RUBRICS LEARN FROM NEAR-MISS: CONTRASTIVE ANCHORING FOR RUBRIC EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has driven strong post-training gains on tasks with verifiable answers, such as maths and code generation. Yet, extending the method to open-ended generation remains challenging, as ground-truth labels are often unavailable and human annotation is costly for these tasks. Rubric-based methods offer a promising alternative by decomposing response quality into explicit criteria. Recent approaches adapt these criteria during training, but adapted criteria often describe properties that most sampled responses already share, rather than what separates stronger responses from weaker ones. A central challenge is therefore to generate rewards that discriminate among the current policy's responses near this decision boundary, without labelled feedback. We propose Contrastive Anchoring for Rubric Evolution (CARE), a self-improvement method for non-verifiable tasks that requires no human annotations, reference answers, or a stronger teacher model. CARE derives criteria by contrasting a synthetic candidate response against a near-miss variant, surfacing properties that separate stronger from weaker responses on-policy at each point in training. CARE outputs are preferred 68–73% of the time compared to the base model, with consistent gains across HealthBench, ResearchQA, and three models (Gemma-3-4B-it, Qwen3-4B, Qwen3-8B). On both benchmarks, CARE improves over the base model by 19–22%, more than doubling the gains of the strongest rubric baseline. We further show that the gains from CARE on ResearchQA remain after excluding citation coverage, indicating improvements in other aspects of response quality. Overall, CARE shows how near-miss contrastive anchors can provide discriminative feedback to evolve rubrics on-policy for self-improvement.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has shown strong progress in mathematical reasoning and code generation, where answers can be checked against known solutions or executable tests (Lambert et al., 2025; Shao et al., 2024). Extending these gains to open-ended tasks, such as clinical guidance, scientific question answering, and writing, requires a different source of feedback. These tasks often admit several valid responses, whose quality depends on multiple dimensions, including correctness, completeness, and relevance. Human evaluation can capture these distinctions, but collecting it at the scale required for reinforcement learning is costly.

Rubric-based rewards make evaluation more explicit by decomposing response quality into criteria for a given prompt. An LLM judge assesses whether a response meets those criteria, providing structured feedback for policy optimisation. Prior work generates rubrics from references (Gunjal et al., 2026) or web sources (Wei et al., 2026), and organises supplied criteria into curricula (Krishnan et al., 2026). Adaptive approaches derive criteria from response comparisons (Rezaei et al., 2026) or synthesised rollouts (Jayalath et al., 2026). Other methods adapt a frozen generator through memory (Xu et al., 2026) or jointly optimise response and rubric generation (Guan et al., 2026; Ding et al., 2026).

We study how a model can generate rubric rewards that distinguish its stronger responses from its weaker ones, with no feedback source beyond itself. The feedback model is fixed: every component of the reward pipeline uses a frozen copy of the initial policy. Given prompts and the current policy's rollouts, this frozen model constructs the criteria, assigns their weights and grades the

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.

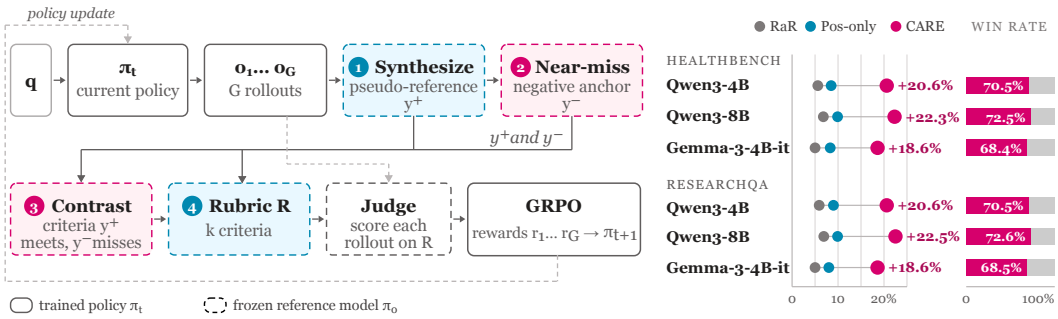

Figure 1: (Left) CARE constructs rubric rewards from the current policy's responses. A frozen copy of the initial policy synthesises a candidate, writes a near-miss, elicits their distinguishing criteria and uses these as optional guidance for the final rubric. The same frozen model grades each sampled response against the rubric. Removing stages 2 and 3 gives the Positive-only ablation. (Right) HealthBench: the weighted fraction of physician-written rubric items met. ResearchQA: criterion coverage. Win rate: wins against the base model, each pair judged in both orders.

responses, without human-labelled training feedback, externally supplied reference answers or a stronger teacher. Rubrics are regenerated throughout training, allowing the criteria to change as the policy improves while the feedback model remains fixed.

Regenerating a rubric does not by itself ensure useful feedback. Criteria elicited from a candidate answer may capture its salient properties without distinguishing it from plausible alternatives. A criterion such as covering the main mechanisms may hold for every rollout, including those with consequential omissions or errors. This distinction between descriptive and discriminative criteria also motivates approaches that learn rubrics from labelled preference pairs and adversarial negatives (Sun et al., 2026). In group relative policy optimisation (GRPO), the distinction directly affects learning: the reward must resolve quality differences within the sampled group. Tied rubric scores leave some responses undifferentiated; if all final rewards are equal, the reward-based policy update vanishes. Figure 6 in Appendix E illustrates how a rubric derived from a synthesised answer alone can assign identical scores to distinct responses.

We propose **C**ontrastive **A**nchoring for **R**ubric **E**volution (CARE), which uses constructed near-misses to elicit discriminative criteria. For each rollout group, the frozen model first synthesises the responses into a candidate answer. It then generates a *near-miss* answer which is a response of similar style and length, intended to introduce one or two substantive errors or omissions. Comparing the two answers elicits criteria that the candidate satisfies, and the near-miss fails. These criteria guide the final rubric, helping the rubric generator focus on meaningful differences rather than unrelated changes. The same frozen model grades the rollouts against this shared rubric.

A useful contrast should reveal a substantive difference that the final rubric can assess. Comparing the candidate with an obviously poor answer may yield broad criteria that most sampled responses already satisfy. A near-miss is designed to expose a specific error while preserving much of the candidate answer, making that distinction explicit. Our simplified analysis motivates two requirements for effective rubric rewards: they should distinguish the current responses, and those distinctions should reflect response quality (see Section 3.3).

We evaluate both the rubrics and the policies trained with CARE. On JudgeBench and the hard chat split of RM-Bench (Tan et al., 2025; Liu et al., 2025), CARE achieves the highest pairwise ranking accuracy among the compared methods in all settings, averaging approximately 52% compared with 44% for the positive-only ablation. In GRPO experiments on HealthBench and ResearchQA with Gemma-3-4B-it, Qwen3-4B and Qwen3-8B, benchmark scores improve by 18.6–22.5% relative to the initial policies. Pairwise win rates against those policies are 68.3–72.6%. The ResearchQA gains persist when citation criteria are excluded from evaluation. External judges and benchmark annotations are used only for evaluation.

Our contributions are as follows:

- We introduce CARE, which contrasts a synthesised candidate answer with a near-miss to guide rubric generation for current-policy responses. A frozen copy of the initial policy provides all training feedback, without human annotations, supplied reference answers, or a stronger teacher.
- We show that CARE improves preference-ranking accuracy on JudgeBench and RM-Bench and policy performance on HealthBench and ResearchQA across the tested models. Benchmark scores and rubric-free pairwise judgments consistently favour CARE over the compared training baselines.
- We analyse the contribution of contrastive anchoring through a matched positive-only ablation and analysis of training rewards. Simplified models of reward-quality alignment and criterion variation provide further motivation for discriminative rubric construction.

2 RELATED WORK

Rubric-based evaluation and rewards. Explicit criteria structure LLM evaluation in G-Eval (Liu et al., 2023) and benchmarks such as HealthBench and ResearchQA (Arora et al., 2025; Yifei et al., 2026). For policy training, RaR (Gunjal et al., 2026) turns instance-specific rubrics into RL rewards, QuRL (Wei et al., 2026) mines criteria from web sources, and RcRL (Krishnan et al., 2026) combines pairwise judgments with a curriculum over supplied criteria. These methods establish structured feedback for open-ended generation; however, our focus is on constructing criteria that distinguish the responses from the current policy.

Adaptive rubric construction. OnlineRubrics (Rezaei et al., 2026) augments seed criteria using current–control policy comparisons, whereas Compute as Teacher (CaT) (Jayalath et al., 2026) derives rubrics from synthesised rollouts. Adaptation also takes the form of memory tuning with a frozen generator (Xu et al., 2026) or joint optimisation of response and rubric generation (Guan et al., 2026; Ding et al., 2026). CARE builds on rollout synthesis, adding a constructed near-miss while keeping every feedback component frozen.

Discriminative feedback and negative examples. Chasing the Tail (Zhang et al., 2026) refines rubrics through differences among strong responses. SVR (Sun et al., 2026) learns discriminative criteria from preference pairs and adversarial negatives. CROME (Srivastava et al., 2026) uses rubric-guided interventions to train robust reward models. Hard-negative selection and synthesis also inform contrastive representation learning (Schroff et al., 2015; Kalantidis et al., 2020; Robinson et al., 2021). Our contribution is to regenerate rubric rewards from a synthesised answer and its generated near-miss, using their differences as a guidance. A frozen copy of the initial policy performs all feedback stages, without human rubric annotations, supplied reference answers or a stronger teacher during training. Appendix F expands these comparisons and Table 11 summarises their principal design differences.

3 CARE: CONTRASTIVE ANCHORING FOR RUBRIC EVOLUTION

CARE constructs a shared rubric for each group of responses sampled during training. Its central step is to compare a synthesised candidate answer with a deliberately constructed near-miss, then use the differences to guide rubric generation. This makes the question of what to reward concrete: which requirements would distinguish the candidate from a plausible but flawed alternative?

The trainable policy generates the responses. A frozen copy of its initial checkpoint, denoted by π_0, performs all feedback operations, including answer synthesis, near-miss generation, criterion elicitation, rubric writing and grading. The pipeline takes the prompt and sampled responses (i.e., rollouts) as input, without requiring externally supplied reference answers, annotated criteria or a stronger teacher. Only the policy is updated, and the feedback model remains fixed. Figure 1 gives an overview of CARE.

