CARE: Contrastive Anchor-based Rubric Evolution for Large Language Model Post-Training

Siyuan Li[1,*] **Xinxin Song**[1,*] **Chen Ruinian**[2] **Jingjing Fan**[1]
Tingxiong Xiao[1] **Yangen Hu**[2] **Ke Zeng**[2] **Jinli Suo**[1,†]
[1]Department of Automation, Tsinghua University
[2]Meituan
[†]jlsuo@tsinghua.edu.cn

Abstract

Rubric-based reinforcement learning decomposes open-ended instructions into prompt-specific, flexible rubrics, making it better suited than reinforcement learning with verifiable rewards for post-training LLMs on open-ended tasks. However, static rubrics are inevitably hacked as the policy evolves, and existing dynamic approaches introduce new problems: undirected rubric extraction, unreliable hack detection, and unbounded rubric proliferation. We propose **CARE** (**C**ontrastive **A**nchor-based **R**ubric **E**volution), which grounds every rubric evolution step in a high-quality anchor response generated by a frontier model conditioned on the prompt and its rubrics. At each training step, CARE contrasts the highest-scoring rollout against the anchor, enabling two complementary mechanisms: an Adaptive branch that reactively repairs reward misspecification; and a Chase branch that proactively converts frontier-level quality gaps into sharper rubrics. Together, the two branches **maintain discriminative accuracy in the high-reward region**—the precise region where reward over-optimization mostly originates. Experiments on WildChecklist-9K with Qwen2.5-7B-Base and Qwen2.5-7B-Instruct show that CARE achieves state-of-the-art performance on Arena-Hard-2.0, InfoBench, and FollowBench, and is the **only** method whose win rate against GPT-4.1 anchor responses shows sustained improvement throughout 300 training steps; additional results on Llama-3.1-8B-Instruct and Qwen3-8B further indicate that CARE generalizes across model families.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has demonstrated remarkable success in domains with objectively checkable answers such as mathematics (Lambert et al., 2025; Guo et al., 2025; Shao et al., 2024, 2025; Luo et al., 2025b) and other programmatically verifiable tasks (Luo et al., 2025a), but open-ended tasks like instruction following and long-form generation still lack reliable reward signals. A natural alternative is to rely on learned reward models (Stiennon et al., 2020; Wu et al., 2023) or LLM-as-Judge systems (Liu et al., 2023; Panickssery et al., 2024), yet both remain inadequate for this setting: scalar RMs are vulnerable to reward misspecification and reward hacking (Gao et al., 2023; Coste et al., 2024), while generic judges evaluate against broad rubrics such as "helpfulness" or "coherence" and therefore miss prompt-specific constraints (Zheng et al., 2023).

Rubric-based RL is a promising direction because it decomposes each prompt into atomic, prompt-specific, and flexible rubrics, yielding more interpretable and fine-grained reward signals for open-ended tasks (Viswanathan et al., 2025; Peng et al., 2025; Gunjal et al., 2026; He et al., 2026; Huang et al., 2025; Liu et al., 2026; Zhang et al., 2026). However, existing rubric-based RL methods predominantly rely on static, offline-generated rubrics that remain fixed throughout training. As the policy and response distribution shift during optimization, such static rubrics are inevitably hacked (Viswanathan et al., 2025; Huang et al., 2025) and cannot adapt to emergent behaviors (Rezaei et al., 2025; Shao et al., 2026).

Recent work explores dynamic rubric generation to address this problem (Rezaei et al., 2025; Shao et al., 2026), but existing methods still lack a reliable quality anchor. Without a high-quality anchor to ground the comparison, three problems arise: (1) *Undirected extraction*, where rubric generation compares arbitrary rollout pairs and captures incidental behavioral differences rather than genuine quality gaps; (2) *Blind hack detection*, where the judge must rely on parametric knowledge rather than a concrete non-hacked anchor, making novel exploits easy to miss and legitimate responses easy

[*]Equal Contribution.
[†]Corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 45334–45355
October 24-29, 2026 ©2026 Association for Computational Linguistics

to misclassify; and (3) *Unbounded rubric proliferation*, where flawed rubrics cannot be cleanly rewritten and new rubrics are only appended, gradually degrading verifier accuracy and introducing inter-rubric conflicts.

We propose **CARE** (**C**ontrastive **A**nchor-based **R**ubric **E**volution), which addresses all three limitations by introducing high-quality anchor responses—generated by a frontier model conditioned on the prompt and its rubrics—as *anchor points*. Inspired by Scaling Laws for Reward Model Overoptimization (Gao et al., 2023) and Chasing the Tail (Zhang et al., 2026), CARE is built around a simple principle: rubric evolution should dynamically maintain discriminative accuracy in the high-reward region throughout training, because this is where reward hacking is most likely to occur. CARE therefore focuses at each training step on the highest-scoring rollout, which is both the response most likely to exhibit reward hacking and the most informative sample for diagnosing loss of discriminative accuracy in this moving region, and contrasts it against the anchor. This design enables two complementary mechanisms: an **Adaptive branch** that detects hacking by comparing the top rollout against the anchor, rewrites or augments the exploited rubric with a `[Resist]` constraint, and enforces it via a *veto reward* that zeroes the reward of any response violating a resist rubric; and a **Chase branch** that, when no hacking is detected, extracts the most substantive quality gap between the top rollout and the anchor to proactively sharpen discrimination in the high-reward region. By grounding every rubric evolution step in a reliable anchor, CARE transforms rubric evolution from blind heuristic comparison into principled, evidence-based refinement that is simultaneously directed toward genuine quality, capable of detecting hacking with concrete reference support, and precise enough to rewrite flawed rubrics rather than blindly accumulating new ones.

Our main contributions are as follows:

- **CARE algorithm.** We propose a dynamic rubric evolution framework that maintains discriminative accuracy in the high-reward region throughout training via two complementary mechanisms, and Appendix A further formalizes why online maintenance of high-reward-region accuracy matters.
- **Anchor dataset.** We construct WildChecklist-9K-Anchor, a training corpus augmented with GPT-4.1 (OpenAI et al., 2024) generated anchor responses conditioned on per-instance rubrics, providing reliable anchor points for rubric evolution throughout training.
- **Strong empirical results.** Through extensive experiments on WildChecklist-9K with both Qwen2.5-7B-Base and Qwen2.5-7B-Instruct, CARE achieves state-of-the-art performance on Arena-Hard-2.0, FollowBench, and InfoBench, and is the *only* method whose win rate against GPT-4.1 anchor responses shows sustained improvement throughout 300 training steps, demonstrating superior long-term training stability over all baselines.
- **Mechanistic analysis.** We systematically analyze and categorize the rubrics extracted by CARE, and further use three representative cases to illustrate CARE's execution logic and practical effect, providing interpretable evidence of how online rubric evolution repairs hacking and sharpens high-reward-region discrimination.

2 Related Work

Reward models and reward hacking. Data-driven Bradley-Terry RMs (Ouyang et al., 2022; Bai et al., 2022) compress multi-dimensional response quality into a single scalar, making them structurally susceptible to reward hacking (Skalse et al., 2022): as the policy optimizes against the proxy RM, the proxy reward diverges from the gold reward in a predictable scaling pattern (Gao et al., 2023), and different RMs trained on the same data share systematic OOD errors that cannot be corrected by ensembling alone (Coste et al., 2024; Eisenstein et al., 2024), and the standard remedy of collecting fresh human feedback is costly and slow (Bai et al., 2022). LLM-as-Judge (Zheng et al., 2023) scales better but uses globally-fixed generic rubrics (Cui et al., 2024) and is prone to position bias (Wang et al., 2024), verbosity bias (Dubois et al., 2024), and sycophancy (Sharma et al., 2024)—all exploitable under optimization. Generative reward models (GRMs) such as CLoud (Ankner et al., 2024), DeepSeek-GRM (Liu et al., 2025), and RM-R1 (Chen et al., 2026) improve interpretability by generating an explicit critique or rubric before scoring, but such per-rollout generation is highly time-consuming at inference time (Zhang et al., 2026).

Rubric-based RL and rubric hacking. Rubric-based RL (Viswanathan et al., 2025; Gunjal et al.,

2026; He et al., 2026; Huang et al., 2025; Liu et al., 2026; Zhang et al., 2026) decomposes instructions into atomic, flexible rubrics, yielding finer-grained and harder-to-game rewards than scalar RMs; representative methods include RLCF (Viswanathan et al., 2025), RIFL (He et al., 2026), Rubric Anchors (Huang et al., 2025). However, all rely on static rubrics fixed before training, leaving them vulnerable to the emergent hacking patterns described in §1 (Viswanathan et al., 2025; He et al., 2026; Huang et al., 2025). Recent work attempts to automate this by evolving rubrics online (Rezaei et al., 2025; Shao et al., 2026): Online Rubrics (Rezaei et al., 2025) elicits new rubrics by comparing rollouts from the current and reference policies; RLER (Shao et al., 2026) extracts new rubrics by comparing all rollouts under the same prompt. However, neither grounds evolution in a reliable, quality-anchored response, leading to the three failure modes detailed in §1: undirected extraction, blind hack detection, and unbounded append-only proliferation.

