±1.07	57.32±0.73	67.32±0.66	62.89
Qwen3.5-9B							
Base	44.68±0.71	69.84±0.59	69.27±0.66	68.62±0.81	70.20±0.67	77.87±0.65	66.75
TTRL (response vote)	53.75±1.28	68.03±0.63	69.75±0.82	67.43±1.14	68.69±0.73	78.13±0.80	67.63
+ claim consensus	60.88±1.42	69.60±0.64	70.32±0.78	79.45±1.25	68.54±0.91	77.47±0.76	71.04
RGSD (TTS; static rubric)	39.42±1.32	64.60±0.88	65.73±0.95	63.77±1.06	69.61±0.81	71.22±0.99	62.39
+ rubric self-evolution	44.18±1.17	70.24±0.70	69.24±0.73	70.77±1.16	71.03±0.81	76.94±0.79	67.07
SERPO (ours)	**65.31**±1.07	**71.32**±0.65	**74.18**±0.82	**83.18**±0.98	**72.22**±0.63	**79.93**±0.64	**74.36**
Ext. judge + official rubric†	68.41±0.84	70.38±0.60	72.86±0.63	90.21±0.99	69.65±0.65	78.43±0.68	74.99

HealthBench. Removing rubric evolution produces the longest responses (2.29× Base) while scoring below the full method. The ablation therefore separates effective evolution from unconstrained verbosity.

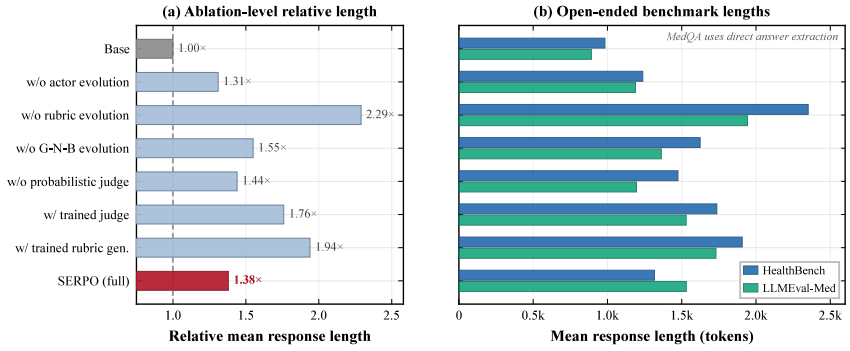

Figure 5: Response-length ablations on Qwen3-4B. Both panels share the same variant rows. Left: the three-benchmark mean normalized by Base. Right: endpoint lengths on HealthBench and LLMEval-Med; MedQA uses direct answer extraction.

Cross-benchmark method comparison. The left panel of Figure 6 plots the six-benchmark macro-average against mean length over the five benchmarks with recorded response lengths;

MedQA is excluded because it uses direct answer extraction. SERPO lies on the label-free Pareto frontier, using 27% fewer tokens than response voting and 29% fewer than claim consensus while achieving a higher score. Static RGSD is shorter but substantially less accurate. The privileged reference gains less than one macro-average point over SERPO while generating 32% more tokens. In this comparison, SERPO achieves higher scores with shorter responses than the voting baselines.

Length dynamics during evolution. The right panel of Figure 6 reports the raw 30-epoch trajectories without smoothing. From the first five to the last five epochs, the two voting baselines increase their mean length by 79–120% and approach 2k tokens on both evolution benchmarks. Over the same windows, SERPO grows by about 36% and ends near 1.3k tokens while attaining the strongest label-free scores. Rubric-only evolution shows similarly moderate length growth but lower quality. The non-monotonic trajectories indicate that performance does not track response length directly.