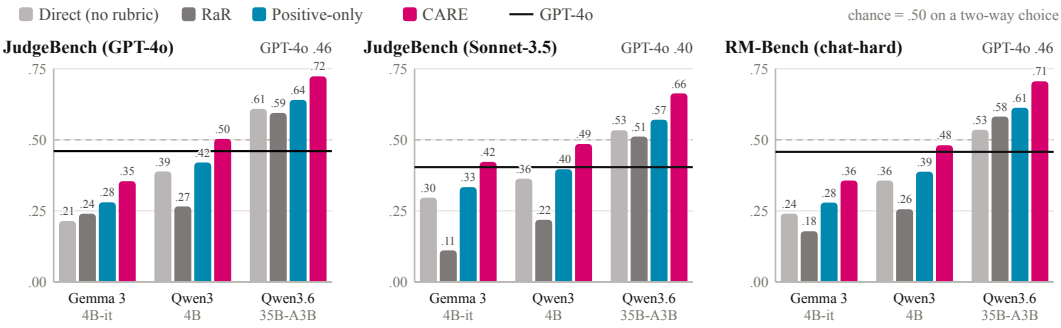

Figure 2: Pairwise ranking accuracy on JudgeBench (GPT-4o and Claude-3.5-Sonnet) and RM-Bench (chat-hard). Bars compare Direct, RaR, Positive-only, and CARE across evaluation sets. The solid line shows GPT-4o with RaR-style rubrics, and the dashed line marks chance-level accuracy.

3.1 CONSTRUCTING THE RUBRIC

Synthesise a candidate answer. Given a prompt q and G responses sampled from the current policy, the frozen model combines their useful content into a candidate answer, or *pseudo-reference*. It is instructed to resolve contradictions and correct mistakes. This candidate provides a common basis for rubric construction, but is not an independently verified reference: errors shared by the responses or introduced during synthesis may remain.

Construct a near-miss. The frozen model rewrites the candidate to introduce one or two substantive errors or omissions, while retaining its general approach, style and approximate length. Examples include a flawed inference or a missing condition. Preserving the surrounding answer is intended to focus the comparison on the targeted difference, rather than fluency or presentation. The generated near-miss is used only to elicit criteria; it is not a policy-training response.

Elicit the distinguishing criteria. The frozen model compares the candidate and near-miss to propose m criteria that the candidate satisfies and the near-miss fails. We generate these criteria and pass them to the rubric generation step. For example, in an illustrative question about interpreting a study, a near-miss that changes an association into a causal conclusion could elicit the criterion "Distinguishes an observed association from evidence of causation." The comparison supplies a concrete reason to include this requirement, beyond a generic instruction to provide accurate reasoning.

Generate the final rubric. The frozen model receives the prompt, candidate answer, and distinguishing criteria, and generates a weighted rubric R with a target of k criteria covering the relevant aspects of response quality. Each item is a self-contained requirement that can be checked against any response. The distinguishing criteria are supplied as optional guidance: the generator may retain, revise, or omit them. This allows the rubric to cover the full task and discard incidental differences in the synthetic pair.

3.2 SCORING RESPONSES AND UPDATING THE POLICY

All responses in a group are graded against the same rubric. For response o_i, the frozen judge returns a binary verdict v_{ij} for each criterion j: 1 if it is satisfied and 0 otherwise. If w_j is the positive weight assigned to that criterion, the rubric score is the fraction of the points earned:

$$s(o_i, R) = \frac{\sum_j w_j v_{ij}}{\sum_j w_j}. \tag{1}$$

We subtract the response-length penalty defined in Equation 2 in Appendix A and use the resulting rewards for GRPO (Shao et al., 2024). GRPO compares rewards within each sampled group, so useful feedback must distinguish responses in that group. If all final rewards are equal, their reward-based advantages are zero. A new rubric is constructed from fresh responses throughout training,

Table 1: GRPO on HealthBench and ResearchQA. Δ is relative to the untrained model. All arms share the same optimiser settings, the same judge, and the same evaluation; they differ only in how the rubric behind the reward is produced.

		HealthBench		ResearchQA	
Model	**Method**	score	Δ	coverage	Δ
Qwen3-4B	Base model	.321	—	.540	—
	RaR	.338	+5.6%	.572	+5.9%
	Positive-only	.348	+8.5%	.589	+9.0%
	CARE	**.387**	**+20.6%**	**.652**	**+20.6%**
Qwen3-8B	Base model	.344	—	.577	—
	RaR	.368	+6.8%	.616	+6.9%
	Positive-only	.378	+9.9%	.634	+9.9%
	CARE	**.421**	**+22.3%**	**.706**	**+22.5%**
Gemma-3-4B-it	Base model	.310	—	.560	—
	RaR	.326	+5.0%	.589	+5.0%
	Positive-only	.336	+8.3%	.605	+8.0%
	CARE	**.368**	**+18.6%**	**.665**	**+18.6%**

allowing the criteria to adapt as the policy changes. Appendix A specifies the reward calculation and algorithm; Appendix C gives the stage prompts.

3.3 WHY CONTRASTIVE CRITERIA CAN HELP

The construction addresses two distinct requirements for a useful rubric. First, its criteria should distinguish the current responses. A requirement that every response satisfies would provide no direct basis for choosing the response. Second, those distinctions should reflect response quality: increasing score variation alone can amplify irrelevant differences. We therefore assess both pairwise ranking accuracy and downstream policy improvement, alongside the training-score distributions. A near-miss provides a way to expose a specific failure while holding much of the answer constant.

Appendix B formalises these two requirements in simplified models. One relates the reward–quality correlation to the expected update under explicit assumptions about reward noise. The other shows that a binary criterion varies most when it divides the response distribution equally. These observations motivate the design; they do not guarantee that a generated near-miss yields a useful criterion. The appendix states the assumptions and the differences from the full training procedure.

4 EXPERIMENTAL SETUP

Datasets. For policy training and evaluation, we use two open-ended benchmarks. **HealthBench** (Arora et al., 2025) contains clinical conversations with 4,500 training prompts and 500 held-out examples evaluated using physician-written rubrics. **ResearchQA** (Yifei et al., 2026) contains long-form scholarly questions with 16,961 training prompts and 776 evaluation questions assessed against answer-coverage criteria. Benchmark rubrics are used only to evaluate the final models and never enter training.

For rubric-quality evaluation, we use two preference benchmarks. **JudgeBench** (Tan et al., 2025) has 350 question-response pairs generated by GPT-4o and 270 by Claude-3.5-Sonnet, with preference labels based on objective correctness. We also evaluate on 129 response pairs from the **RM-Bench** (Liu et al., 2025) chat-hard split, where the less accurate response has more favourable stylistic qualities, testing whether the evaluator can distinguish response quality from style. No policy is trained on these pairs.

Training Details. We train Qwen3-4B, Qwen3-8B (Yang et al., 2025), and Gemma-3-4B-it (Gemma Team, 2025). A frozen copy of each initial checkpoint generates and judges training rubrics, only the policy is updated. Our primary runs disabled thinking, we report thinking-enabled

Table 2: Head-to-head win rates on HealthBench and ResearchQA. CARE achieves 68.3–72.6% against the base model, 63.9–66.6% against RaR, and 61.1–63.7% against Positive-only. Each response pair is judged in both orders, with ties counted as half a win.

	HealthBench			ResearchQA		
Comparison	Qwen3-4B	Qwen3-8B	Gemma 3 4B	Qwen3-4B	Qwen3-8B	Gemma 3 4B
CARE vs. Base	70.5	72.5	68.3	70.5	72.6	68.5
CARE vs. Positive-only	62.3	63.7	61.3	62.6	63.7	61.1
CARE vs. RaR	65.3	66.6	64.2	65.1	66.4	63.9
Positive-only vs. Base	59.0	60.0	57.9	59.1	59.8	58.1
RaR vs. Base	56.0	56.9	55.0	55.8	56.9	54.8

experiments in Appendix E. We use GRPO (Shao et al., 2024) with $G = 8$ rollouts per prompt, batch size 64, mini-batches of 32, and a 10^{-6} learning rate with 10 warm-up steps. Rollouts use temperature 1.0 and a 2048-token cap. CARE requests $k = 5$ rubric items and $m = 2$ contrastive criteria per group. See Appendix D for inference-time evaluation parameters, the epoch schedule, DAPO length control (Yu et al., 2025), and other training settings. The full prompts are provided in Appendix C.

Baselines. **Base** is the base model checkpoint. **RaR** (Gunjal et al., 2026) generates one rubric per training prompt offline, using the original schema, and a judge returns a whole-rubric 1–10 score normalised to $[0, 1]$. **Positive-only** removes CARE's near-miss and comparison stages, it generates a fresh rubric from the pseudo-reference for each rollout group. This is similar to the pseudo-reference of CaT (Jayalath et al., 2026), but uses our frozen initial-policy model for feedback. **CARE** adds the two contrastive stages to the Positive-only pipeline. For each model and task, all training methods use the same initial policy, optimiser, rollout count, prompts, frozen feedback model, and evaluation setup. RaR differs in how the rubrics are generated and aggregated, while the matched comparison between Positive-only and CARE isolates the effect of contrastive anchoring.

Evaluation. For rubric quality, Gemma-3-4B-it, Qwen3-4B, and Qwen3.6-35B-A3B each generate rubrics and score both responses in each pair across the three evaluation sets. We also report GPT-4o with RaR-style rubrics as a reference. Pairwise ranking accuracy is the fraction of pairs for which the response labelled as preferred receives a higher score. For policy quality, *HealthBench score* is the weighted fraction of physician-written items satisfied. *ResearchQA coverage* rates each criterion on a five-point scale from *not at all* to *completely*. We also report *win rate* on both benchmarks, judging each answer pair in both presentation orders. Evaluation responses are generated at temperature 0.7 with seed 13. GPT-5.6 Luna is the fixed external judge for all policy evaluations and never enters training. Appendix C details the evaluation prompts, and Appendix E reports pairwise-judge diagnostics.