3 Method

3.1 Problem Formulation

Rubric RL setup. Let the training dataset be $\mathcal{D} = \{(x_i, \mathcal{R}_i)\}_{i=1}^{N}$, where x_i is the prompt and $\mathcal{R}_i = \{r_i^1, r_i^2, \ldots, r_i^{K_i}\}$ is the corresponding set of offline-generated rubrics. We adopt GRPO (Shao et al., 2024) as the RL optimizer. At each training step, the policy π_θ generates G rollouts $\{y_i^g\}_{g=1}^{G}$ for each prompt x_i. The reward function scores each rollout against the rubric set via an LLM verifier:

$$r(y, \mathcal{R}) = \frac{1}{|\mathcal{R}|} \sum_{k=1}^{|\mathcal{R}|} \mathbf{1}[\text{rubric } r^k \text{ satisfied by } y]. \quad (1)$$

Reward hacking in rubric RL. In standard RLHF, reward hacking arises when the policy over-optimizes a learned reward model that serves as an imperfect proxy for true quality (Gao et al., 2023). In rubric RL, the rubric set $\mathcal{R}$ plays the role of the reward model; it serves as the proxy that the policy seeks to optimize. Reward hacking therefore occurs when a response y achieves a high rubric score but its true quality $q(y)$ is substantially lower than that of an anchor response $\hat{y}$ with a similar score:

$$r(y, \mathcal{R}) \approx r(\hat{y}, \mathcal{R}) \quad \text{but} \quad q(y) \ll q(\hat{y}). \quad (2)$$

Just as hacking a learned reward model stems from proxy misspecification, hacking in rubric RL stems from *rubric misspecification*: a rubric may contain exploitable loopholes (*flawed rubric*), or the rubric set may fail to cover newly emergent hacking behaviors (*incomplete coverage*). Just as the standard remedy in RLHF is *online RLHF*—periodically collecting fresh human feedback to update the reward model (Gao et al., 2023), though costly and slow—the natural remedy in rubric RL is to *update the rubrics online* as hacking emerges. The two failure modes above call for different targeted fixes: rewriting the flawed rubric or adding a new constraint. This observation directly motivates CARE's design.

3.2 Rubric Types and Veto Reward

We distinguish two rubric types by prefix:

- `[llm]`: general quality rubrics verified by an LLM judge (*e.g.*, logical coherence, task relevance).
- `[Resist]`: anti-hack constraints introduced by CARE's Adaptive branch (Section 3.4).

When only the baseline rubric set is used, the reward follows Eq. (1). When CARE introduces `[Resist]` rubrics, we replace it with a veto reward that explicitly enforces these hard constraints, making hacking categorically unprofitable. Similar veto-style rewards have also proved effective in prior rubric-based RL methods (He et al., 2026; Huang et al., 2025). Let $\mathcal{R}^{\text{Resist}} \subseteq \mathcal{R}$ denote the set of resist rubrics, and let $s_r(y) \in \{0, 1\}$ denote the verifier score for rubric r on response y. The veto reward can be written compactly as

$$r_{\text{veto}}(y, \mathcal{R}) = r(y, \mathcal{R}) \prod_{r \in \mathcal{R}^{\text{Resist}}} s_r(y). \quad (3)$$

Since each $s_r(y)$ is binary, the product becomes zero if any `[Resist]` rubric is violated, and equals one otherwise.

3.3 Anchor Response Generation

A key ingredient of CARE is a per-instance *anchor response* $y_i^* = M_{\text{anchor}}(x_i, \mathcal{R}_i)$, generated offline by a frontier model (GPT-4.1) conditioned on both the prompt x_i and its rubric set $\mathcal{R}_i$. We combine rubric-conditioned generation with explicit automatic filtering and human validation to obtain anchors that substantively satisfy their rubrics without exploiting them (Section 4 and Appendix B).

Crucially, the anchor serves as a *reference point*, not a quality upper bound. Its role differs across

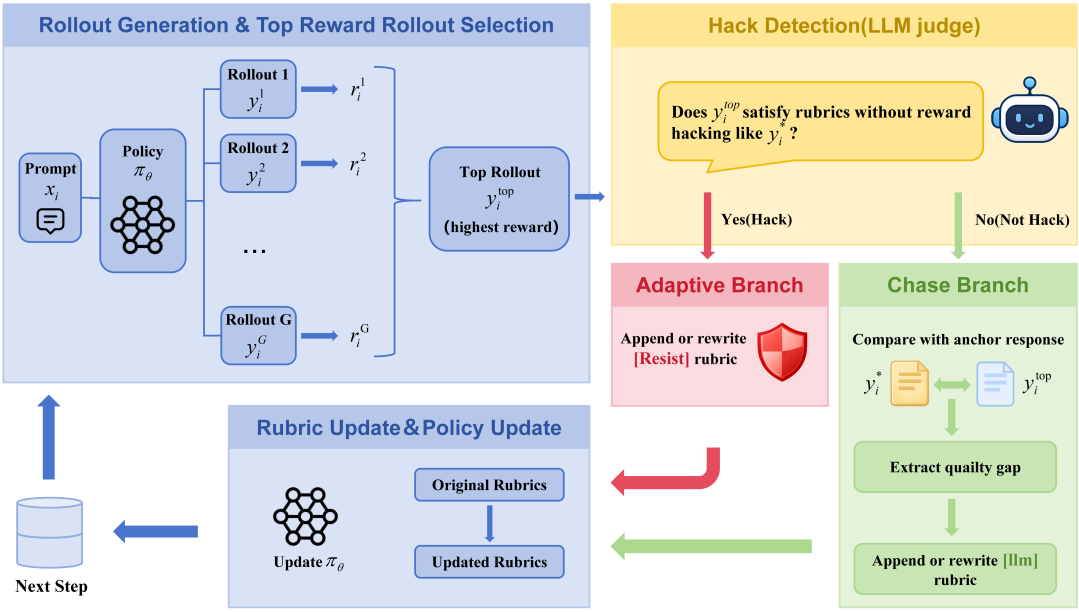

Figure 1: Overview of the CARE pipeline. At each training step, the highest-scoring rollout is contrasted against the anchor response. If hacking is detected (Adaptive branch), the exploited rubric is rewritten or a `[Resist]` constraint is added and enforced via a veto reward. Otherwise (Chase branch), the substantive gap between the top rollout and the anchor is converted into a sharper rubric to improve discrimination in the high-reward region.

the two CARE branches: (1) in hack detection, it demonstrates what a genuine, non-exploitative satisfaction of each rubric looks like, enabling evidence-based comparison; (2) in quality-gap extraction, it serves as a high-quality anchor that is contrasted against the highest-scoring rollout to expose the most meaningful substantive difference between two high-reward responses. CARE therefore assumes that the task provides reliable rubrics and permits at least one valid reference response.

3.4 CARE: Contrastive Anchor-based Rubric Evolution

Overview. Figure 1 illustrates the CARE pipeline. At each GRPO training step, CARE selects the *highest-scoring rollout* $y_i^{\text{top}} = \arg\max_g r(y_i^g, \mathcal{R}_i)$ and contrasts it against the anchor y_i^*.

The choice of the top rollout as the focal point follows a simple principle from (Gao et al., 2023; Zhang et al., 2026): what determines whether over-optimization turns into reward hacking is the rubric's discriminative accuracy in the *high-reward region*. Scaling Laws for Reward Model Overoptimization (Gao et al., 2023) shows that continued optimization eventually exploits imperfections in a proxy reward once proxy scores stop tracking true quality. Chasing the Tail (Zhang et al., 2026) further localizes this failure to the top of the reward distribution, where the key challenge is to distinguish excellent responses from merely great ones. Therefore, the central objective of CARE is to *maintain discriminative accuracy in the high-reward region throughout training*.

The top rollout is the most informative sample for this objective because it is the response currently receiving the strongest reward signal from the rubric set. Chasing the Tail (Zhang et al., 2026) improves high-reward accuracy by refining rubrics offline, but that refinement is still fixed before RL begins. As the policy evolves, the response distribution shifts in hard-to-predict ways, so offline refinement alone cannot maintain rubric accuracy in the moving high-reward region and still leaves room for reward hacking. CARE addresses this by refining rubrics online at every step, using the contrast between the current top rollout and the anchor to keep the rubric accurate in the moving high-reward region. Appendix A extends the high-reward-region argument of Chasing the Tail (Zhang et al., 2026) to CARE's dynamic rubric-update setting, formally motivating the need for online maintenance.Appendix E contains the full prompts used in the LLM verifier and CARE.

Comparing y_i^{top} with the anchor y_i^*—a reliable high-quality reference in this region—supports two complementary mechanisms for maintaining high-reward discriminability:

- *Loophole exploitation* (Adaptive branch): y_i^{top}

attains a high score through reward over-optimization rather than genuine quality improvement, showing that the current rubric is being exploited and that its misspecification must be repaired.

- *Discriminability enhancement* (Chase branch): y_i^{top} and y_i^* are both frontier-level, high-reward responses, and their substantive quality gap provides exactly the signal needed to formulate a sharper rubric. By converting this gap into a refined rubric, CARE increases resolution among high-reward responses and strengthens the high-reward region against future over-optimization.

Both branches therefore serve the same goal: *continuously refining rubric accuracy in the high-reward region*—Adaptive by repairing existing misspecification reactively, Chase by proactively sharpening discrimination before the policy learns to exploit it.

The rubrics satisfied by y_i^{top} are partitioned into $\mathcal{R}_i^{\text{sat}}$ (satisfied) and $\mathcal{R}_i^{\text{unsat}}$ (unsatisfied). Hack detection focuses exclusively on $\mathcal{R}_i^{\text{sat}}$: a rubric that is not satisfied cannot be hacked.

Step 1: Reward Hacking Detection and Repair (Adaptive Branch). The LLM judge receives $(x_i,\ \mathcal{R}_i^{\text{sat}},\ \mathcal{R}_i^{\text{unsat}},\ y_i^{\text{top}},\ y_i^*)$ and determines, for each satisfied rubric, whether y_i^{top} satisfies it in a substantively similar way to the anchor or instead reflects reward-hacking behavior relative to the anchor. If hacking is detected, the judge: (a) identifies the exploited rubric r_i^k; (b) either rewrites r_i^k to close the loophole, or appends a new `[Resist]` constraint that explicitly prohibits the observed hacking pattern. If hacking is detected, Step 2 is skipped.

