RRC: UNLOCKING GENERATIVE REWARD MODELS IN LLM REINFORCEMENT LEARNING VIA RANKING-BASED REWARD CONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in reward modeling show a paradigm shift from discriminative reward models to generative reward models. However, despite their strong capabilities in response ranking, generative reward models have not realized their potential in reinforcement learning (RL). Our analysis reveals that this limitation arises from a *mismatch* between the comparative nature of generative reward modeling and the scalar scoring paradigm adopted by existing RL algorithms. To bridge this gap, we propose a **R**anking-based **R**eward **C**onstruction (RRC) approach, which enables generative reward models to provide more effective RL learning signals by deriving rewards from relative preference rankings. RRC introduces two complementary strategies: self-competitive ranking, which exploits comparisons among sampled responses, and anchor-guided ranking, which enables scalable ranking-based reward construction with a small set of reference responses. Experiments across open-ended chat and reasoning benchmarks demonstrate that RRC substantially improves RL training with generative reward models, achieving consistent gains over existing reward construction approaches (*e.g.*, 35.8%→41.3% on AlpacaEval2 and 8.0%→11.2% on ArenaHardV2).

1 INTRODUCTION

Reward models play a central role in reinforcement learning (RL) for large language models (LLMs) (Wang et al., 2024a; Ji et al., 2025; Xiao & Zhu, 2025; Wang et al., 2025a; 2026a). They are typically trained in a *discriminative* manner to assign scalar reward scores to sampled responses, which serve as learning signals for RL optimization. Despite their empirical success, discriminative reward models suffer from inherent limitations. Building on the backbone of LLMs, discriminative reward models do not fully utilize the text generation capabilities of LLMs. Consequently, they do not take advantage of key intrinsic strengths of LLMs, such as instruction fine-tuning and explicit chain-of-thought (CoT) reasoning (Wei et al., 2022; Zhang et al., 2025). This *capability gap* is not merely a practical inconvenience but rather a fundamental barrier. It prevents the reward model from capitalizing on insights revealed by recent advances in LLMs, including the use of large-scale fine-tuning to enhance generalization (Shi et al., 2023; Yang et al., 2024) and the use of additional inference-time computation to improve performance (Muennighoff et al., 2025).

To overcome these limitations, recent research has explored *generative* approaches to developing reward models (Zhao et al., 2026; Hong et al., 2025). Unlike traditional discriminative reward models, generative reward models represent preferences through text generation. For instance, given an input and a pair of responses, a generative reward model can produce a textual judgment that explicitly reflects which response is better and why. Although generative reward models have demonstrated strong performance in response ranking, their advantages have not yet been fully transferred to RL (Chen et al., 2025; Wang et al., 2026b; Guo et al., 2025b). As a result, how to better leverage generative reward models within RL remains an open research question. To date, existing strategies are significantly insufficient to address this challenge. For example, GenRM (Zhang et al., 2025) uses the probability of a designated preference token as the reward scores. However, this approach often suffers from probability collapse when CoT reasoning is employed. Additionally, GRAM (Wang et al., 2025b) removes explicit CoT reasoning and generates only preference tokens, thereby producing scalar reward scores like discriminative reward models.

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.

All of the existing efforts overlook a fundamental distinction between discriminative and generative reward modeling paradigms: discriminative reward models are inherently designed to produce scalar reward scores, whereas generative reward models are naturally formulated to perform comparative ranking. We argue that forcing generative reward models into a scoring-based role within RL can lead to suboptimal preference prediction and policy optimization. To substantiate this claim, we design an analysis experiment that systematically compares discriminative and generative reward models across two settings (see Figure 1): response ranking and RL. *The results reveal a clear mismatch between the comparative nature of generative reward modeling and the scalar scoring paradigm adopted by existing RL algorithms.*