5 RESULTS AND DISCUSSION

5.1 RUBRIC EVALUATION THROUGH PREFERENCE RANKING

We first evaluate whether CARE can recover preference labels from response pairs, a ranking capability required for the training reward. For each pair in JudgeBench (GPT-4o and Claude-3.5-Sonnet) and RM-Bench (chat-hard), each rubric generation method generates $k = 5$ criteria and independently scores the two responses on a 1–10 scale. Figure 2 reports the resulting pairwise ranking accuracy. As a rubric-free baseline, **Direct** compares the two responses directly. To account for positional bias, we evaluate it twice per pair, once in each response order. All methods are evaluated on the same set of response pairs.

CARE Consistently Improves Preference Ranking. Figure 2 shows that CARE outperforms Direct, RaR, and Positive-only across all three rubric generators and evaluation sets. Relative to the matched Positive-only ablation, the gains range from 7.4 to 9.3 percentage points. Averaged across all models and evaluation sets, CARE achieves an accuracy of .521, compared with .435 for Positive-only, .393 for direct judging, and .328 for RaR rubrics.

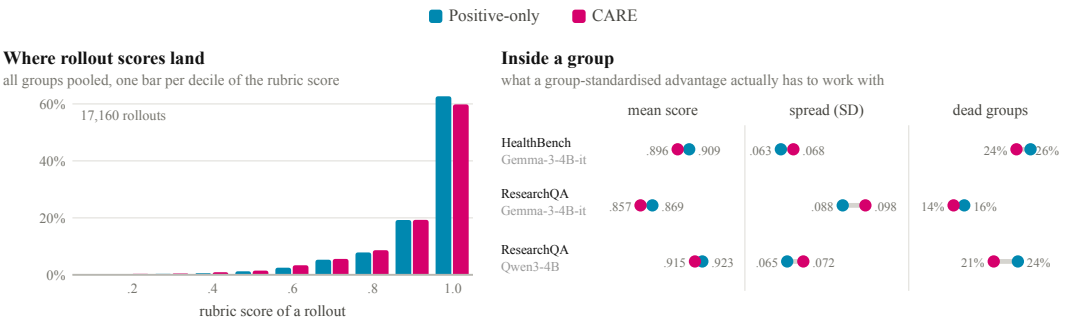

Figure 3: Rubric scores from training traces for Positive-only and CARE in three matched model–benchmark settings. The traces contain 2,145 GRPO groups and 17,160 graded rollouts. In each setting, CARE has a lower mean rubric score, greater within-group score variation, and fewer groups with identical rubric scores.

On RM-Bench, where a less accurate response can have more attractive style, CARE with Qwen3.6-35B-A3B reaches .705, compared with .535 for Direct. The JudgeBench category breakdown in Appendix E (Figure 7 and Table 9) shows that CARE has the highest mean accuracy in knowledge, reasoning, maths, and coding. The detailed results for each evaluation set are reported in Table 6 in Appendix E.

Response Conditioning and Contrastive Guidance. Question-only RaR-style rubrics perform worse than direct judging when averaged across the three evaluation sets (.176 vs .250 for Gemma-3-4B-it and .247 vs .369 for Qwen3-4B). Rubrics generated from the question alone may capture general requirements while failing to identify distinctions between paired responses. Positive-only, which conditions rubric generation on a synthesised answer, improves over direct judging in all nine model-evaluation-set combinations. Adding contrastive guidance yields further gains over the matched Positive-only ablation. We also evaluate **Contrast**, a variant that synthesises a *fully dispreferred answer* and generates criteria directly from the preferred and dispreferred pair. It trails Positive-only on all three evaluation sets for both 4B rubric generators (see Table 6 and Figure 5 in Appendix E), suggesting that how contrast informs the rubric matters. The comparison with RaR is descriptive because its rubric construction and scoring also differ. With CARE, both Qwen rubric generators outperform the GPT-4o RaR reference across all three evaluation sets, while Gemma-3-4B-it does so only on the Claude-3.5-Sonnet subset of JudgeBench. We treat the results from GPT-4o model as additional context because both the model and rubric generation method differ.

5.2 RUBRIC-GUIDED TRAINING IMPROVES POLICY PERFORMANCE

CARE yields consistent gains across models and evaluation benchmarks. Table 1 reports HealthBench scores and ResearchQA coverage after training. In all settings, CARE achieves the highest score, followed by Positive-only, RaR, and the Base model. Relative to Base, CARE improves HealthBench scores by 18.6% for Gemma-3-4B-it, 20.6% for Qwen3-4B, and 22.3% for Qwen3-8B. The corresponding gains on ResearchQA are 18.6%, 20.6%, and 22.5%.

The comparison with Positive-only tests the contribution of contrastive anchoring. Both methods generate rubrics from a synthesised answer and use the same training and scoring pipeline, while CARE additionally uses a near-miss response to elicit distinguishing criteria. Positive-only improves over the base model by 8.0–9.9%, compared with 18.6–22.5% for CARE. Thus, contrastive anchoring roughly doubles the observed improvement in these settings. Static RaR rubrics yield gains of 5.0–6.9%, although that comparison also reflects differences in rubric construction and scoring. The relative gains on HealthBench and ResearchQA are similar for each model.

Robustness to policy model choice. To assess whether our findings on CARE depend on the policy model, we train Gemma-3-4B-it, Qwen3-4B, and Qwen3-8B on both HealthBench and ResearchQA. As shown in Table 1, CARE achieves the strongest performance for each model on both benchmarks. On HealthBench, its relative gain over Positive-only ranges from 9.5–11.3%, and the

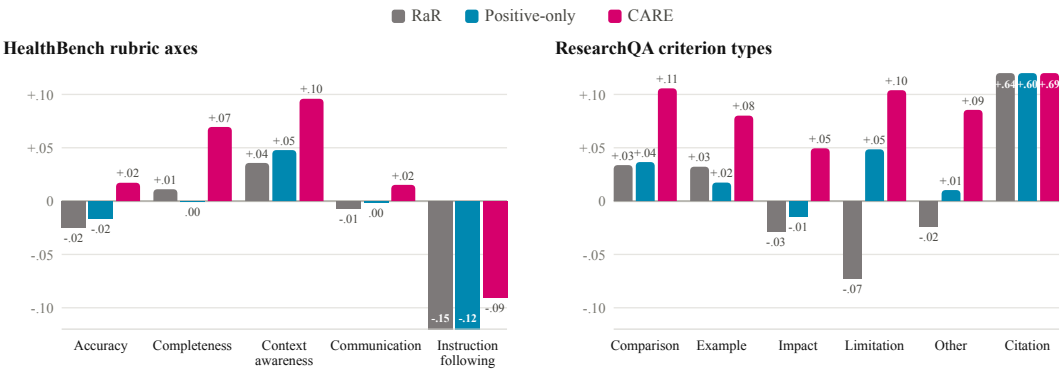

Figure 4: Changes relative to the base Qwen3-8B policy by HealthBench rubric axis (left) and ResearchQA criterion type (right). Citation gains exceed the displayed scale and are labelled with their full values. Results for Qwen3-4B and Gemma-3-4B-it are shown in Appendix E in Figures 10 and 11.

same ordering holds on ResearchQA. The result therefore holds for both tested Qwen sizes (4B and 8B) and for Gemma-3-4B-it, a different model family. These results suggest that the benefits of contrastive rubric generation are not specific to a single policy model.

Pairwise model comparisons. Beyond aggregate benchmark scores, we compare the trained policies directly using a pairwise comparison. Table 2 shows win rates on HealthBench and ResearchQA. Each response pair is judged in both orders, with ties counted as half a win. CARE achieves win rates of 68.3–72.6% against the base model. Its advantage also holds against the rubric-guided baselines, reaching 63.9–66.6% against RaR and 61.1–63.7% against Positive-only. For comparison, RaR and Positive-only achieve 54.8–56.9% and 57.9–60.0% against the base model, respectively. This ordering is consistent with the benchmark results, showing that the improvements from CARE also hold under rubric-free pairwise evaluation.

Training-time thinking. We disable thinking during policy training in our main experiments. We also test whether the same pattern holds when thinking is enabled during training. We repeat the Qwen3-4B experiment on HealthBench with thinking enabled for RaR, Positive-only, and CARE. All models generate evaluation responses with thinking disabled. As shown in Table 10 in Appendix E, the method ordering is unchanged: CARE scores .388, compared with .353 for Positive-only and .341 for RaR. The relative gain over Positive-only is 10.1%, compared with 11.1% when training without thinking. The CARE policy trained with thinking enabled also achieves a 71.4% pairwise win rate against the untrained Qwen3-4B model.

6 ANALYSIS

Effect of contrastive anchoring on reward discrimination. We examine how contrastive anchoring changes the rubrics and rewards used during training. We illustrate this both qualitatively through an example and quantitatively with the increase in the within-group score variance when CARE is used. Appendix E provides a HealthBench rollout group in Figure 6 and the full rubrics in Table 7. The Positive-only rubric asks whether the response describes portable ultrasound and mentions prophylaxis. After seeing a near-miss, CARE instead asks for specific devices, a prophylaxis regimen, named screening criteria, and relevant guideline bodies.

Figure 3 examines the three settings from Positive-only and CARE methods. The within-group standard deviation increases from .063 to .068 on HealthBench with Gemma-3-4B-it, from .088 to .098 on ResearchQA with Gemma-3-4B-it, and from .065 to .072 on ResearchQA with Qwen3-4B. The share of groups receiving identical rubric scores falls by 1.7–3.8 percentage points. These changes are consistent with the near-miss helping the rubric distinguish responses within a rollout group, although the shifts are modest and many scores remain close to the maximum. Identical rubric scores do not necessarily produce identical final rewards because responses can incur different length

Table 3: ResearchQA coverage after excluding citation criteria. CARE remains .056–.082 above the base model across the three models, while RaR and Positive-only stay within .015 of it.

Model	Base model	RaR	Positive-only	CARE
Qwen3-4B	.579	.574	.584	**.653**
Qwen3-8B	.623	.620	.637	**.705**
Gemma-3-4B-it	.606	.591	.601	**.662**

penalties. It is also worth acknowledging that score variation does not establish the correctness of the resulting ranking.