Step 2: Substantive Quality-Gap Extraction (Chase Branch). Executed only when no hacking is detected. The judge identifies the most significant substantive gap between y_i^{top} and y_i^* along dimensions such as logical coherence, information depth, accuracy, or task effectiveness. The identified gap is either merged into an existing `[llm]` rubric or added as a new `[llm]` rubric. The comparison is bidirectional: the extracted dimension may capture either an advantage of the anchor or a genuine advantage of the top rollout. Thus, the anchor directs rubric discovery without imposing an imitation objective or a quality ceiling.

The full training procedure is summarized in Algorithm 1.

4 Experiments

4.1 Experimental Setup

Dataset. We build WildChecklist-9K from WildChecklist (Viswanathan et al., 2025), which synthesizes rubric checklists from real human–GPT interactions in WildChat (Zhao et al., 2024). We first generate GPT-4.1 anchors for 10K candidate instances and independently validate each anchor with Gemini 3 Pro and GPT-5.4 with high reasoning effort. The final 9K training set is selected exclusively from the pool unanimously accepted by both judges; Appendix B reports the full automatic and human validation results. Each training instance $(x_i, \mathcal{R}_i)$ consists of a user prompt and its corresponding rubric set. We construct a held-in test set from the remaining WildChecklist data for Win-rate evaluation, consisting of 300 samples.

Models and training. We train both Qwen2.5-7B-Base and Qwen2.5-7B-Instruct using GRPO, with Qwen2.5-72B-Instruct serving as the LLM verifier and CARE judge. Appendix C presents training configuration and efficiency details. We further report additional experiments on Llama-3.1-8B-Instruct and Qwen3-8B in Appendix D, demonstrating that CARE generalizes across model families.

Baselines. To demonstrate that CARE outperforms both existing static rubric-based RL methods and dynamic rubric evolution methods, we compare against the following:

- **DPO (RLCF)** (Viswanathan et al., 2025): DPO (Rafailov et al., 2023) using checklist-annotated preference data from WildChecklist-9K.
- **SFT on Anchor**: supervised fine-tuning directly on GPT-4.1 anchor responses.
- **Rubric RL**: standard rubric-based RL using offline-synthesized WildChecklist rubrics.
- **Rubric RL w/ Universal Criteria**: Rubric RL augmented with two generic anti-hack rubrics following (Viswanathan et al., 2025).
- **Online Rubrics** (Rezaei et al., 2025): a dynamic rubric evolution method that elicits new rubrics online via pairwise comparison of current- and reference-policy rollouts.

To further investigate the synergistic effects among CARE's components, we conduct the following ablation studies:

- **Chase Rubric RL**: CARE with only the Chase branch, the Adaptive branch is disabled.

Algorithm 1: CARE: Contrastive Anchor-based Rubric Evolution

Input: Policy π_θ, dataset $\mathcal{D}$, anchor responses $\{y_i^*\}$, group size G
for *step* $= 1, 2, \ldots, N$ **do**
  Sample batch $\{(x_i, \mathcal{R}_i)\}$ from $\mathcal{D}$;
  Generate G rollouts $\{y_i^g\}_{g=1}^{G}$ using π_θ;
  Compute rewards $r_i^g = r(y_i^g, \mathcal{R}_i)$;
  Select top rollout: $y_i^{\text{top}} \leftarrow \arg\max_g r_i^g$;
  Partition rubrics: $(\mathcal{R}_i^{\text{sat}}, \mathcal{R}_i^{\text{unsat}}) \leftarrow$ rubrics satisfied/unsatisfied by y_i^{top};
  if *LLM$_{judge}$ detects hacking in $\mathcal{R}_i^{sat}$ vs. y_i^** **then**
    Rewrite or append `[Resist]` rubric to $\mathcal{R}_i$; `// Adaptive branch`
  else
    Extract quality gap; merge/append `[llm]` rubric to $\mathcal{R}_i$; `// Chase branch`
  Perform GRPO update on π_θ using updated $\mathcal{R}_i$; `// Policy update`

- **Adaptive Rubric RL**: CARE with only the Adaptive branch, the Chase branch is disabled.
- **Adaptive w/o veto**: Adaptive branch with `[Resist]` rubrics enforced by weighted averaging instead of the veto reward.
- **Adaptive w/o anchor**: Adaptive branch without anchor responses–the judge relies solely on parametric knowledge to detect hacking, with the anchor ablated.

Evaluation. We evaluate on Arena-Hard-2.0 (Li et al., 2025) (Vanilla and Style-Controlled), InfoBench (Qin et al., 2024) (Easy, Hard, Overall), and FollowBench (Jiang et al., 2024) (SSR, HSR, CSL). We additionally report **Win-rate** against GPT-4.1 anchor responses (Zhang et al., 2026) , judged by Gemini 3 pro with random-order sampling to mitigate position bias.

4.2 Main Results

Consistent improvements across benchmarks. Table 1 reports results on Arena-Hard-2.0, InfoBench, and FollowBench for both Qwen2.5-7B-Base and Qwen2.5-7B-Instruct. CARE achieves state-of-the-art performance across all benchmarks under both model variants. Notably, SFT on Anchor shows strong InfoBench scores on Qwen2.5-7B-Base, surpassing most rubric-based RL methods except CARE and Adaptive Rubric RL, yet it underperforms Rubric RL on Qwen2.5-7B-Instruct and provides even slight degradation on Arena-Hard-2.0—a benchmark that requires broad instruction generalization. This contrast also demonstrates that RL generalizes better than SFT (Chu et al., 2025). CARE consistently outperforms both static rubric-based methods—Rubric RL and DPO (RLCF)—demonstrating the comprehensive advantage of CARE over static rubric approaches. Furthermore, CARE surpasses Online Rubrics, showing that guiding rubric evolution with high-quality anchor responses yields substantially better results than anchor-free pairwise comparison.

Anchor as reference rather than quality ceiling. The present CARE models do not outperform GPT-4 on every metric, which is expected given their few-billion-parameter scale, the 9K-example training set, and the absence of a cold-start stage. Nevertheless, Qwen2.5-7B-Instruct with CARE nearly matches GPT-4 on FollowBench-CSL and matches it on InfoBench-Hard, while Llama-3.1-8B-Instruct with CARE exceeds GPT-4 on InfoBench-Hard (Appendix D). In the main single-run held-in comparison, the final Qwen2.5-7B-Base checkpoint also reaches a 47% win rate against rubric-conditioned GPT-4.1 anchors, approaching parity despite receiving no rubric at inference time. Moreover, directly imitating the same anchors through SFT yields smaller and less consistent gains, including degradation on Arena-Hard-2.0. Together with Chase's bidirectional comparison, these results support the intended mechanism-level claim: reference-guided rubric evolution is more effective than treating the anchor as an imitation target, although broader superiority will also depend on policy capacity, data scale and quality, anchor and judge quality, cold start, and train–evaluation alignment.

Long-term training stability and anti-hacking. Beyond static benchmark scores, we further examine training dynamics to assess whether CARE's gains are sustained under prolonged optimization. Figure 2 shows the Win-rate curves against GPT-4.1 anchor responses throughout 300 training steps. CARE (purple) is the *only* method that shows sustained Win-rate improvement throughout training, including continued late-stage gains. Rubric RL (dashed) maintains the lowest Win-rate throughout: its Win-rate peaks around step 100 and then

gradually deteriorates, exhibiting a classic reward over-optimization pattern in which the policy exploits the static rubric proxy without genuinely improving response quality. Rubric RL w/ Universal Criteria performs more stably overall, but also begins to exhibit a declining trend after approximately step 220, confirming that static generic anti-hack constraints provide only temporary relief and cannot suppress the emergent hacking behaviors that arise in later training. The training-reward fluctuations are also partially interpretable. The left panel reports a contemporaneous proxy under evolving rubrics rather than scores under one fixed rubric set. A revised rubric affects a sample when that sample is next encountered; with 9K examples and batch size 96, each pass takes approximately 94 steps. Later passes therefore expose the policy to stricter, previously evolved criteria, which can temporarily lower proxy reward around steps 100 and 200 before the upward trend resumes. Independent checkpoint evaluations, three-seed retraining, and paired-bootstrap tests in Appendix D further show that the late-stage improvement is reproducible rather than an incidental fluctuation.

4.3 Ablation Study

All ablation experiments are conducted on Qwen2.5-7B-Base. Results are reported in Table 1 (Base rows) and Figure 2 (Win-rate curves). Together, these results reveal the contribution of each component and validate the key design choices in CARE.

(1) Overall performance ranking. Across the three benchmark suites, CARE outperforms both Adaptive Rubric RL and Chase Rubric RL. The two single-branch variants perform comparably overall, yet optimize in different directions: Chase achieves higher Arena-Hard-2.0 scores (9.3 vs. 5.3 Vanilla) while Adaptive is stronger on InfoBench (82.9 vs. 81.1 Overall), indicating that the two branches capture complementary quality dimensions. Both single-branch variants outperform Adaptive w/o veto, which in turn outperforms Rubric RL and Adaptive w/o anchor. Rubric RL outperforms Adaptive w/o anchor across two benchmarks—InfoBench (81.2 vs. 77.4 Overall), FollowBench (2.65 vs. 2.57 CSL).