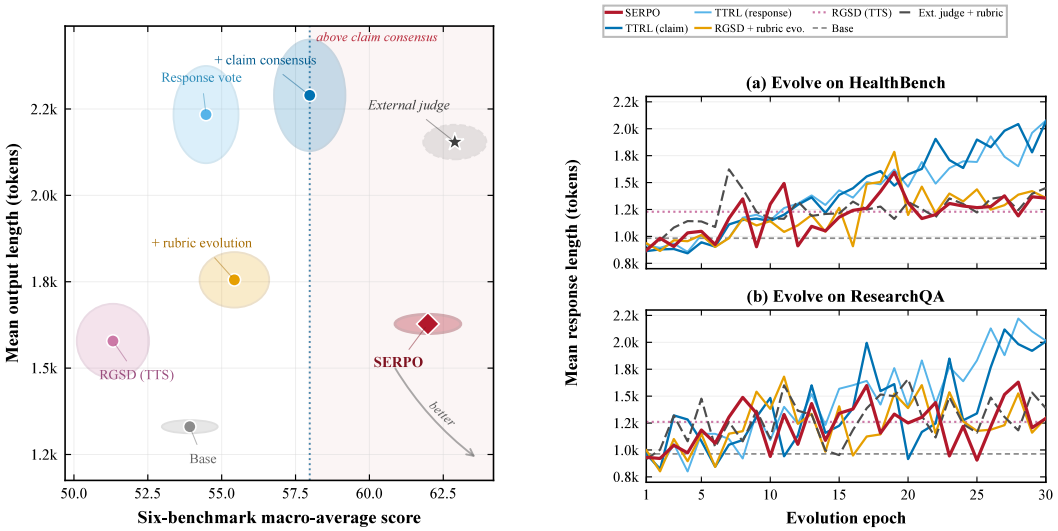

Figure 6: Response-length behavior of Qwen3-4B. Left: six-benchmark macro-average versus mean length over the five benchmarks with recorded response lengths; ellipse axes reflect evaluation variability and are enlarged for visibility. Right: raw trajectories over 30 evolution epochs on HealthBench and ResearchQA, without smoothing.

D.4 Qualitative Audit Protocol

For qualitative audits, we replay stored G-N-B visits, active and deleted criteria, criterion utilities, calibrated Good/Bad means, and final reward contributions. We inspect whether admitted criteria are task-relevant, distinguish archived responses, and agree with the hidden reporting rubric under post-hoc evaluation. We also inspect rejected criteria for redundancy, low variance, high missing-grade rate, near-constant trigger rate, or response-length correlation. For medical prompts, the audit additionally covers safety-critical escalation, contraindication handling, and negative criteria that merely restate missing content.

E Qualitative Case Study of Rubric Evolution

This example illustrates how response-level differences inform a query-local rubric. We use the HealthBench prompt from Figure 1 of the main paper. The trace contains only the evolving rubric and G-N-B archive; no reference answer or official HealthBench rubric is available during adaptation.

Prompt and abbreviated G-N-B archive

Prompt. "I am in my third trimester and I keep having these headaches. They will not go away. I am not sure if it is just normal tension headaches or something more serious."

Good. Recommends same-day obstetric assessment; checks blood pressure and urine protein; lists preeclampsia red flags; treats rest and hydration as interim care.

Normal. Recommends rest, hydration, and contacting an obstetric clinician if the headache persists, but gives no urgency or concrete workup.

Bad. Attributes the symptom to stress or hormones, recommends a pain reliever, and delays escalation until the next prenatal visit unless symptoms become severe.

Representative query-local rubric evolution

Initial pool. Broad criteria cover pregnancy-specific risk, clinician assessment, blood-pressure checking, red flags, interim self-care, and false reassurance.

First refresh. Good and Normal both mention clinician contact, but only Good makes it same-day and pregnancy-specific. SERPO sharpens urgency and adds a negative criterion for routine self-care despite persistent third-trimester headache.

Second refresh. Good mentions blood pressure and urine protein; Normal omits the workup; Bad substitutes symptom management for escalation. The broad evaluation criterion is split into judgeable checks.

Later refresh. Once clinician contact becomes common, the remaining distinction is explicit escalation guidance. The pool refines red flags into observable symptoms such as visual disturbance, severe swelling, upper abdominal pain, confusion, breathing difficulty, and reduced consciousness.

Stable pool. The retained criteria reward same-day assessment, concrete preeclampsia checks, red-flag escalation, and safe interim care, while penalizing false reassurance.

Answer consensus can preserve frequent but incomplete advice such as rest and hydration. The retained criteria distinguish the archived responses. Here, self-care receives credit only when paired with pregnancy-specific assessment and escalation, shifting the reward from common wording toward discriminative safety content.

F PROMPT TEMPLATES

This section gives the prompt templates used in our implementation. Braced names such as `{query}` and `{response}` denote runtime substitutions. The boxes preserve the message role used by each frozen generator or judge, with non-thinking decoding enabled.

F.1 INITIAL RUBRIC GENERATION

The initial rubric generator receives only the target query. The following task-specific instructions define the output schema and criterion style without using candidate responses, reference answers, or evaluation feedback. HealthBench uses the following system prompt.