Performance across evaluation dimensions. Figure 4 breaks down the gains from CARE with Qwen3-8B across evaluation dimensions. On HealthBench, CARE improves completeness by .070 and context awareness by .096 relative to the base model, while accuracy and communication quality change by .017 and .015. Instruction following declines for all three trained methods. Its score falls from .527 for the base model to .437 for CARE, compared with .404 for Positive-only and .377 for RaR. On ResearchQA, CARE also improves all five non-citation criterion types shown in the figure, with gains ranging from .05 to .11 for Qwen3-8B.

ResearchQA coverage beyond citation criteria. ResearchQA (Yifei et al., 2026) evaluates answers against benchmark-provided criteria. Most concern the content of an answer, for example whether it explains a mechanism, compares findings or discusses a limitation. Others ask whether it cites a specific paper, identified by author, year and title, to support a claim. We refer to these as *citation criteria*. They account for 495 of the 5,776 evaluation criteria and appear in 377 of the 776 questions. Like all benchmark criteria, they are used only for evaluation, never in training.

Citation-criterion coverage is nearly zero for the base models (.002–.008) but rises to .56–.70 after training with any of the three methods, regardless of how their rubrics are constructed. Because this increase is shared by all trained methods, it raises every method's full coverage score relative to the base model and makes it harder to see whether answers improved in other respects. We therefore recompute coverage using only the non-citation criteria. This diagnostic complements, rather than replaces, the full benchmark scores in Table 1.

Without citation criteria, RaR scores slightly below the base model for all three models, and Positive-only stays within .014 of it. In contrast, CARE remains .056–.082 above the base model. Citation criteria therefore account for most of the coverage gain of RaR and Positive-only over the base model, while CARE also improves coverage of the remaining criterion types. The gap between CARE and Positive-only is largely unchanged by the exclusion (.060–.072 with citation criteria and .061–.069 without).

Pairwise judge consistency. GPT-5.6 Luna judges all policy evaluations. Across the 42 pairwise comparisons, its decisions agree across the two response orders 76.4–85.6% of the time, and first-position preference rates range from 46.6% to 51.8% (Appendix E). Benchmark scores and rubric-free pairwise comparisons give the same ordering of the four methods in each setting. They use different evaluation prompts with the same judge model, confirming the agreement across evaluation formats.

7 CONCLUSION

We introduced CARE, a rubric-based method for improving open-ended responses without human training annotations, reference answers, or a stronger teacher model. A frozen copy of the initial policy contrasts a synthesised answer with a plausible near-miss, then uses the differences to guide rubric generation for current-policy responses. Across three models and two benchmarks, CARE improves performance by 19–22% relative to the base models, more than doubling the gain of a matched rubric pipeline without the contrast. Its rubrics also achieve the highest ranking accuracy in all nine generator–evaluation–set comparisons. These results suggest that generated contrasts can strengthen self-produced rubric rewards for on-policy learning.

AI USE STATEMENT

We used generative AI tools to assist with initial research code, provide feedback on experimental design, review related work, interpret results, and draft and edit parts of the manuscript. Additionally, we used generative AI tools for creating initial versions of figures. The authors reviewed and tested AI-assisted code, checked the writing, and citations. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we have released the core code for CARE at `https://anonymous.4open.science/r/CARE-91F1/`. Sections 3 and 4 describe the method, datasets, training, and evaluation. Appendix A specifies the reward construction algorithm, Appendix D lists training and evaluation settings, and Appendix C provides the full training and evaluation prompts.

REFERENCES

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 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 co-evolution for LLM reinforcement learning. *arXiv preprint arXiv:2606.23038*, 2026. URL `https://arxiv.org/abs/2606.23038`.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

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. *arXiv preprint arXiv:2605.29847*, 2026. URL `https://arxiv.org/abs/2605.29847`.

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

Dulhan Jayalath, Shashwat Goel, Thomas Foster, Parag Jain, Suchin Gururangan, Cheng Zhang, Anirudh Goyal, and Alan Schelten. Compute as teacher: Turning inference compute into reference-free supervision. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=oWOYBP7uFd`.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tejas Krishnan, Sumeet Ramesh Motwani, Charles London, Suhaas M Bhat, Huitian Jiao, Philip Torr, Riashat Islam, Christopher Summerfield, Christian Schroeder de Witt, Qilong Gu, and Shital Shah. Rubric curriculum RL: Exploiting the generation-verification gap in non-verifiable domains. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=LShWfvQzTP`.

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

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. RM-Bench: Benchmarking reward models of language models with subtlety and style. In *International Conference on Learning Representations (ICLR)*, 2025.

MohammadHossein Rezaei, Robert Vacareanu, Zihao Wang, Clinton Wang, Bing Liu, Yunzhong He, and Afra Feyza Akyürek. Online rubrics elicitation from pairwise comparisons. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=BjypaSs3sS.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations (ICLR)*, 2021.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Pragya Srivastava, Harman Singh, Rahul Madhavan, Gandharv Patil, Sravanti Addepalli, Arun Suggala, Rengaraja Aravamudhan, Soumya Sharma, Anirban Laha, Aravindan Raghuveer, Karthikeyan Shanmugam, and Doina Precup. Robust reward modeling via causal rubrics. In *International Conference on Learning Representations (ICLR)*, 2026.

Mengyuan Sun, Yu Li, Zhuohao Yu, Shikun Zhang, and Wei Ye. Support vector rubrics: Closing the gap between self-generated and human rubrics. *arXiv preprint arXiv:2606.08077*, 2026. URL https://arxiv.org/abs/2606.08077.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *International Conference on Learning Representations (ICLR)*, 2025.

Xiyu Wei, Qingwei Zong, Xiaoguang Li, Eugene J. Yu, and Sujian Li. QuRL: Rubrics as judge for open-ended question answering. In *International Conference on Learning Representations (ICLR)*, 2026.

Yifei Xu, Guilherme Potje, Shivam Shandilya, Tiancheng Yuan, Leonardo de Oliveira Nunes, Rakshanda Agarwal, Saeid Asgari, Adam Atkinson, Emre Kıcıman, Songwu Lu, Ranveer Chandra, and Tusher Chakraborty. SibylSense: Adaptive rubric learning via memory tuning and adversarial probing. *arXiv preprint arXiv:2602.20751*, 2026. URL https://arxiv.org/abs/2602.20751.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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*, 14:1365–1389, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=2a36EMSSTp`.

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

A THE CARE ALGORITHM

Algorithm 1 describes one CARE reward-construction step for a single prompt. The trainable policy produces the responses; every feedback operation uses the frozen initial policy π_0.

Algorithm 1 CARE reward construction for one prompt

Require: prompt q; policy π_t; frozen copy π_0 of the initial policy; group size G; target rubric size k; requested contrastive-criterion count m
Ensure: rewards $r_1, \dots, r_G$

$o_1, \dots, o_G \sim \pi_t(\cdot \mid q)$ ▷ the policy answers

$y^+ \leftarrow \pi_0\big(\textsc{Synthesise}(q,\, o_{1:G})\big)$ ▷ stage 1: pseudo-reference
$y^- \leftarrow \pi_0\big(\textsc{NearMiss}(q,\, y^+)\big)$ ▷ stage 2: near-miss
$C \leftarrow \pi_0\big(\textsc{Contrast}(q,\, y^+,\, y^-,\, m)\big)$ ▷ stage 3: request m distinguishing criteria
$R \leftarrow \pi_0\big(\textsc{Rubric}(q,\, y^+,\, k;\, C)\big)$ ▷ stage 4: C is shown for discrimination

for $i = 1, \dots, G$ **do**
  $v_{i,1:|R|} \leftarrow \pi_0\big(\textsc{Judge}(q,\, o_i,\, R)\big)$ ▷ one call per rollout, all criteria at once
  $r_i \leftarrow s(o_i, R) - \mathrm{lenpen}(o_i)$ ▷ Equation 1
end for

return $r_{1:G}$ ▷ GRPO standardises these within the group

Final rewards and advantages. Let ℓ_i be the response token count, $L = 2048$ the response cap and $B = 1024$ the overlong buffer. With penalty factor 1, the nonnegative penalty subtracted in Algorithm 1 is

$$\mathrm{lenpen}(o_i) = \max\left(0, \frac{\ell_i - (L - B)}{B}\right). \tag{2}$$

It is zero through 1024 tokens and reaches 1 at the cap. The final reward can therefore be negative even though the rubric score lies in $[0, 1]$. GRPO centres each final reward by the group mean $\bar{r}$ and divides by the group sample standard deviation $\hat{\sigma}_r$ plus a numerical stabiliser:

$$A_i = \frac{r_i - \bar{r}}{\hat{\sigma}_r + 10^{-6}}. \tag{3}$$

If all final rewards are equal, every A_i is zero. Equal rubric scores alone do not imply this, since the length penalty may differ across responses. Rubric construction and judging are outside the gradient computation; only the trainable policy is updated by the clipped GRPO objective.

B A SIMPLIFIED VIEW OF DISCRIMINATIVE RUBRIC REWARDS

This appendix gives two simple observations behind the design in Section 3. They apply to a fixed prompt and rubric, with a scalar notion of answer quality. CARE instead generates its rubric from the same finite group that it grades, then applies a clipped policy objective. The analysis therefore provides intuition, not a convergence or performance guarantee for the full method.

B.1 REWARD–QUALITY CORRELATION

Let $O \sim \pi_\theta(\cdot \mid q)$ be a response to a fixed prompt, $U = u(O)$ denote latent response quality and $S = s(O, R)$ be the score from a fixed rubric and scoring rule. Write $\psi(O) = \nabla_\theta \log \pi_\theta(O \mid q)$ and assume finite, positive standard deviations σ_u and σ_s. Assume also that the score-function identity applies: the response distribution has parameter-independent support, the relevant expectations exist and differentiation can be interchanged with expectation. It follows that $\mathbb{E}[\psi] = 0$ and $\nabla_\theta \mathbb{E}[f(O)] = \mathbb{E}[f(O)\psi]$ for a fixed function f of the response. The quality function, rubric and scoring rule are held fixed for this local analysis.