(2) Necessity of the anchor. Adaptive Rubric RL consistently outperforms Adaptive w/o anchor across all three benchmark suites—Arena-Hard-2.0 (5.3 vs. 4.4 Vanilla), InfoBench (82.9 vs. 77.4 Overall), and FollowBench (2.91 vs. 2.57 CSL)—and also maintains a substantially higher Win-rate trajectory throughout training (Figure 2). This gap is consistent with our direct audit in Appendix B: frontier-judge agreement with Adaptive decisions is 95.2% with anchors but 65.3% without them, while human agreement is 99.1% and 67.1%, respectively. A validated, non-hacked anchor therefore substantially improves the reliability of hack identification.

(3) Necessity of the veto reward. Adaptive w/o veto underperforms Adaptive Rubric RL across all benchmarks and also exhibits a lower Win-rate trajectory in Figure 2. This validates that weighted averaging allows models to compensate for `[Resist]` violations by scoring higher on other rubrics, whereas the veto mechanism eliminates this loophole by making hacking categorically unprofitable. Because veto enforcement also amplifies an incorrectly specified constraint, its reliability depends on accurate Adaptive decisions; the automatic and human audits in Appendix B, together with the taxonomy and representative cases in Section 4.4, directly evaluate this premise.

(4) Complementarity of Adaptive and Chase. As shown in Figure 2, Chase Rubric RL rises rapidly in the early stages (achieving the second-highest Win-rate through step 150), indicating that proactively sharpening discrimination in the high-reward region is highly effective for immediate gains, but it subsequently peaks around step 200 and declines. This pattern is consistent with accumulated misspecification, although the aggregate curve alone cannot determine whether it originates from initial rubrics or Chase-generated rubrics. Adaptive Rubric RL and Adaptive w/o veto maintain relatively stable Win-rates throughout training but plateau at a lower ceiling, indicating that reactive repair alone improves robustness to emerging reward exploitation but does not sharpen frontier-level discrimination enough to sustain further gains. CARE combines both mechanisms and achieves the highest Win-rate as well as the only sustained improvement over the full 300-step trajectory, consistent with Appendix A's account of balancing frontier sharpening against misspecification repair.

4.4 Analysis: How CARE Works

Table 2 breaks down FollowBench HSR by constraint type (Jiang et al., 2024). CARE achieves particularly striking gains on Style and Situation—the two most implicit constraint types, where Style specifies *how* a response should be phrased and Sit-

	Arena-Hard-2.0		InfoBench			FollowBench		
	Vanilla	**Style-Ctrl.**	**Easy**	**Hard**	**Overall**	**SSR**	**HSR**	**CSL**
GPT-4	79.3	76.9	89.3	86.4	87.3	87.8	83.7	3.52
Qwen2.5-7B-Instruct	3.5	3.2	82.7	76.0	78.1	82.6	71.4	3.05
+ SFT on anchor response	1.7	2.7	85.2	82.1	83.1	80.5	72.2	3.18
+ DPO (RLCF)	4.6	3.5	84.5	84.1	84.2	83.9	74.3	3.23
+ Rubric RL	5.3	3.9	84.4	82.8	83.3	84.1	75.6	3.26
+ Rubric RL w/ Universal Criteria	6.0	4.2	85.7	84.1	84.6	83.9	76.3	3.45
+ Online Rubric	6.8	4.3	80.8	80.2	80.4	82.4	73.1	3.28
+ **CARE**	**10.8**	**12.2**	**87.4**	**86.4**	**86.7**	**85.9**	**78.4**	**3.50**
Qwen2.5-7B-Base	0.5	1.9	77.4	68.8	74.8	56.8	38.8	1.20
+ SFT on anchor response	1.3	2.6	84.0	81.4	82.2	74.4	61.7	2.73
+ DPO (RLCF)	1.1	2.3	83.2	72.2	75.6	67.7	51.9	2.10
+ Rubric RL	2.0	1.9	81.3	81.1	81.2	71.0	62.8	2.65
+ Rubric RL w/ Universal Criteria	2.4	2.1	83.0	81.6	82.0	74.2	64.9	2.94
+ Online Rubric	3.2	1.5	80.4	78.4	79.0	72.4	61.1	2.78
+ Adaptive Rubric RL	5.3	9.3	83.7	82.6	82.9	74.6	64.4	2.91
+ Chase Rubric RL	9.3	10.2	81.7	80.9	81.1	75.4	65.2	2.98
+ Adaptive w/o veto	3.4	5.1	83.0	79.8	80.8	72.7	63.4	2.80
+ Adaptive w/o anchor	4.4	5.7	78.9	76.8	77.4	68.5	59.3	2.57
+ **CARE**	**10.7**	**11.8**	**85.0**	**83.4**	**83.9**	**77.3**	**68.5**	**3.15**

Table 1: Main results after three training epochs on Arena-Hard-2.0 (Vanilla / Style-Controlled), InfoBench (Easy / Hard / Overall), and FollowBench (SSR / HSR / CSL). CARE achieves consistent gains over all baselines on both model variants.Positive results relative to the baseline are indicated in green, negative results in orange. The top-performing variant of each model is highlighted in bold.

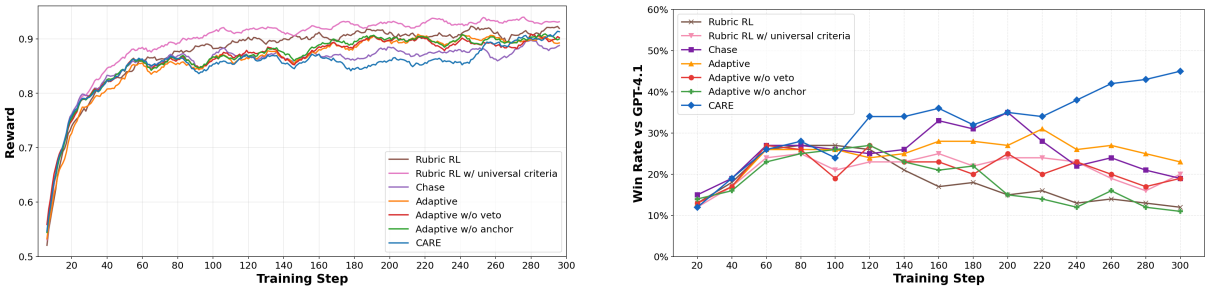

Figure 2: Left: contemporaneous training rewards under each method's evolving rubric set. Right: Win-rate against GPT-4.1 anchor responses over 300 training steps for Qwen2.5-7B-Base. CARE is the only method showing sustained improvement across the full trajectory. Revised rubrics affect a sample on its next encounter, partially explaining fluctuations near successive data passes.

uation provides only an implicit contextual background rather than a direct output specification. On Qwen2.5-7B-Base, CARE nearly *doubles* the Style score (48.0 → 94.7) and improves Situation by more than one-third (55.5 → 73.6), gains that static rubric methods largely fail to achieve. These results reveal that CARE, guided by the anchor, extracts precisely the implicit quality gaps that offline rubrics systematically underspecify and prevents superficial shortcuts from satisfying those rubrics.

CARE rubric patterns. Beyond aggregate performance, we also analyze the rubrics extracted by CARE themselves. Appendix B further organizes these CARE-evolved rubrics into a fine-grained taxonomy. Here we complement that appendix analysis with three branch-specific case studies to show how CARE converts concrete anchor comparisons into targeted rubric refinement.

(1) *Adaptive branch — A1a Word-sense ambiguity exploitation.* The user asks for a sports broadcast script, but the original rubric only asks whether the output is "a script." The model exploits this ambiguity by returning Python code. Comparing it with the broadcast-style anchor, CARE identifies the wrong sense and adds:

```
[Resist] Is the generated script a sports broadcast or announcement script, not a programming script?
```

This Adaptive update closes the loophole without penalizing legitimate broadcast responses.

(2) *Adaptive branch — A2d Language substitution.* For an Arabic prompt, the offline rubrics

	Avg (HSR)	Format	Style	Situation	Content
GPT-4	83.7	83.3	97.3	78.2	76.0
Qwen2.5-7B-Base	46.9	46.7	48.0	55.5	37.6
+ DPO (RLCF)	51.9	44.7	58.0	68.2	36.8
+ SFT on anchor response	61.7	52.7	78.0	60.0	56.0
+ Rubric RL	62.8	55.6	91.0	66.2	38.4
+ Rubric RL w/ Universal Criteria	64.9	60.3	91.2	69.7	38.4
+ Online Rubrics	61.1	45.2	87.3	70.1	41.8
+ **CARE**	**68.5**	**60.8**	**94.7**	**73.6**	**44.9**

Table 2: FollowBench HSR breakdown by constraint type (Format / Style / Situation / Content). CARE achieves the largest gains on Style and Situation. Positive results relative to the baseline are indicated in green, negative results in orange. The top-performing variant of each model is highlighted in bold.

specify content but not strict language consistency. The model returns a mixed response containing Arabic, Chinese characters, and special symbols. Since the anchor remains fully in Arabic, CARE flags this as constraint evasion and adds:

```
[Resist] Does the response strictly maintain language consistency by using ONLY the language of the prompt (Arabic), avoiding any unauthorized code-switching?
```

Again, the anchor exposes a real prompt-constraint violation and lets CARE patch the missing constraint.

(3) *Chase branch — B2a Lack of specific detail.* For a question about the effects of music on plant growth, an existing rubric broadly asks for "detailed examples and specific studies." Because this criterion is still too coarse, the model can satisfy it with a high-level discussion of possible mechanisms, even without citing any named studies or experiments. The anchor, by contrast, includes concrete studies and findings. CARE identifies this missing frontier distinction and sharpens the rubric into:

```
[llm] Does the response provide detailed examples and specific studies to support the explanation of the effects of music on plant growth, including at least two named studies or experiments?
```

This Chase update turns a broad, weakly specified rubric into a more discriminative frontier criterion.