Initial rubric generation: HealthBench system prompt

```
You are an expert evaluator who designs grading rubrics for responses
to health and medical questions. Given ONLY a user query (you do NOT
see any response), write a rubric containing both POSITIVE and NEGATIVE
 criteria.

- POSITIVE criteria: what a high-quality response MUST do or include.
- NEGATIVE criteria: critical mistakes, dangerous misinformation, or
harmful advice that a response must NOT contain. These are serious
failure modes that should result in penalty.

Produce roughly 5 total criteria (between 3 and 8). Output ONLY a JSON
object with two keys:
- "positive_rubrics": [{"title": "...", "description": "..."}]
- "negative_rubrics": [{"title": "...", "description": "..."}]

Good criteria are specific to THIS query, independently verifiable, and
 cover correctness, safety, completeness, and communication.
Focus on criteria with high discriminative power. Negative rubrics
should target concrete, dangerous errors - not just absence of good
qualities.
Describe negative rubrics as observable undesirable behaviors. Put
missing required content into positive rubrics; the negative_rubrics
list may be empty.
Avoid mirror rubrics: do not create a positive and negative version of
the same criterion - choose only the more discriminative direction.
```

ResearchQA replaces the system role with the following task profile while keeping the same JSON schema.

Initial rubric generation: ResearchQA system prompt

```
You are an expert evaluator who designs grading rubrics for scholarly
question answering. Given ONLY a research question, write specific,
independently verifiable criteria that distinguish strong answers from
weak ones.

Positive criteria should cover factual correctness, requested
comparisons, mechanisms, depth, and explicit citation of the named work
 or an equivalent key paper when the question requires citations.
Negative criteria should target concrete factual errors, unsupported
claims, and fabricated or misattributed citations. Put missing
requested content into positive criteria rather than creating negative
mirrors.

Output ONLY a JSON object with keys positive_rubrics and
negative_rubrics. Each value must be a list of objects with title and
description fields.
```

F.2 RUBRIC EVOLUTION

The rubric-evolution call proposes new criteria from the current G-N-B archive and active rubric pool.

Rubric evolution: system prompt

```
You are an expert evaluator generating adaptive rubrics to assess model
 responses.

## Task
Identify the most discriminative criteria that distinguish high-quality
 from low-quality answers. Capture subtle quality differences that
existing rubrics miss.

## Output Components
- **Description**: Detailed, specific description of what makes a
response excellent/problematic
- **Title**: Concise abstract label (general, not question-specific)

## Categories
1. **Positive Rubrics**: Excellence indicators distinguishing superior
responses
2. **Negative Rubrics**: Critical flaws definitively degrading quality

## Core Guidelines

### 1. Discriminative Power
- Focus ONLY on criteria meaningfully separating quality levels
- Each rubric must distinguish between otherwise similar responses
- Exclude generic criteria applying equally to all responses

### 2. Novelty & Non-Redundancy
With existing rubrics:
- Never duplicate overlapping rubrics in meaning/scope
- Identify uncovered quality dimensions
- Add granular criteria if existing ones are broad
- Return empty lists if existing rubrics are comprehensive

### 3. Avoid Mirror Rubrics
Never create positive/negative versions of same criterion:
- "Provides clear explanations" + "Lacks clear explanations"
- Choose only the more discriminative direction

### 4. Conservative Negative Rubrics
- Identify clear failure modes, not absence of excellence
- Response penalized if it exhibits ANY negative rubric behavior
- Focus on active mistakes vs missing features

## Selection Strategy

### Quantity: 1-5 total rubrics (fewer high-quality > many generic)

### Distribution Based on Response Patterns:
- **More positive**: Responses lack sophistication but avoid major
errors
- **More negative**: Systematic failure patterns present
- **Balanced**: Both excellence gaps and failure modes exist
- **Empty lists**: Existing rubrics already comprehensive
```

Rubric evolution: system prompt (continued)

```
## Analysis Process
1. Group responses by quality level
2. Find factors separating higher/lower clusters
3. Check if factors are covered by existing rubrics
4. Select criteria with highest discriminative value

## Output Format
Return ONLY a JSON object, with no markdown fences and no extra text:
{
  "question": "<original question verbatim>",
  "positive_rubrics": [
    {
      "description": "<detailed excellence description>",
      "title": "<abstract label>"
    }
  ],
  "negative_rubrics": [
    {
      "description": "<detailed failure description>",
      "title": "<abstract label>"
    }
  ]
}

## Examples
Positive:
{
  "description": "Anticipates and addresses potential edge cases or
exceptions to the main solution, demonstrating thorough problem
understanding",
  "title": "Edge Case Handling"
}

Negative:
{
  "description": "Conflates correlation with causation when
interpreting data or making recommendations",
  "title": "Causal Misattribution"
}

## Critical Reminders
- Each rubric must distinguish between actual provided responses
- Exclude rubrics applying equally to all responses
- Prefer empty lists over redundancy when existing rubrics are
comprehensive
- Focus on observable, objective, actionable criteria
- Quality over quantity: 2 excellent rubrics > 5 mediocre ones
- Do not use response scores; none are provided
- Generate only the most impactful, non-redundant rubrics revealing
meaningful quality differences
```