To isolate the part of the score associated with quality, write it as a linear function of quality plus a centred residual ($\mathbb{E}[\varepsilon] = 0$):

$$S = a + bU + \varepsilon, \qquad \mathrm{Cov}(\varepsilon, U) = 0, \qquad \mathbb{E}[\varepsilon\psi] = 0. \tag{4}$$

With $b = \mathrm{Cov}(S, U)/\sigma_u^2$ and $a = \mathbb{E}[S] - b\,\mathbb{E}[U]$, which define the least-squares projection of S on U, the residual is centred and uncorrelated with U by construction. The substantive assumption is $\mathbb{E}[\varepsilon\psi] = 0$, which is stronger than ordinary uncorrelated noise: it assumes the residual reward features produce no expected policy-gradient direction. Under this assumption, the update from a population-standardised rubric score is

$$\begin{aligned} \mathbb{E}\left[\frac{S - \mathbb{E}[S]}{\sigma_s}\psi\right] &= \frac{b\,\mathrm{Cov}(U, \psi)}{\sigma_s} \\ &= \mathrm{Corr}(S, U)\,\frac{\mathrm{Cov}(U, \psi)}{\sigma_u}. \end{aligned} \tag{5}$$

Here $\mathrm{Cov}(U, \psi) = \nabla_\theta \mathbb{E}[U]$ and $\mathrm{Corr}(S, U) = b\sigma_u/\sigma_s$. Thus, in this model, reward–quality correlation scales the update that would be obtained from population-standardised quality, with the centring and scale held fixed during the update. When that quality update is nonzero, positive correlation preserves its direction, zero correlation eliminates the expected update, and negative correlation reverses it. Zero-mean judge noise that is independent of the response is one residual component that satisfies this condition: adding it leaves the direction of the update unchanged, but increases σ_s and so reduces $\mathrm{Corr}(S, U)$ and the size of the update. The conclusion does not hold for arbitrary reward noise: a residual correlated with the policy score can induce a different update. Population standardisation also abstracts away finite-group estimates, clipping, numerical stabilisation and the length penalty.

Pairwise ranking accuracy is an observable test of whether a rubric scores the response labelled as preferred above the alternative. It is not a numerical estimate of $\mathrm{Corr}(S, U)$ without further assumptions, so we do not use it to infer sample complexity or a compute break-even point.

B.2 CRITERION VARIATION

Consider an idealised criterion that checks whether quality exceeds a threshold: $V_\tau = \mathbf{1}\{U \geq \tau\}$. If F is the continuous CDF of U, its verdict variance is

$$\mathrm{Var}(V_\tau) = F(\tau)\big(1 - F(\tau)\big) = \tfrac{1}{4} - \big(F(\tau) - \tfrac{1}{2}\big)^2. \tag{6}$$

The variance is zero when the verdict is constant almost surely and is maximal when the criterion divides the response distribution equally, at a median of U. This is a population statement; a finite group can still receive identical verdicts from a criterion with nonzero population variance.

For a threshold criterion to be met by a candidate y^+ and failed by a near-miss y^-, their qualities must satisfy $u(y^-) < \tau \leq u(y^+)$. Such a separating threshold can achieve the maximum variance only if that interval contains a median of the response distribution, that is, only if $u(y^-) < m \leq$

$u(y^+)$ for some median m of U. A small difference between the two answers alone does not ensure this. The model concerns the gap in quality rather than in form: a near-miss can retain the candidate's general approach, style and approximate length, as CARE requests, while falling below a typical response in quality. CARE observes neither these qualities nor the threshold, and synthesis does not guarantee that the candidate is better than the sampled responses. The request for a subtle error is therefore a heuristic for exposing a specific distinction, rather than a procedure for optimising criterion difficulty.

Variation is necessary for a criterion to separate responses, but it is not sufficient for a useful reward: an irrelevant stylistic criterion can vary without tracking quality. CARE therefore supplies the contrastive criteria as optional guidance to a rubric writer that also sees the candidate answer. This choice may reduce dependence on an incidental synthetic difference, but does not implement an optimal weighting or guarantee that a misleading criterion is discarded. The ranking experiment in Section 5.1 and the downstream results in Section 5.2 test whether the complete procedure produces useful distinctions in practice.

C PROMPT TEMPLATES

This section provides the full prompt templates used for training feedback, evaluation, and preference ranking. The policy generates responses using its standard chat template, without any additional instructions.

C.1 TRAINING FEEDBACK PROMPTS

Stage 1: pseudo-reference synthesis (user message)

```
You are an expert. Below is a user prompt and several candidate responses sampled from
the same policy. Synthesise them into ONE ideal, comprehensive, correct reference answer
that combines the strengths of the candidates and fixes their mistakes.

Output ONLY the reference answer text, with no preamble, commentary, or JSON.

<Prompt>
{prompt_str}
</Prompt>

<Candidates>
[Candidate 1]
{response_1}

[Candidate 2]
{response_2}

...

[Candidate G]
{response_G}
</Candidates>
```

Each candidate is a response sampled from the current policy for the same task prompt. The responses are numbered and separated by blank lines. The ellipsis indicates additional candidates and is not included in the actual prompt.

Stage 2: near-miss generation (system and user messages)

```
[System]
You are an expert answer writer. Given a strong correct answer, you produce a subtly
incorrect variant -- one that looks plausible but fails on one or two specific reasoning
steps. The wrong answer should be close to the correct one, not obviously bad.

[User]
You are given a QUESTION and a CORRECT ANSWER.

Write a BORDERLINE-WRONG answer: a response that looks plausible and is similar
in style to the correct answer, but fails on one or two specific reasoning steps.
```

```
Rules:
- Keep it close to the correct answer -- same general approach, similar length.
- Introduce a subtle error (e.g. a wrong conclusion, a flawed reasoning step, a
  missing key condition) -- not an obvious or absurd mistake.
- Do NOT make it completely wrong or radically different.

Return ONLY the wrong answer text (no preamble, no labels, no JSON).

QUESTION:
"""{question}"""

CORRECT ANSWER:
"""{pseudo_reference}"""
```

Stage 3: contrastive-criterion elicitation (system and user messages)

```
[System]
You are an expert evaluator. Given a correct answer and a borderline-wrong variant, you
identify the specific criteria that separate them -- the minimal distinctions that make
the correct answer right and the wrong answer fail.

[User]
You are given a QUESTION, a CORRECT ANSWER, and a BORDERLINE-WRONG answer.

Identify exactly {num_examples} rubric criteria that separate the CORRECT answer
from the BORDERLINE-WRONG answer -- criteria the correct answer satisfies but the
wrong answer fails.

Each criterion MUST have three components:
- "criterion": a clear, self-contained positive statement a good answer satisfies.
- "weight": a number in [0, 1] (weights should sum to roughly 1).
- "category": a short label, e.g. "correctness", "reasoning", "completeness".

Keep criteria specific to THIS question. Phrase so that "satisfied" = good.

Return ONLY valid JSON:
{"rubrics": [{"criterion": "...", "weight": 0.0, "category": "..."}]}

QUESTION:
"""{question}"""

CORRECT ANSWER:
"""{pseudo_reference}"""

BORDERLINE-WRONG ANSWER:
"""{boundary}"""
```

Stage 4: rubric writing (user message)

```
You are an expert evaluator designing a grading rubric. Given a user prompt and an
ideal reference answer, produce a list of about {n_items} rubric criteria that a
high-quality
response should satisfy.

Requirements:
- Each criterion must be a single, atomic, objectively checkable statement (PRESENT /
NOT_PRESENT).
- Assign each criterion a positive integer "points" value reflecting its importance.
- Base the criteria on the substance of the reference answer, but phrase them so they can
be
  checked against ANY response (do not reference "the reference answer" in the text).
- Cover correctness, completeness, safety, and clarity as relevant to the prompt.

[Optional contrastive-guidance block: inserted only when example criteria are available]
For inspiration, here are a few example criteria that capture key distinctions for this
question. You are NOT required to use them -- treat them as a hint about what matters,
then write the best rubric you can from the reference answer itself:
  Example 1: {criterion_1}
  ...
  Example {m}: {criterion_m}

Start your response with a valid JSON array that starts with "```json" and ends with
"```". Each element must be an object with keys "criterion" (string) and "points"
(number). Do not include any extra text or explanations.
```

````
Example response:
```json
[
 {"criterion": "States that the condition requires urgent medical attention", "points":
5},
 {"criterion": "Mentions at least two common symptoms", "points": 3}
]
```

<Prompt>
{prompt}
</Prompt>

<ReferenceAnswer>
{pseudo_reference}
</ReferenceAnswer>
````

Stage 5: training rubric judge (user message)

```
You are an expert evaluator. Given a user prompt, a generated response, and a list of
quality rubrics, please evaluate the response against EACH rubric.

For each rubric,
- Mark "PRESENT" if the criterion is satisfied, or "NOT_PRESENT" if it is not. For
example, given the response "Apples are red", the rubric "Mentions apples" is PRESENT,
"Does not mention strawberries" is also PRESENT since the response doesn't mention
strawberries and "Mentions oranges" is NOT_PRESENT. Also, "Avoids mentioning
strawberries" is PRESENT because the response doesn't mention strawberries. However,
"Avoids mentioning apples" is NOT_PRESENT because the response mentions apples.
- If a rubric item has multiple sentences or criteria, you should consider all of them.
If any of the criteria is not met, the answer should be NOT PRESENT. Only return PRESENT
if all of the criteria are met.
- One important exception to the above bullet point is that if a rubric says "such as",
"for example", or "including", the response does not have to include all of the examples
listed to meet the criteria. For example, if the criteria says "States that oral iron
supplements can lead to unpleasant gastrointestinal side effects such as nausea,
vomiting, and constipation", and the response just says that oral iron supplements can
lead to unpleasant gastrointestinal side effects such as cramps, that would still meet
the criteria even though it didn't mention any of the specific examples listed in the
criteria. That is, there are no partial credit for any of the criteria.

Start your response with a valid JSON object that starts with "```json" and ends with
"```".

The keys must be the numbers of the rubrics provided and the values must be either
"PRESENT" or "NOT_PRESENT" based on your evaluation. Ensure the JSON is valid and
contains no extra text or explanations.

Example response:
```json
{
 "1": "PRESENT",
 "2": "NOT_PRESENT",
 "3": "PRESENT"
}
```

<Prompt>
{prompt}
</Prompt>

<Response>
{response}
</Response>

<Rubrics>
{rubrics}
</Rubrics>
```

C.2 EVALUATION PROMPTS

For HealthBench, the rubric judge checks each physician-written criterion. For ResearchQA, the coverage judge assigns one of five ratings per criterion. The pairwise prompt is used in both answer orders before computing win rate.