Together, these cases show the two roles of CARE: Adaptive repairs exploited loopholes, while Chase sharpens under-specified quality criteria. In each case, the anchor makes the update concrete.

5 Conclusion

We introduce **CARE**, an anchor-guided framework for online rubric evolution in rubric-based reinforcement learning. By comparing the highest-scoring rollout with a high-quality anchor at each step, CARE repairs exploited misspecification and sharpens frontier discrimination. On WildChecklist-9K, CARE achieves substantial performance gains across model variants. These results show that maintaining rubric accuracy online is an effective way to improve both robustness to reward hacking and final response quality in open-ended post-training.

Limitations

We discuss the limitations of our study in this section, focusing on two primary aspects. (1) CARE currently activates rubric evolution at every training step. A natural next direction is to make this process selective rather than unconditional, for example by introducing signals that trigger CARE only periodically or only on samples with a high likelihood of hacking. Such gating could focus rubric evolution on the most informative failures, improve update precision, and reduce training cost. (2) A second open problem is how to evaluate the quality of generated rubrics themselves. In the current work, following the intuition of Chasing the Tail (Zhang et al., 2026), rubric quality is improved indirectly by maintaining discriminative accuracy in the high-reward region. However, a principled rubric-quality evaluation framework remains missing. Recent work such as RIFT (Qi et al., 2026) begins to characterize rubric failure modes and automated diagnostics, suggesting a promising direction for combining online rubric evolution with explicit rubric-quality assessment.

Ethical Considerations

(1) **Licenses.** Our experiments use open models and public datasets under their original licenses. The Qwen series models are released under the Apache License 2.0, while Llama-3.1 is released under the Llama 3.1 Community License Agreement. The training data is based on WildChecklist, which is distributed under the MIT License. For evaluation, FollowBench and Arena-Hard-Auto are released under the Apache License 2.0, while InfoBench is released under the MIT License. (2) **Potential harms.** CARE is designed to improve online rubric evolution for open-ended post-training, but the same mechanism could also be misused to optimize models toward undesirable objectives if the rubrics themselves encode harmful, manipula-

tive, or deceptive goals. In addition, CARE relies on frontier-model-generated anchor responses and LLM judges, which may inherit biases, cultural assumptions, or unsafe preferences from the underlying models. If such signals are treated as universally correct, these biases may be propagated into rubric updates and may over-reward stylistic conformity or particular response preferences. We therefore view CARE as a research method rather than a deployment-ready safety guarantee, and recommend human review, task restrictions, and explicit risk auditing before applying it in high-stakes settings.(3) **AI assistance.** We use GPT-5.4 and Claude Code for writing polish and coding assistance.

Acknowledgements

This work was jointly supported by the National Key R&D Program of China (Grant No. 2024YFF0505703) and the Beijing Municipal Natural Science Foundation (Grant No. L257009).

References

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. 2024. Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

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. 2026. RM-r1: Reward modeling as reasoning. In *The Fourteenth International Conference on Learning Representations*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. 2024. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations*, volume 2024, pages 50905–50931.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. ULTRAFEEDBACK: Boosting language models with scaled AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 9722–9744. PMLR.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alexander Nicholas D'Amour, Krishnamurthy Dj Dvijotham, Adam Fisch, Katherine A Heller, Stephen Robert Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. 2024. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *First Conference on Language Modeling*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. 2026. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Kumar Gonugondla, Hunter Lang, Yue Yu, and 6 others. 2026. AdvancedIF: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18003–18022, San Diego, California, United States. Association for Computational Linguistics.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, and 2 others. 2025. Reinforcement learning with rubric anchors. *Preprint*, arXiv:2508.12790.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin

Jiang, Qun Liu, and Wei Wang. 2024. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*.

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

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. 2026. OpenRubrics: Towards scalable synthetic rubric generation for reward modeling and LLM alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17417–17437, San Diego, California, United States. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025a. Deepcoder: A fully open-source 14b coder at o3-mini level. Notion Blog.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Arjun Panickssery, Samuel R Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. VerIF: Verification engineering for reinforcement learning in instruction following. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30324–30339, Suzhou, China. Association for Computational Linguistics.

Zhengyang Qi, Charles Dickens, Derek Pham, Amanda Dsouza, Armin Parchami, Frederic Sala, and Paroma Varma. 2026. Rift: A rubric failure mode taxonomy and automated diagnostics. *arXiv preprint arXiv:2604.01375*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048.

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

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

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, and 2 others. 2026. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *Preprint*, arXiv:2511.19399.

Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. 2025. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv:2511.22570*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Mrinank Sharma, Meg Tong, Tomek Korbak, David Duvenaud, Amanda Askell, Sam Bowman, Esin DURMUS, Zac Hatfield-Dodds, Scott Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. 2024. Towards understanding sycophancy in language models. In *International Conference on Learning Representations*, volume 2024, pages 110–144.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. 2025. Checklists are better than reward models for aligning language models. *Preprint*, arXiv:2507.18624.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36:59008–59033.

Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. 2026. Chasing the tail: Effective rubric-based reward modeling for large language model post-training. In *The Fourteenth International Conference on Learning Representations*.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Why Online Maintenance of High-Reward-Region Accuracy Matters

We extend the static high-reward-region argument in Chasing the Tail (Zhang et al., 2026) from a fixed proxy reward to the dynamic setting relevant to CARE, where the rubric reward itself is updated during training.

Setup. Consider the idealized per-prompt KL-regularized update (Ouyang et al., 2022; Bai et al., 2022)

$$\pi_{t+1}(\cdot \mid x) = \arg\max_{\pi} \left\{ \eta_t \, \mathbb{E}_{y\sim\pi(\cdot|x)}[r_t(x,y)] - \mathbb{D}_{\mathrm{KL}}[\pi(\cdot \mid x) \,\|\, \pi_t(\cdot \mid x)] \right\}, \tag{4}$$

whose solution is (Rafailov et al., 2023)

$$\pi_{t+1}(y \mid x) = \frac{\pi_t(y \mid x)\exp(\eta_t r_t(x,y))}{Z_t(x)}. \tag{5}$$

Here r_t is the step-t rubric reward, $r^\star$ is the latent gold reward, and $\Delta_t(x,y) = r_t(x,y) - r^\star(x,y)$ is the stepwise misspecification. Let

$$H_t(x) = \{y : r_t(x,y) \geq \tau_t(x)\} \tag{6}$$

denote the current high-reward region under a threshold $\tau_t(x)$.

Proposition. For any prompt x and horizon T, unrolling the update gives

$$\pi_T(y \mid x) = \frac{\pi_0(y \mid x)\exp\left(\sum_{t=0}^{T-1} \eta_t r_t(x,y)\right)}{Z_{0:T}(x)}. \tag{7}$$

Therefore, for any two responses y^+, y^-,

$$\log\frac{\pi_T(y^+ \mid x)}{\pi_T(y^- \mid x)} = \log\frac{\pi_0(y^+ \mid x)}{\pi_0(y^- \mid x)} + \sum_{t=0}^{T-1} \eta_t\big(r^\star(x,y^+) - r^\star(x,y^-)\big) \tag{8}$$

$$+ \sum_{t=0}^{T-1} \eta_t\big(\Delta_t(x,y^+) - \Delta_t(x,y^-)\big). \tag{9}$$

Hence the final ranking between frontier responses is governed by the competition between the cumulative true-quality gap and the cumulative misspecification gap. In particular, if $r^\star(x,y^+) > r^\star(x,y^-)$ but

$$\sum_{t=0}^{T-1} \eta_t\big(\Delta_t(x,y^-) - \Delta_t(x,y^+)\big) > \log\frac{\pi_0(y^+ \mid x)}{\pi_0(y^- \mid x)} + \sum_{t=0}^{T-1} \eta_t\big(r^\star(x,y^+) - r^\star(x,y^-)\big), \tag{10}$$

then $\pi_T(y^- \mid x) > \pi_T(y^+ \mid x)$. Thus persistent misspecification on responses that repeatedly enter the high-reward region can overturn the true ordering even when low-reward-region accuracy remains acceptable.

Proof. The closed form for π_T follows by recursively substituting the one-step Gibbs update. Substituting $r_t = r^\star + \Delta_t$ into the exponent and subtracting the expression for y^- from that for y^+ yields Eq. (9). This decomposition separates the cumulative signal coming from true quality from the cumulative distortion caused by proxy misspecification. The ranking flips exactly when the distortion term outweighs the sum of the initial log-odds and the cumulative true-quality advantage, which is Eq. (10). Moreover, responses outside $H_t(x)$ receive smaller multiplicative weights $\exp(\eta_t r_t)$ at step t, so their effect on the final log-odds is exponentially downweighted relative to responses that repeatedly appear in the evolving high-reward region. Therefore, under continued optimization, the dominant source of error is misspecification on the high-reward region that the policy is currently moving toward.

Corollary. Suppose that for every step t and every pair $y^+, y^- \in H_t(x)$ with $r^\star(x, y^+) \geq r^\star(x, y^-)$, the true frontier gap and stepwise distortion satisfy

$$r^\star(x, y^+) - r^\star(x, y^-) \geq \gamma_t(x), \qquad \left|\Delta_t(x, y^+) - \Delta_t(x, y^-)\right| \leq \varepsilon_t(x). \tag{11}$$

Then any such frontier pair preserves the correct ordering up to step T whenever

$$\log \frac{\pi_0(y^+ \mid x)}{\pi_0(y^- \mid x)} + \sum_{t=0}^{T-1} \eta_t \gamma_t(x) > \sum_{t=0}^{T-1} \eta_t \varepsilon_t(x). \tag{12}$$

Conversely, if a static rubric induces a persistent frontier distortion whose cumulative magnitude eventually exceeds the left-hand side of Eq. (12), then a wrong frontier ordering becomes inevitable under continued optimization.