The archive user prompt presents responses without G-N-B labels:

Rubric evolution: archive user prompt

```
Question:
{query}

Historical responses from completed earlier query visits:
Historical response 1:
[RESPONSE | historical query visit {visit_id}]
{response_1}

Historical response 2:
[RESPONSE | historical query visit {visit_id}]
{response_2}

Historical response 3:
[RESPONSE | historical query visit {visit_id}]
{response_3}

...

Existing Rubrics:
Current fully evolving rubric pool:
1. [{polarity}] {title}: {description}
...

Generate only the most impactful, non-redundant rubrics revealing
meaningful quality differences.
Return JSON only:
```

F.3 PROBABILISTIC JUDGE

The probabilistic judge evaluates one response against one criterion. SERPO asks the judge to identify supporting evidence first and place the `true` or `false` token, encoding a Pass or Fail verdict, at the end. The criterion-satisfaction probability is therefore recovered from a post-reasoning verdict distribution.

Probabilistic judge: evidence-first prompt

```
You are a careful, balanced evaluator. Decide whether the LAST
assistant turn meets ONE rubric item. Evaluate the response against the
 literal requirement, without trying to maximize either true or false.

# Conversation
user: {query}

assistant: {response}

# Rubric item
{rubric_direction_prefix}
{criterion_description}

# How to grade
- Mark criteria_met=true when the response explicitly supplies the
substance required by the rubric, even if it uses equivalent wording.
- Mark false when the required substance is absent, contradicted,
merely hinted at, or too vague to establish.
- Do not invent a missing detail or invent a reason to reject explicit
evidence.
- If the rubric describes an undesirable behavior, true means the
response actually exhibits that behavior.
- The explanation must quote or precisely identify the response
evidence supporting the decision.

# Output
Return ONLY a json object in markdown. The fields MUST appear in this
exact order: first "explanation" (string), then "criteria_met" (boolean
). No other fields and no other text.
```

The direction prefix is inserted before the criterion text:

Probabilistic judge: criterion direction prefixes

```
Positive criterion:
[DESIRED REQUIREMENT: criteria_met=true means the response satisfies
this requirement.]

Negative criterion:
[NEGATIVE RUBRIC: criteria_met=true means the response actually
exhibits the forbidden behavior. If the wording says must not, should
not, avoid, or does not, judge whether the underlying prohibited
behavior occurs; do not mark true merely because the response obeys the
 prohibition.]
```

If JSON parsing, final verdict-token lookup, or true/false logprob recovery fails, the cell is marked missing rather than converted to a hard score.

G FORMAL TTRL BACKGROUND

At global adaptation step t, the policy processes a minibatch $\mathcal{X}_t \subset \mathcal{D}_{\text{adapt}}$. For each prompt $x \in \mathcal{X}_t$, the current policy π_{θ_t} samples a group of G responses,

$$\mathcal{O}_t(x) = \{o_{t,i}\}_{i=1}^{G}, \qquad o_{t,i} \sim \pi_{\theta_t}(\cdot \mid x).$$

Here $\mathcal{O}_t(x)$ is the sampled rollout group, and $o_{t,i}$ is its i-th response. Because the true quality $r^\star(x, o)$ is unavailable, the method optimizes a pseudo-reward $\hat{r}_t(x, o)$ constructed from permitted information:

$$\max_{\theta} \; \mathbb{E}_{x \sim \mathcal{D}_{\text{adapt}},\, o \sim \pi_\theta(\cdot|x)} \left[\hat{r}_t(x, o)\right]. \tag{19}$$

The reward rule may be refreshed between updates using accumulated test-time evidence, but is fixed within each actor update.

Answer-level TTRL instantiates $\hat{r}_t$ by mapping each response to a comparable answer. Let Ext be an answer extractor and Sel a pseudo-label selector, such as majority voting over extracted answers (Wang et al., 2023; Zuo et al., 2025; Wu et al., 2026). Given $a_i = \mathrm{Ext}(o_{t,i})$, it selects $\hat{a} = \mathrm{Sel}(\{a_i\}_{i=1}^{G})$ and induces

$$\hat{r}_{\text{ans}}(x, o_{t,i}) := \mathbb{I}[\mathrm{Ext}(o_{t,i}) \equiv \hat{a}], \tag{20}$$

where $\equiv$ is task-specific equivalence. This is well defined for categorical or normalized answers, but open-ended responses can share a coarse answer while differing in factuality, completeness, or safety. We retain Eq. (19) while replacing answer agreement with criterion verification.