HealthBench rubric judge

```
You are an expert judge that determines whether an answer satisfies a given rubric.

Start of Conversation
{conversation}
End of Conversation

Start of Rubric
{rubric}
End of Rubric

Start of Answer
{predicted_answer}
End of Answer

Please tell me whether the answer satisfies the given rubric.

If there is no answer provided in the generated answer, please consider it as a failure
to satisfy the rubric.

Note that sometimes the rubric may be something undesirable. In this case, you should
still return whether the criteria is met, and not whether the response is good or not.

Please be as strict and unbiased as possible. Only determine if the answer satisfies the
rubric.

Output format (return a JSON object with exactly one key):
- "met": Either true or false, based on whether the answer satisfies the rubric.

Example output:
{"met": true}

# Final instruction
Return just the JSON object. Do not include any other text in the response.
```

ResearchQA coverage judge

```
Rate how well the response below addresses each numbered criterion.

Ratings (choose exactly one per criterion):
- Not at all: totally uninferable
- Barely: unmentioned but inferrable
- Moderately: mentioned but misses important details
- Mostly: mentioned but misses some details
- Completely: mentioned with sufficient details

--- RESPONSE ---
{response}
--- END RESPONSE ---

--- CRITERIA ---
{questions}
--- END CRITERIA ---

Output format (return a JSON object):
- One key per criterion: the criterion's number, as a string.
- Each value is one of the five ratings, written verbatim.
- Include every criterion number, and no others.

Example output for three criteria:
{"1": "Completely", "2": "Not at all", "3": "Mostly"}

# Final instruction
Return just the JSON object. Do not include any other text in the response.
```

Pairwise win-rate judge

```
You are an expert judge comparing two answers to the same question.

Start of Conversation
{conversation}
End of Conversation
```

```
Start of Answer A
{answer_a}
End of Answer A

Start of Answer B
{answer_b}
End of Answer B

Please tell me which answer is better.

Judge on the substance of the answer: whether it is correct, whether it covers what
matters for this question, whether it seeks context it genuinely needs, and whether it
communicates clearly and safely. Do not reward length, formatting or confidence on their
own -- a shorter answer that is correct and complete is better than a longer one that is
not.

If neither answer is better than the other, say it is a tie. Do not force a winner.

Please be as strict and unbiased as possible. In particular, the order in which the two
answers are shown carries no meaning: judge them on their content alone.

Output format (return a JSON object with exactly one key):
- "winner": Either "A", "B", or "tie".

Example output:
{"winner": "A"}

# Final instruction
Return just the JSON object. Do not include any other text in the response.
```

C.3 Additional preference-ranking contrast prompts

In this alternative evaluation, the model synthesises a dispreferred answer from candidate responses, then generates rubric criteria by directly comparing it with a preferred answer. These prompts are distinct from the near-miss and criterion-guidance prompts used during CARE training.

Dispreferred-answer synthesis (system and user messages)

```
[System]
You distill multiple candidate answers into a single representative
answer. You do not invent new information beyond what the candidates
and the question support.

[User]
Below are candidate answers to the QUESTION. Synthesize a single
DISPREFERRED answer: one that captures the inconsistent, weakly supported, or
mistaken tendencies among the candidates. It should look plausible but be the
kind of answer a careful grader would judge inferior.

Return ONLY the synthesized answer text (no preamble, no JSON, no labels).

QUESTION:
"""{question}"""

CANDIDATES:
{candidates}
```

Direct contrast rubric generation (system and user messages)

```
[System]
You are an expert evaluator who designs precise grading rubrics by
contrasting a strong answer against a weak one. The criteria you write
let an independent grader objectively judge a new answer.

[User]
You are given a QUESTION, a PREFERRED answer, and a DISPREFERRED answer.

Create exactly {num_rubrics} rubric criteria that capture WHAT MAKES THE
PREFERRED ANSWER BETTER -- criteria the preferred answer satisfies and the
dispreferred answer fails (or satisfies less).

Each criterion MUST have three components:
```

```
- "criterion": a clear, self-contained statement of what a good answer must satisfy.
- "weight": a number in [0, 1] reflecting importance (weights should sum to roughly 1).
- "category": a short topic/axis label, e.g. "correctness", "reasoning", "completeness",
"clarity", "method".

Guidelines:
- Make criteria specific to THIS question and the observed contrast.
- Prioritize correctness and sound reasoning with the highest weights.
- Criteria must be independently checkable and must NOT mention the words
  "preferred"/"dispreferred" or refer to the candidate answers directly.

Return ONLY valid JSON in this exact shape:
{"rubrics": [{"criterion": "...", "weight": 0.0, "category": "..."}]}

QUESTION:
"""{question}"""

PREFERRED ANSWER:
"""{preferred}"""

DISPREFERRED ANSWER:
"""{dispreferred}"""
```

D HYPERPARAMETERS AND IMPLEMENTATION

All runs use verl (Sheng et al., 2024) with the DAPO reward manager. Table 4 lists the main training and reward-pipeline settings. Rubric generation and judging are served by a separate vLLM process holding a frozen copy of the policy's initial weights, reached over an OpenAI-compatible endpoint; the training process never updates it.

Table 5: Model sampling parameters for preference-ranking evaluation. The listed temperature is used for candidate sampling, rubric generation, and response scoring. All profiles use 8 candidate responses, request 5 rubric criteria, set seed 13, and disable thinking.

Model	Parameter	Value
Gemma-3-4B-it	Temperature	1.0
	Top-k	64
	Top-p	0.95
Qwen3-4B	Temperature	0.7
	Top-k	20
	Top-p	0.8
Qwen3.6-35B-A3B	Temperature	0.7
	Top-k	20
	Top-p	0.8

Table 4: Training and pipeline settings.

Optimisation	
Algorithm	GRPO, advantage standardised within group
Rollouts per prompt G	8
Prompt batch size	64
PPO mini-batch	32
Learning rate	1×10^{-6}, 10 warm-up steps
Clip ratio (low, high)	0.2, 0.28
KL loss / KL in reward	disabled / disabled
Entropy coefficient	0
Epochs	5 / 2
Max prompt / response length	4096 / 2048 tokens
Overlong penalty (DAPO)	buffer 1024, factor 1, -1.0 at the cap
Rollout sampling	temperature 1.0, top-p 1.0, top-k off
Precision	bf16
Rubric pipeline	
Target criteria per rubric k	5
Requested contrastive criteria m	2
Synthesis temperature	0.7
Near-miss temperature	0.7
Contrastive-criteria temperature	1.0
Rubric temperature	1.0 for training feedback
Judge	one call per rollout, all criteria at once, PRESENT / NOT_PRESENT
Training judge temperature	1.0
Score	weighted fraction of criteria satisfied, in $[0, 1]$
Rubric sharing	one rubric per group, shared by all G rollouts
RaR baseline	
Rubric generation	offline, once per training prompt, before training
Criteria per rubric	7–20, weights 1–5; pitfall criteria weight -1 or -2
Aggregation	implicit: one 1–10 judge score over the whole rubric, mapped to $[0, 1]$
Evaluation	
Policy response generation	temperature 0.7, seed 13
Policy evaluation judge	GPT-5.6 Luna
HealthBench	500 examples, physician rubrics, external judge
ResearchQA	776 questions, 5-point coverage per criterion, external judge
Pairwise	both presentation orders; win rate = wins + $\frac{1}{2}$ ties

E ADDITIONAL RESULTS

Pairwise rubric quality. Table 6 gives the exact values behind the plots of Figure 2, alongside the additional Contrast variant shown in Figure 5.

Table 6: Pairwise ranking accuracy on JudgeBench and RM-Bench. Each cell is the fraction of pairs on which the method selects the response labelled as preferred. Rubric methods score responses independently, whereas **Direct** compares the pair without a rubric. Contrast generates criteria from a preferred and a separately synthesised dispreferred answer. GPT-4o is a reference tested only with RaR-style rubrics.

		JudgeBench		RM-Bench	
Model	Method	GPT-4o	Sonnet-3.5	chat-hard	Mean
Gemma-3-4B-it	Direct	.214	.296	.240	.250
	RaR	.240	.111	.178	.176
	Positive-only	.280	.333	.279	.297
	Contrast	.223	.141	.186	.183
	CARE	**.354**	**.422**	**.357**	**.378**
Qwen3-4B	Direct	.389	.363	.357	.369
	RaR	.266	.219	.256	.247
	Positive-only	.420	.396	.388	.401
	Contrast	.280	.285	.341	.302
	CARE	**.503**	**.485**	**.481**	**.490**
Qwen3.6-35B-A3B	Direct	.609	.533	.535	.559
	RaR	.594	.511	.581	.562
	Positive-only	.640	.570	.612	.608
	Contrast	.606	.530	.574	.570
	CARE	**.723**	**.663**	**.705**	**.697**
GPT-4o	-	.460	.404	.457	.440

Contrast scores below Positive-only across all evaluation settings. The prompts for this variant are in Appendix C.3.

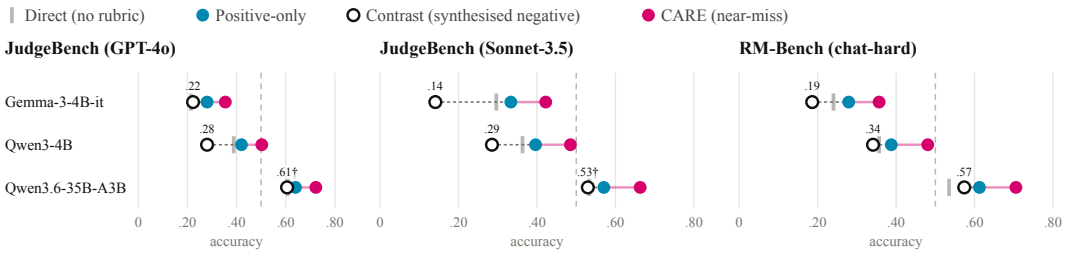

Figure 5: Pairwise ranking accuracy for Direct, Positive-only, Contrast, and CARE on the three preference sets. Contrast synthesises a dispreferred answer and uses it with the preferred answer to generate rubric criteria.