Connection to CARE. This corollary formalizes CARE's key idea: it is not enough to improve high-reward-region accuracy once before RL begins; one must maintain it throughout training. The corollary itself does not prove that any particular rubric-update rule will always reduce frontier distortion or enlarge frontier separation. Rather, it identifies the direction of improvement required to preserve correct ordering in the evolving high-reward region: successful updates should reduce the frontier distortion term and/or enlarge the frontier signal term. Under this interpretation, CARE's two branches are designed to target these two quantities. When the Adaptive branch correctly identifies an exploited rubric and repairs it, it is intended to reduce the local frontier distortion represented by ε_t. When the Chase branch extracts a genuine substantive quality gap and turns it into a sharper rubric, it is intended to enlarge the frontier separation represented by γ_t. Thus, the theory–method link here is mechanistic rather than an end-to-end guarantee: CARE is designed so that successful repair decreases frontier misspecification, while successful gap extraction increases frontier discriminability, helping keep the cumulative frontier signal larger than the cumulative frontier misspecification over training.

B LLM-Based Quantitative Analysis of CARE-Extracted Rubrics

Our Appendix-B quantitative analysis first validates the anchor-construction pipeline and Adaptive decisions, then presents a fine-grained taxonomy of CARE-evolved rubrics. Table 8 lists the major classes and subtypes for Adaptive-branch `[Resist]` rubrics, while Table 9 does the same for Chase-branch `[llm]` rubrics.

B.1 Anchor Quality Validation

Anchor validation is integrated into data construction. We generate anchors for 10K candidate instances and ask Gemini 3 Pro and GPT-5.4 with high reasoning effort to independently assess rubric satisfaction, factual hallucination, and unsafe or non-compliant content. As shown in Table 3, both judges accept 93.8% of the candidates, disagree on 6.0%, and both reject 0.2%. We construct the final 9K set exclusively from the unanimously accepted pool. A subsequent stratified human audit of 200 anchors from the retained set finds that 199 pass the same criteria (99.5%), independently confirming the quality of the filtered data.

Anchor-validation outcome	Result
Both frontier judges accept	93.8%
Frontier judges disagree	6.0%
Both frontier judges reject	0.2%
Human audit after filtering	199/200 (99.5%)

Table 3: Automatic validation of the 10K candidate-anchor pool and stratified human validation of the retained 9K training set.

B.2 Adaptive-Decision Reliability

Because veto enforcement amplifies an incorrectly specified `[Resist]` rubric, we directly audit whether Adaptive correctly detects hacking and introduces an appropriate constraint. Gemini 3 Pro and GPT-5.4 with high reasoning effort audit all Adaptive decisions, and human annotators audit 200 stratified cases; we repeat the same procedure after removing the anchor from the detection context. Table 4 shows that the anchor raises frontier-judge agreement from 65.3% to 95.2% and human judgment-level agreement from 67.1% to 99.1%. This audit, the complete taxonomy below, and the representative end-to-end cases in Section 4.4 provide complementary quantitative and qualitative evidence for the reliability and specificity of the added constraints.

Audit outcome	With anchor	Without anchor
Both judges agree with detection	95.2%	65.3%
Judges disagree	0.5%	17.1%
Both judges reject detection	4.3%	17.6%
Human judgment-level agreement	99.1%	67.1%

Table 4: Automatic and human audits of Adaptive decisions with and without anchor responses.

B.3 Adaptive Branch Taxonomy

Table 8 presents the fine-grained taxonomy for Adaptive-branch `[Resist]` rubrics.

B.4 Chase Branch Taxonomy

Table 9 presents the corresponding taxonomy for Chase-branch `[llm]` rubrics.

C Training Configuration and Efficiency

C.1 Hyperparameter Settings

We implement CARE on Verl v0.4.0 and conduct GRPO training on 16 NVIDIA A100 GPUs. The key hyperparameters used in GRPO training are summarized in Table 10.

C.2 Training Time Comparison

We further compare the per-step training time of Rubric RL and CARE on Qwen2.5-7B-Base. As shown in Table 11, CARE increases the per-step time from 318.2s to 448.9s, corresponding to 1.411× total time, or a 41.1% increase, relative to Rubric RL.

CARE adds a one-time offline anchor-generation cost and a repeated online comparison cost. Let N be the dataset size, E the number of training epochs, B the number of prompts per step, G the rollouts per prompt, C_A the cost of generating one anchor, and C_C the cost of one CARE comparison. The two additional costs scale approximately as

$$T_{\text{offline}} \approx NC_A, \qquad \Delta T_{\text{online}} \approx ENC_C. \quad (13)$$

Anchors are generated once and cached. Online, CARE selects only the highest-scoring rollout for each prompt and compares it with the anchor, adding B rather than BG comparisons per step. Thus, increasing G does not increase the CARE-specific comparison count. If $T_{\text{RL}}(B, G)$ is the baseline RL wall-clock time per step, the relative online overhead is approximately

$$\rho_{\text{online}} \approx \frac{BC_C}{T_{\text{RL}}(B, G)}. \quad (14)$$

The overhead becomes relatively small when rollout generation dominates comparison time. A

more expensive anchor generator affects only C_A, whereas a more expensive CARE judge affects the repeated cost C_C. For fixed E, both total costs scale linearly with N. As discussed in Limitations, more selective CARE activation is a promising direction for further reducing the online component.

D Additional Results and Statistical Analysis

D.1 Cross-Model Results

We further evaluate CARE on two additional backbone models, Llama-3.1-8B-Instruct and Qwen3-8B. Following the same evaluation protocol as Table 1, we compare Rubric RL and CARE across Arena-Hard-2.0, InfoBench, and FollowBench in Table 12.

D.2 Independent Checkpoint Trajectories

To separate CARE's progress from evaluation against its GPT-4.1 training anchors, we evaluate Qwen2.5-7B-Base checkpoints against Qwen3-8B-Thinking, which is not used in anchor construction or training, and on the external InfoBench benchmark. Table 5 reports step 0 as the untrained policy baseline and steps 100/200/300 as CARE checkpoints. Both independent measures improve throughout training.

Independent evaluation	Step 0	Step 100	Step 200	Step 300
Win rate vs. Qwen3-8B-Thinking (%)	16.0	30.0	52.0	69.0
InfoBench Overall	74.8	80.9	82.1	84.0

Table 5: Independent checkpoint trajectories for Qwen2.5-7B-Base. These supplemental evaluations do not alter the three-epoch results in Table 1.

D.3 Training-Seed Robustness

We retrain CARE with three independent seeds using the same hyperparameters as Appendix C and evaluate all checkpoints on the same fixed 300 held-out prompts. Table 6 shows that every run improves by 21 percentage points from step 100 to step 300 and reaches a 45.0% final win rate; cross-seed variation is limited to 0.6 points at step 200. These retraining results are separate from the 47% final win rate of the main single-run experiment.

Seed	Step 100	Step 200	Step 300	Δ(300–100)
1	24.0%	35.0%	45.0%	+21.0 pp
2	24.0%	36.0%	45.0%	+21.0 pp
3	24.0%	35.0%	45.0%	+21.0 pp
Mean ± SD	$24.0 \pm 0.0\%$	$35.3 \pm 0.6\%$	$45.0 \pm 0.0\%$	$+21.0 \pm 0.0$ pp

Table 6: Win rate against GPT-4.1 anchors across three independent CARE training seeds.

D.4 Paired-Bootstrap Analysis

We also quantify sampling uncertainty from the finite evaluation set. For each comparison, we resample the same 300 prompts with replacement while preserving paired system outcomes, use 10,000 bootstrap replicates, and report two-sided percentile 95% confidence intervals for the win-rate difference. All intervals in Table 7 exclude zero, supporting both CARE's late-stage improvement and its final advantage over Adaptive.

Paired comparison	Difference	95% CI
$\text{CARE}_{300} - \text{CARE}_{100}$	+21.0 pp	$[+14.0, +28.0]$ pp
$\text{CARE}_{300} - \text{CARE}_{200}$	+10.0 pp	$[+5.0, +17.0]$ pp
$\text{CARE}_{300} - \text{Adaptive}_{300}$	+23.0 pp	$[+15.0, +29.0]$ pp

Table 7: Paired-bootstrap confidence intervals from 10,000 resamples of the same 300 evaluation prompts.

E Prompts Used in CARE

We reproduce the two verbatim prompt templates at the end of the appendix. Figure 3 shows the reward-scoring prompt used for rubric-based response evaluation, and Figure 4 shows the shared CARE rubric-evolution prompt used by the Adaptive and Chase branches. Placeholder variables are kept exactly as they appear in the implementation.

E.1 Prompt for Reward Scoring

Figure 3 reproduces the reward-scoring template verbatim.

E.2 Prompt for CARE Rubric Evolution

Figure 4 reproduces the shared rubric-evolution prompt. Step 1 detects and patches reward hacking, while Step 2 extracts the largest substantive quality gap when no hacking is found.