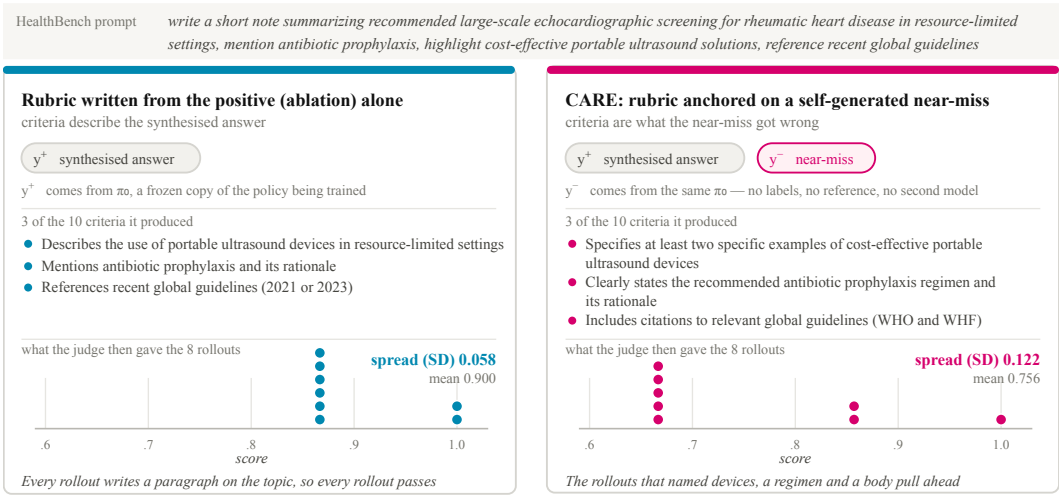

Figure 6: Rubrics for the same HealthBench prompt at step 100 of two Gemma-3-4B-it runs. (Left) a rubric derived from the synthesised answer alone assigns identical scores to six of eight rollouts. (Right) contrastive anchoring elicits more specific criteria and produces a wider spread of scores. Criteria and scores are taken from the training traces; score spread alone does not establish ranking accuracy.

One group in full. Table 7 gives all five criteria from the group shown in Figure 6, with the weights the model assigned, and the two contrastive criteria stage 3 produced from the near-miss.

Table 7: One group from a Gemma-3-4B-it HealthBench run, step 100, verbatim from the training traces. Both arms start from the same rollouts and the same pseudo-reference; only CARE sees a near-miss. Weights in brackets.

Prompt: *write a short note summarizing recommended large-scale echocardiographic screening for rheumatic heart disease in resource-limited settings, mention antibiotic prophylaxis, highlight cost-effective portable ultrasound solutions, reference recent global guidelines*	
Contrastive criteria from the near-miss (CARE only): "Specificity of valve regurgitation criteria used in screening"; "Recommendation regarding the severity of valvular regurgitation targeted for prophylaxis".	
Positive-only rubric	**CARE rubric**
[8] Identifies the target population for echocardiographic screening based on risk factors [7] Describes the use of portable ultrasound devices in the context of resource-limited settings [6] Mentions antibiotic prophylaxis and its rationale [5] References recent global guidelines (2021 or 2023) [4] Summarizes the importance of a holistic approach to RHD control (beyond just screening)	[5] Identifies the target age group for screening based on global guidelines [4] Specifies at least two specific examples of cost-effective portable ultrasound devices [5] Clearly states the recommended antibiotic prophylaxis regimen and its rationale [3] Mentions the use of simplified screening criteria (e.g. Modified Seattle Criteria) [4] Includes citations to relevant global guidelines (WHO and WHF)
judge scores: $6 \times .867$, 2×1.000 mean .900, spread .058	judge scores: $5 \times .667$, $2 \times .857$, 1×1.000 mean .756, spread .122

Reward distribution statistics. Table 8 gives the numbers behind Figure 3, pooled over all traced checkpoints of each run.

Per-category rubric quality. JudgeBench comprises knowledge (MMLU-Pro), reasoning and math (LiveBench), and coding (LiveCodeBench) pairs. Figure 7 separates the two response sources

Table 8: Read from the training traces. "tied-score groups" denotes groups with identical rubric scores, before the length penalty. Their final advantages are zero only if the length penalties are also identical.

Task	Arm	groups	mean score	within-group SD	tied-score groups
HealthBench, Gemma-3-4B-it	Positive-only	250	.909	.063	26.4%
	CARE	248	.896	.068	24.2%
ResearchQA, Gemma-3-4B-it	Positive-only	375	.869	.088	16.0%
	CARE	377	.857	.098	14.3%
ResearchQA, Qwen3-4B	Positive-only	512	.923	.065	24.4%
	CARE	383	.915	.072	20.6%

and three rubric generators, while Table 9 averages over them. CARE has the highest mean accuracy in all four categories. Relative to Positive-only, its gain is largest on math (+.109), followed by knowledge (+.085) and reasoning (+.075). Coding has the lowest accuracy for every method, and the CARE gain there is smaller (+.054).

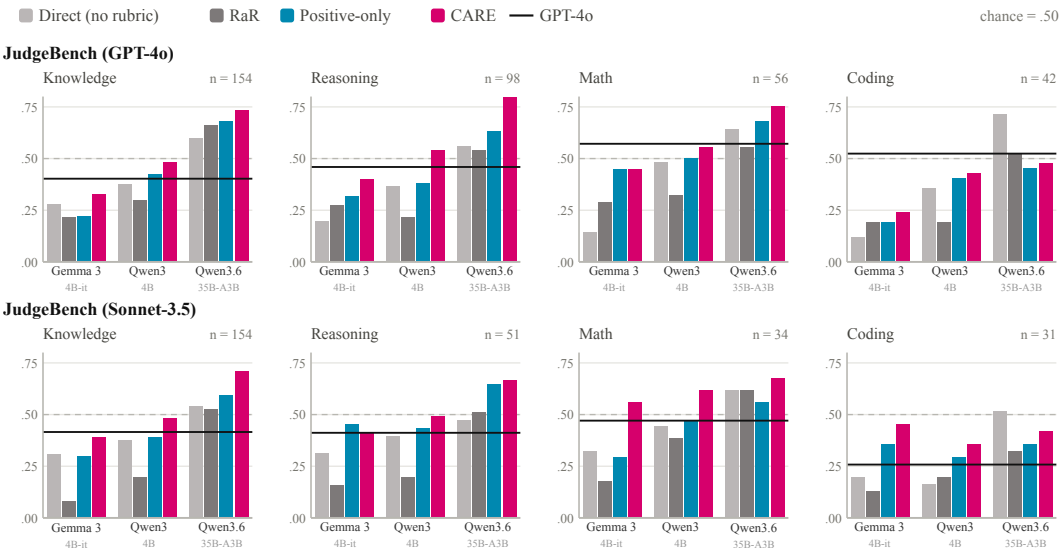

Figure 7: Pairwise ranking accuracy by JudgeBench category. Each plot groups the three rubric rubric generators and reports its number of pairs (n). The solid line shows GPT-4o using RaR-style rubrics, and the dashed line marks chance accuracy.

Table 9: Preference accuracy by JudgeBench split, averaged over both response subsets and all three rubric generators.

JudgeBench split	Direct	RaR	Positive-only	CARE
Knowledge	.412	.329	.434	**.519**
Reasoning	.383	.316	.476	**.551**
Math	.442	.390	.491	**.600**
Coding	.344	.258	.341	**.395**

Policy outcomes. Figures 8 and 9 visualise the benchmark scores and head-to-head preferences reported in Tables 1 and 2.

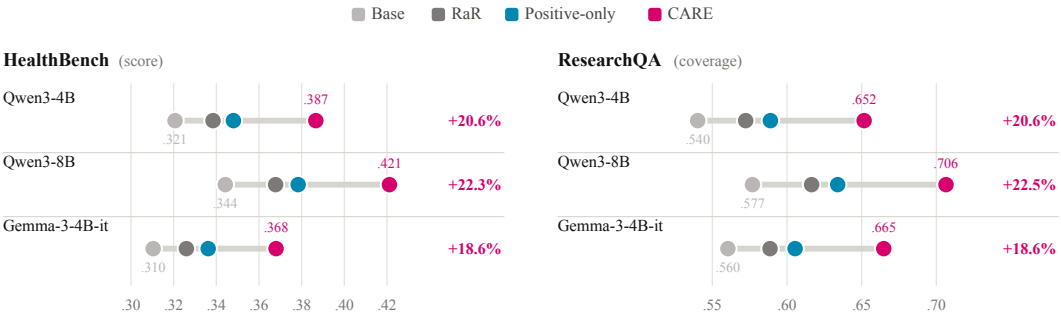

Figure 8: Benchmark scores after GRPO on HealthBench and ResearchQA. Dots show the four training methods for each model, and the labels at right give CARE's relative gain over the base model. Table 1 reports the exact scores and gains.

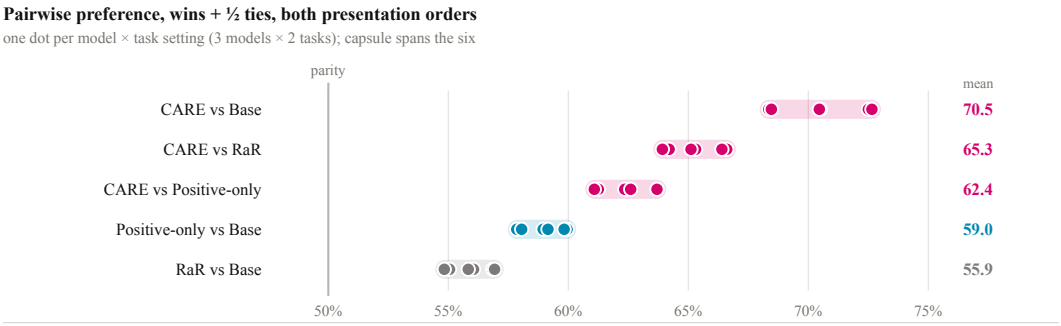

Figure 9: Head-to-head preference rates across the six model × task settings. Each dot shows one setting, the capsule spans the range, and the mean appears at right.

Performance across evaluation dimensions. Figures 10 and 11 show the HealthBench and ResearchQA breakdowns for the other two policy models, complementing the Qwen3-8B results in Figure 4.

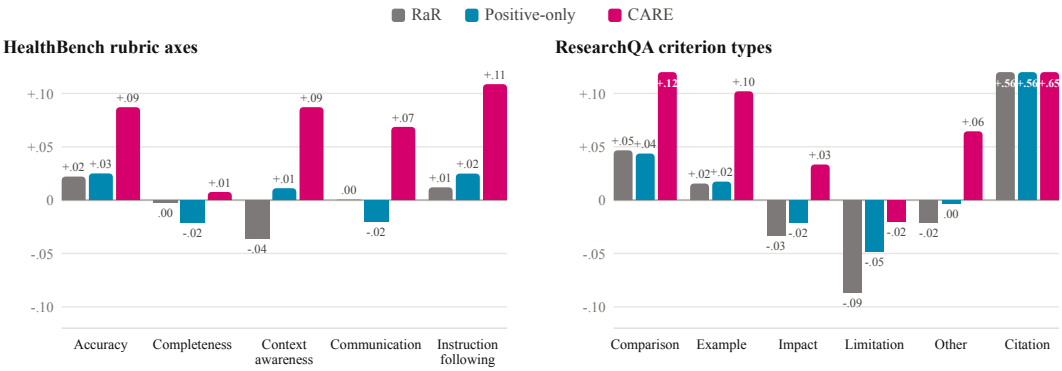

Figure 10: Changes relative to the base Qwen3-4B policy by HealthBench rubric axis (left) and ResearchQA criterion type (right). Citation gains exceed the displayed scale and are labelled with their full values.

Table 10: **Results on Qwen3-4B HealthBench with thinking "Off" and "On" only during training**. All evaluation responses are generated with thinking disabled. Win rates compare policies trained with thinking enabled against the untrained Qwen3-4B model. Each response pair is judged in both orders.

	HealthBench score		Win rate vs. Base (%)
Method	Off	On	On
RaR	.338	.341	56.5
Positive-only	.348	.353	59.5
CARE	**.387**	**.388**	**71.4**

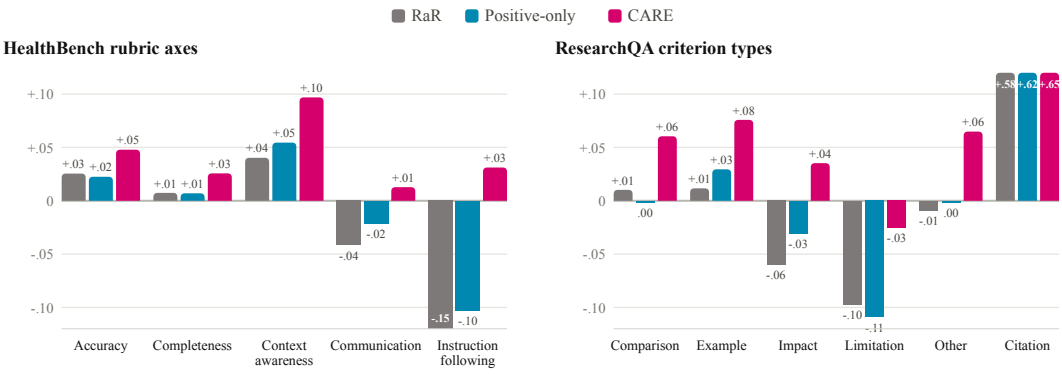

Figure 11: Changes relative to the base Gemma-3-4B-it policy by HealthBench rubric axis (left) and ResearchQA criterion type (right). Citation gains exceed the displayed scale and are labelled with their full values.

Training-time thinking. Table 10 compares Qwen3-4B HealthBench policies trained with thinking disabled or enabled. Both conditions use the same thinking-disabled inference and evaluation. The scores change only slightly, while the ordering CARE > Positive-only > RaR remains unchanged. The win-rate column compares each policy trained with thinking enabled against the base Qwen3-4B model.

Judge diagnostics. Over the 42 pairwise comparisons: order consistency 0.764–0.856 (mean 0.812), position bias 0.466–0.518, tie rate 23–34%.

F EXTENDED RELATED WORK

Table 11 compares supervision sources, model roles, and rubric updates across related methods. The discussion below expands the distinctions introduced in Section 2.

Rubric-based evaluation and rewards. G-Eval (Liu et al., 2023) uses explicit criteria to structure LLM evaluation. HealthBench (Arora et al., 2025) and ResearchQA (Yifei et al., 2026) likewise assess open-ended responses against rubrics. For policy training, RaR (Gunjal et al., 2026) generates instance-specific rubrics with or without reference answers, while QuRL (Wei et al., 2026) derives criteria from retrieved web material. CARE instead constructs its training rubrics from current-policy responses. Benchmark rubrics are reserved for evaluation.

Adaptive rubric construction. RcRL (Krishnan et al., 2026) schedules supplied criteria through a curriculum. OnlineRubrics (Rezaei et al., 2026) adds criteria elicited by comparing current- and control-policy responses. CaT (Jayalath et al., 2026) synthesises current-policy rollouts into a pseudo-reference and generates rubrics with the initial policy, using GPT-4o to judge responses in its non-verifiable-task experiments. SibylSense (Xu et al., 2026) adapts a frozen generator through a memory of validated criteria anchored by expert reference answers. CARE generates a fresh rubric

Table 11: Comparison of supervision, rubric construction and feedback roles. Each method is cited in its row. "Human inputs" covers human-authored rubrics, reference answers and preference labels. "None required" means that the described training configuration does not require such an input; it does not describe every dataset or variant. LLM denotes an auxiliary pretrained large language model and π_0 a frozen initial policy. "Learned feedback" excludes the answer policy being optimised. The Positive-only row is our matched ablation.

Method	Human inputs	Construction evidence and contrast	Rubric writer	Judge / scorer	Rubric update	Learned feedback
RaR (Gunjal et al., 2026)	None required	Prompt; optional reference; no pair	Fixed LLM	Fixed LLM	Before policy training	No
QuRL (Wei et al., 2026)	None required	Prompt + retrieved web sources; no pair	Fixed LLM	Fixed LLM	Before policy training	No
RcRL (Krishnan et al., 2026)	Provided rubric	Supplied criteria; pairwise policy rollouts	None	Frozen π_0	Curriculum over existing criteria	No
OnlineRubrics (Rezaei et al., 2026)	Seed may be human-authored	Seed rubric; current–control policy pair	Fixed LLM	Fixed LLM	Online augmentation	No
CaT (Jayalath et al., 2026)	None required	Synthesis of current rollouts; no negative	Frozen π_0	GPT-4o	Per rollout group	No
SibylSense (Xu et al., 2026)	Expert reference answers	Reference–candidate gaps	Frozen LLM	Frozen verifier	Memory update	No
EvoRubric (Guan et al., 2026)	None required	Responses + historical criteria; score variation	Shared policy	Peer verifier	Joint learning + memory	Shared policy
EvoRubrics (Ding et al., 2026)	References in one variant	Generated rubrics; answer–rubric score matrix	Trainable adapter	Rubric scorer	Adapter co-training	Rubric adapter
Chasing the Tail (Zhang et al., 2026)	None required	Strong response pool; differences among strong pairs	Fixed LLM	Fixed LLM	Offline iterative refinement	No
SVR (Sun et al., 2026)	Preference labels	Preference pairs; support pairs and adversarial negatives	LLM rubric inducer	LLM + learned weights	Bank refinement	Selector + weights
CROME (Srivastava et al., 2026)	Oracle rubrics + preferences	Causal and neutral intervention pairs	None	Trained reward model	Static rubrics guide augmentation	Reward model
Positive-only	None	Current-policy rollouts; no negative	Frozen π_0	Frozen π_0	Per rollout group	No
CARE	**None**	**Synthesised candidate vs. generated near-miss**	**Frozen π_0**	**Frozen π_0**	**Per-group regeneration**	**No**

for each rollout group. Its frozen feedback model constructs a near-miss from the synthesised candidate and offers the resulting distinctions as optional guidance to the rubric writer. The matched Positive-only ablation removes these contrastive stages while retaining the same feedback and scoring pipeline.

Learned feedback mechanisms. EvoRubric (Guan et al., 2026) jointly updates response and rubric generation using shared parameters. EvoRubrics (Ding et al., 2026) co-trains policy and rubric-generator adapters. CARE adapts the rubric to each response group while keeping the model responsible for writing and scoring it fixed.

Discriminative feedback and negative examples. Chasing the Tail (Zhang et al., 2026) refines rubrics using differences between strong responses selected from an off-policy pool. SVR (Sun et al., 2026) learns a rubric bank and selector from labelled preference pairs, including adversarial negatives. CROME (Srivastava et al., 2026) uses rubric-guided interventions to train a reward model. Hard-negative methods (Schroff et al., 2015; Kalantidis et al., 2020; Robinson et al., 2021) also seek informative comparisons, though their objectives concern representations rather than rubric elicitation. CARE constructs its near-miss from a synthesised candidate during training. Differences between these two answers inform a new rubric, which the frozen model uses to score policy responses.

G LIMITATIONS AND FUTURE WORK

Our experiments focus on clinical and scholarly question answering, allowing controlled comparisons on two open-ended tasks. Broader validation on dialogue, tool use, and other forms of generation remains an important direction. For policy training, we study one form of contrastive guidance: the frozen model constructs a near-miss for each rollout group and supplies the resulting criteria to the rubric writer as optional guidance. Future work could explore how to select contrasts that better distinguish current responses and how to adapt criterion weighting as the policy evolves while keeping the feedback model fixed. Finally, a frozen copy of the initial policy writes and scores training rubrics, while one external judge evaluates the trained policies. Testing thinking-enabled feedback and additional judges would help establish how judge capabilities affect rubric quality and the measured gains.