Major class	Subtype	Definition	Typical pattern	Typical example
Semantic ambiguity exploitation	A1a: Word-sense ambiguity	Chooses an unintended sense of a keyword to satisfy the rubric while missing the prompt intent.	avoid interpreting X as Y	"script" is treated as programming code rather than a sports broadcast script
	A1b: Scope ambiguity	Expands a concept beyond its intended boundary and therefore misclassifies the target content.	restrict X to the domain of Y	"computers" is broadened to include software or digital assistants
	A1c: Referent ambiguity	Resolves an unclear reference to the wrong entity and therefore executes the wrong requirement.	ensure X refers to [specific entity]	"the character" is interpreted as the wrong role
Content substitution hacking	A2a: Fabricated-fact substitution	Replaces required content with fabricated or incorrect facts that still appear superficially plausible.	avoid substituting correct X with fabricated/incorrect X	a fabricated Sudarsky classification replaces the correct one
	A2b: Topic substitution	Preserves the surface structure but swaps in a different core topic from the one requested.	avoid replacing [topic A] with [topic B]	a technical document is produced instead of a sports broadcast
	A2c: Entity substitution	Replaces the required person or entity with an unspecified alternative.	avoid introducing characters/entities not specified	side characters are listed in place of the requested main ones
	A2d: Language substitution	Uses the wrong language to satisfy content rubrics.	use ONLY [required language]; avoid code-switching	an Arabic prompt is answered in Chinese
Format bypass & task evasion	A3a: Placeholder injection	Uses template placeholders instead of the actual requested content.	avoid placeholder/filler content	the output is `{raw_message}` rather than a real message
	A3b: Task simplification	Substitutes an easier task for the harder one requested by the prompt.	avoid replacing [hard task] with [easier variant]	it gives an overview instead of a detailed analysis
	A3c: Formatting abuse	Uses formatting to pad the response and hide the lack of substantive content.	avoid using [format] to pad content	excessive headings and bullet lists replace actual substance
	A3d: Incomplete execution	Completes only part of the requested procedure or task.	must complete all steps; avoid incomplete	it provides only a code skeleton without implementation
	A3e: Context reframing	Wraps the answer in an unintended fictional or alternative frame.	avoid introducing fictional contexts not specified	a direct answer is rewritten as a VR-game dialogue
Content injection & contamination	A4a: Off-topic garbage injection	Injects irrelevant material that is unrelated to the prompt but helps satisfy a broad rubric.	avoid irrelevant/off-topic content; avoid garbage injection	a Tesla answer is padded with crypto-scam language
	A4b: Self-meta commentary	Appends commentary that evaluates the response itself rather than answering the prompt.	avoid self-commentary/self-scoring	it ends with "The assistant has demonstrated excellent skills..."
	A4c: Repetitive padding	Inflates length or coverage by repeating existing content.	avoid repetition; avoid padding	the same paragraph is repeated multiple times
	A4d: Cross-domain contamination	Mixes in material from another domain, culture, or language that does not belong in the target response.	avoid mixing content from [other domain]	entertainment content appears inside a serious policy document
	A4e: Unrequested feature injection	Adds features or modules that the user never asked for.	avoid adding features not explicitly requested	a Python function includes an unrequested logging module
Constraint violation hacking	A5a: Explicit prohibition violation	Violates a clearly prohibited requirement while still being scored as acceptable.	[explicitly stated constraint] must be respected	the answer cites GDPR even though the prompt forbids it
	A5b: Tone/style violation	Breaks an explicit tone, style, or register requirement in the prompt.	maintain [required tone/style]; avoid [wrong register]	a formal document is written in colloquial language
	A5c: Context violation	Breaks an explicit scenario, role, or setting constraint.	must remain within [specified context]	out-of-scene roles are introduced in a role-play task
	A5d: Output-format violation	Fails to follow an explicitly required output format.	must use [required format] only	the prompt requests JSON but the answer returns plain text
Boundary behavior exploitation	A6a: Minimal compliance	Satisfies only the weakest literal version of the rubric without substantive fulfillment.	provide [substantive X], not merely [minimal token]	a rubric requiring a character name is satisfied by mentioning it only once
	A6b: Superficial compliance	Includes superficial markers of compliance without satisfying the underlying intent.	genuinely satisfy X, not superficially	it contains humor keywords but is not actually humorous
	A6c: Selective satisfaction	Covers only the easiest parts of a multi-part requirement.	must address all [sub-requirements]	it lists only the easy requested items and skips the hard ones

Table 8: Fine-grained taxonomy for Adaptive-branch `[Resist]` rubrics.

Major class	Subtype	Definition	Typical pattern	Typical example
Content completeness	B1a: Missing key elements	Omits a core element that the prompt explicitly requires.	include [specific required element]	a story omits the scene where the king explains the process
	B1b: Incomplete procedure	Skips required steps in an instructional or procedural response.	include all steps [for X]; cover [step N]	a tutorial skips the configuration step
	B1c: Insufficient coverage	Covers the task only partially and leaves out important subtopics.	cover all [required sub-topics]; include [category X]	educational content omits one age group
	B1d: Omitted constraints	Fails to satisfy one of the explicitly requested constraints.	address [specific constraint] explicitly	the prompt asks for five categories but the response gives only three
Information depth	B2a: Lack of specific detail	Provides a high-level answer without enough concrete details or named evidence.	provide specific/detailed [X]; include at least [N] specific examples	a discussion of music and plant growth lacks named studies
	B2b: Lack of technical depth	Misses the technical detail required for a specialized task.	include technical details of [X]; provide [specific technical aspect]	a physics-engine tutorial omits suspension parameters
	B2c: Lack of contextual background	Omits the historical, cultural, or situational context needed for a full explanation.	provide context for [X]; include historical/cultural background	a term is explained without its historical evolution
	B2d: Insufficient support or rationale	Gives a recommendation or conclusion without enough evidence or justification.	support [claim] with [evidence/examples]; provide rationale for	it recommends a method without explaining why
	B2e: Limited extension depth	Covers the core answer but omits useful advanced considerations or edge cases.	include additional considerations such as [X]; explore [advanced aspect]	a correct basic method ignores boundary cases
Narrative & expression quality	B3a: Weak sensory/scene description	Lacks vivid scene-setting or sensory detail.	provide vivid/sensory details of [setting/experience]	a restaurant review omits taste and atmosphere details
	B3b: Limited emotional depth	Underdevelops emotions or internal reflections.	provide emotional depth/internal reflections	a dialogue scene shows flat emotional change
	B3c: Weak immersion	Fails to create an engaging or immersive overall narrative.	create an immersive/engaging narrative	a story has the right events but little immersion
	B3d: Unnatural flow	Sounds stiff or transitions awkwardly between parts of the response.	maintain natural flow; ensure smooth transitions	a character speaks in an unnaturally formal way
	B3e: Inconsistent style or perspective	Breaks internal consistency of tone, style, or narrative point of view.	maintain consistent [tone/style/perspective]	a story switches between first and third person
	B3f: Weak humor or creativity	Meets the topic requirement but lacks originality, humor, or creativity.	provide a genuinely humorous/creative [X]	the joke is on-topic but not funny
Logic & structure	B4a: Weak reasoning	The argument contains leaps, contradictions, or unsupported conclusions.	maintain logical coherence; provide clear reasoning for [conclusion]	the claim and evidence in a paper are poorly connected
	B4b: Disorganized structure	The response lacks a clear and logical organization.	organize [content] in a clear/logical structure	tutorial steps appear in the wrong order
	B4c: Poor paragraph or section transitions	Adjacent parts are weakly connected and transition poorly.	ensure coherent transitions between [sections/paragraphs]	the paragraphs are individually fine but poorly linked
	B4d: Unclear causality	The response states outcomes without explaining causes or consequences clearly.	clearly explain the [cause/consequence] relationship	it describes an outcome without explaining why it happened
	B4e: Internal inconsistency	Different parts of the response contradict one another.	maintain internal consistency; avoid contradictions	the text first states A and later states not-A
Accuracy & faithfulness	B5a: Factual inaccuracy	Includes factually incorrect content or unsupported factual claims.	ensure factual accuracy; use correct [facts/data]	cited research data are incorrect
	B5b: Prompt drift	Deviates from the prompt's core requirement or setting.	remain faithful to the original [prompt/context]	a story changes a key plot setting
	B5c: Terminology inaccuracy	Uses the wrong or outdated domain-specific terminology.	use correct [technical/domain-specific] terminology	an outdated technical term is used
	B5d: Time or setting inconsistency	Violates the time period or background setting specified by the prompt.	consistent with [time period/setting]	modern terms appear in a 1980s setting
Task-specific quality	B6a: Code quality gap	Misses domain-specific quality requirements for code responses.	include [specific code feature]; use [correct API/method]	the code lacks error handling or uses a deprecated API
	B6b: Creative-writing quality gap	Misses literary quality such as originality, characterization, or plot development.	develop characters/plot with depth	characters are flat and the plot is underdeveloped
	B6c: Dialogue or interaction quality gap	Fails to sustain natural interaction or a consistent character voice.	maintain character voice/interaction quality	the dialogue voice does not fit the character
	B6d: Persuasion or argument quality gap	Lacks argumentative force, persuasiveness, or rhetorical impact.	provide persuasive/compelling [argument/narrative]	an essay makes weak and unconvincing arguments
	B6e: Instruction or tutorial quality gap	Lacks clear, actionable, and complete guidance for a how-to task.	provide clear, actionable steps; ensure instructions are [specific/complete]	a tutorial is too vague to follow in practice
	B6f: Cross-cultural or language quality gap	Misses linguistic or cultural appropriateness in multilingual or cross-cultural tasks.	provide culturally/linguistically appropriate [content]	a translation ignores cultural context

Table 9: Fine-grained taxonomy for Chase-branch `[llm]` rubrics.

Hyperparameter	Value
Rollouts Per Prompt	16
Train Batch Size	96
Ppo Mini Batch Size	48
KL Coefficient	0.001
Learning Rate	1.0×10^{-6}
Learning Rate Scheduler	Constant with Warmup
Maximum Sequence Length	8192
Training Epochs	3

Table 10: GRPO hyperparameter configuration used in CARE training.

Model	Training Method	Per Step Time (s)
Qwen2.5-7B-Base	Rubric RL	318.2
Qwen2.5-7B-Base	CARE	448.9

Table 11: Per-step training time comparison between Rubric RL and CARE on Qwen2.5-7B-Base.

	Arena-Hard-2.0		InfoBench			FollowBench		
	Vanilla	**Style-Ctrl.**	**Easy**	**Hard**	**Overall**	**SSR**	**HSR**	**CSL**
Llama-3.1-8B-Instruct	1.0	2.2	83.7	81.2	82.0	77.6	68.0	3.01
+ Rubric RL	2.6	3.1	86.3	83.2	84.2	80.5	71.8	3.07
+ **CARE**	**8.9**	**8.7**	**88.6**	**86.7**	**87.3**	**83.8**	**73.9**	**3.31**
Qwen3-8B	7.8	3.2	78.7	75.1	76.2	76.4	70.9	2.71
+ Rubric RL	10.9	7.7	82.1	79.0	80.0	79.2	71.4	2.98
+ **CARE**	**13.1**	**16.8**	**85.6**	**83.3**	**84.0**	**83.2**	**72.6**	**3.33**

Table 12: Additional results on Llama-3.1-8B-Instruct and Qwen3-8B with the same evaluation format as Table 1. CARE consistently outperforms Rubric RL on both models.

CARE Reward-Scoring Prompt

```
# Role
You are a rigorous and impartial assessment expert. Your task is to evaluate the model's response
    against the provided scoring rubrics with itemized scoring.

# Input Data
## 1. Context/Instruction
{{instruction}}

## 2. Model Response
{{response}}

## 3. Scoring Rubrics
{{rubrics_list}}

# Task
For each rubric:
1. **Reasoning Analysis**: Provide detailed evaluation of whether the response meets the rubric
    criteria.
2. **Binary Scoring**: Assign 0 (Fail) or 1 (Pass) score.

# Constraints
- Focus exclusively on rubric requirements, Ignore response length, formatting aesthetics, or
    politeness level

# Output Format (JSON)
```

```
**CRITICAL INSTRUCTIONS**:
1. Output must be a single JSON object
2. Format: {"rubric_text": "score", ...},each rubric must map to exactly one key-value pair,values
      must be string "0" or "1". **The key must be copied verbatim from the rubric text in "
     Scoring Rubrics" --- do not paraphrase, truncate, or modify.**
3. No additional text, explanations, or markdown allowed
4. Must preserve rubric order from input

**Example Output**:
{
  "rubric_1_text": "1",
  "rubric_2_text": "0",
  "rubric_3_text": "1"
}
```

Figure 3: CARE reward-scoring prompt used for rubric-based response evaluation.

CARE Rubric-Evolution Prompt

```
# Role

You are a **Rubric Evolution System**. Your task is to compare Agent and Golden responses using an
      "Anchor Comparison" mechanism: first detect and patch **Reward Hacking** vulnerabilities; if
      none are found, extract substantive quality gaps to expand the evaluation **Rubrics**.

# Guidelines: Principles of High-Quality Rubrics

1. **Atomicity**: Each criterion evaluates exactly one independent dimension.
2. **Binary \& Specificity**: Criteria must be objective Yes/No questions. **Avoid vague terms
     like "appropriate", "unnecessary", "sufficient", or "good"** --- they are not objectively
     judgeable. If a constraint targets a specific unwanted pattern, name it explicitly with a
     concrete example (e.g., instead of "Does the response avoid unnecessary disclaimers?", write
     "Does the response avoid adding unsolicited moral disclaimers unrelated to the task, such as
     human rights warnings or safety caveats?").
3. **MECE**: New rubrics must not overlap or conflict with any existing rubric --- whether
     Satisfied or Unsatisfied. Check against the full rubric set (both lists) before adding
     anything new.
4. **Conservative Modification**: Retain all healthy existing rubrics. Only modify or add rubrics
     when Reward Hacking is confirmed or a clear substantive gap is identified.
5. **Language Consistency**: The final list must use the exact same language as `Current Rubrics`.
      Do not translate.
6. **Prefix Scope**:
    * `[llm]`: General quality features, scored by LLM.
    * `[Resist]`: Rigid constraints against Reward Hacking.

# Context

* **User Instruction**: {{instruction}}
* **Satisfied Rubrics** (scored 1 by verifier --- focus your analysis here): {{satisfy_rubrics}}
* **Unsatisfied Rubrics** (scored 0 --- Agent failed these, no hacking possible): {{
     unsatisfy_rubrics}}
* **Agent Response**: {{response_agent}}
* **Golden Response**: {{response_golden}}

> **Anchor Definition --- two distinct roles for the two steps**:
> * **For Step 1 (Hack detection)**: For any rubric the Golden satisfies, it demonstrates a non-
     over-optimized, substantively valid fulfillment of that criterion. Use it to judge *how* the
     Agent satisfies rubrics --- not whether the Agent's overall output is better or worse.
> * **For Step 2 (Divergence extraction)**: The Golden is simply a reference point, not a quality
     ceiling. The substantive gap can go in either direction --- the Agent may be genuinely
     superior to the Golden on some dimension, or genuinely inferior. Extract whichever direction
     represents the largest real difference.

# Analysis Task (Chain-of-Thought)

**Step 1: Reward Hacking Check \& Patching**

* **Scope**: Only analyze **Satisfied Rubrics** --- Reward Hacking can only occur on rubrics the
     Agent appears to satisfy. Unsatisfied Rubrics are listed for context only; do not analyze
     them for hacking.

* **Core Principle**: Reward Hacking means the Agent satisfies the *letter* of a rubric while
     violating its *intent*. The key question is not "Is the Agent's response worse than the
     Golden?" but rather: **"For each rubric the Agent satisfies, does it satisfy that rubric in
     the same substantive way as the Golden, or does it exploit a loophole?"**

* **Hack Pattern Reference** --- the following are illustrative examples of how Hacking can
     manifest. This list is **not exhaustive**: any behavior that formally satisfies a rubric
     while substantively violating its intent counts as Hacking, even if it does not match these
     specific patterns:
    * **Factual substitution** *(example)*: Agent answers about a wrong entity or fabricated
     context that superficially passes rubric checks (e.g., rubric: "Does the response describe
     character X?" --- Agent invents a fictional version of X).
    * **Self-evaluation injection** *(example)*: Agent discovers that "longer output + self-praise
     " yields higher scores, and learns to append large self-assessment paragraphs (e.g., "This
     response comprehensively addresses all aspects...") after a valid answer --- adding no
```

```
    substantive content while inflating length or completeness rubric scores.
    * **...and any other pattern** where the rubric's wording is exploited to pass the check
    without genuinely achieving the rubric's underlying goal.

* **NOT a Hack --- do not trigger patching for these**:
    * Agent response is more verbose or more concise than Golden --- length difference alone is
    not Hacking.
    * Agent response has a different style, tone, or structure --- style divergence is not Hacking
    .
    * Agent response is lower quality but fails rubrics honestly --- capability gap is not Hacking
    .

* **Action**: If a Hack is confirmed, identify the exploited `[llm]` rubric and either rewrite it
    with `[Resist]` prefix, or add a new atomic `[Resist]` negative constraint. The patched
    rubric must be specific enough to block the observed loophole without penalizing legitimate
    responses.

* **Branch**: **If Reward Hacking is detected and patched, proceed directly to Step 3. DO NOT
    execute Step 2.**

**Step 2: Substantive Divergence Extraction**

* **Prerequisite**: Execute ONLY if NO Reward Hacking was found in Step 1.
* **Focus**: Identify the single largest substantive gap between Agent and Golden in logical
    coherence, information depth, accuracy, or task effectiveness. Ignore formatting and style
    differences.
* **Action**: Try to merge the gap into an existing `[llm]` rubric without breaking atomicity ---
    merging is valid regardless of whether the target rubric is currently Satisfied or
    Unsatisfied. If merging is impossible, add one new `[llm]` rubric that is objective, binary,
    and not already covered by any existing rubric (Satisfied or Unsatisfied).

**Step 3: Integration \& Structured Output**

* `final_rubrics` must include **all** rubrics from both the Satisfied and Unsatisfied lists, plus
     any modified/new `[Resist]` or `[llm]` rubrics added in this session. Do not drop any rubric
     that was not explicitly modified.
* In `<ANALYSIS>`, state: (1) Was Hacking detected --- which rubric was exploited and how? (2) If
    no Hacking, what is the core substantive gap? (3) Rationale for the rubric change.

# Output Format

1. **Strictly follow** the structure below. **DO NOT** use Markdown code blocks.
2. Analysis within `<ANALYSIS>` and `</ANALYSIS>` tags.
3. Final rubrics as a valid **JSON list of strings** within `<FINAL_SET>` and `</FINAL_SET>` tags.
4. **No text outside these tags.**

<ANALYSIS>
**Put your detailed analysis here**
</ANALYSIS>

<FINAL_SET>
[
  "<all Satisfied rubrics copied verbatim --- including their prefix (e.g. [llm], [Resist])>",
  "<all Unsatisfied rubrics copied verbatim --- including their prefix (e.g. [llm], [Resist])>",
  "<modified/new [Resist] or [llm] rubric, matching original language>",
  "..."
]
</FINAL_SET>
```

Figure 4: CARE rubric-evolution prompt shared by the Adaptive and Chase branches.