In this paper, we explore a solution to this limitation that leverages generative reward models solely for response ranking rather than requiring them to generate scalar reward scores in RL. To this end, we propose a **R**anking-based **R**eward **C**onstruction (RRC) approach, which constructs scalar reward scores directly from the relative preference orderings among sampled responses. Specifically, our RRC approach comprises two ranking mechanisms: self-competitive ranking and anchor-guided ranking, each providing a principled way to transform ranking information into reward scores. In self-competitive ranking, we construct reward scores from the mutual comparisons among sampled responses. This basic idea is that responses generated by the current policy naturally form a competitive set, allowing relative quality differences to be exploited and guiding policy optimization. In anchor-guided ranking, we consider scalability to sampled response sets by introducing a small number of anchor responses as reference points. This design reduces the number of calls to the generative reward model from $\mathcal{O}(m^2)$ to $\mathcal{O}(m \cdot n)$, where m is the number of sampled responses and n is the number of anchor responses. The resulting reward scores can be seamlessly integrated into existing RL algorithms, such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025).

Through extensive experiments on six benchmarks including open-ended chat and reasoning tasks, we show that RRC can provide a more effective way to leverage generative reward models within RL. Notably, compared to the probability-based reward baselines, RRC achieves significant improvements in RL performance (*e.g.*, 35.8%→41.3% on AlpacaEval2, 8.0%→11.2% on the Arena-HardV2, and 52.9%→57.3% on the MMLU-Redux).

Beyond these gains, our results show that RL performance with RRC can be effectively scaled by increasing inference-time computation in generative reward models and by expanding the number of anchor responses (see Figure 3). This is appealing, as RRC introduces new dimensions for scaling RL. These empirical scaling behaviors are further supported by our theoretical analysis in Appendix A, which suggests that RRC relaxes the constraints imposed by scalar rewards derived from discriminative reward models, thereby enabling a higher attainable performance ceiling for RL.

2 PRELIMINARIES

2.1 TRAINING REWARD MODELS

We define a reward model as a function $r_\phi(x, y)$, where ϕ denotes the model parameters, x is the input, and y is the corresponding response. Throughout this work, an *input* refers to an arbitrary token sequence provided to the LLM, such as "*Why the sky appears blue?*", and a *response* denotes the token sequence generated by the LLM conditioned on that input, such as "*This is because shorter-wavelength light is scattered more strongly by molecules*". To date, mainstream reward model architectures can be categorized into two types: *discriminative* and *generative*.

Discriminative Reward Models. Discriminative reward models produce scalar reward scores using a regression-style architecture. They are typically implemented as a stack of Transformer encoder layers, or as a Transformer decoder without a Softmax layer at the output. In practice, a pre-trained LLM is commonly used as the backbone. The concatenated input–response pair (x, y) is fed into the backbone model, and the final-layer hidden representation is then mapped to a scalar reward score via a linear projection. This model can be trained via a Bradley-Terry loss function (Bradley & Terry, 1952):

$$\mathcal{L}_{\mathrm{d}} = -\mathbb{E}_{(x,y_a,y_b)\sim D_r}\big[\log(\sigma(r_\phi(x, y_a) - r_\phi(x, y_b)))\big] \tag{1}$$

where D_r is the training dataset consisting of tuples of input x and response pair (y_a, y_b) with the preference $y_a \succ y_b$. Once trained, the reward model can be naturally used as a scoring function that assigns a numerical reward $r_\phi(x, y)$.

Generative Reward Models. Compared to discriminative reward models, generative reward models fully leverage the text generation capabilities that LLMs are fundamentally designed for (Liang et al., 2025; Wang et al., 2025b), producing reward signals through natural language generation rather than direct scalar scoring. Specifically, they use an LLM to generate preference-related tokens, given a natural language prompt c and a tuple (x, y_a, y_b). The prompt describes the task in natural language, and the model predicts a label token w that aligns with the human preference l, where $l = \text{A}$ denotes preference for y_a, and $l = \text{B}$ indicates preference for y_b. The model can be trained by

$$\mathcal{L}_{\text{g}} = -\mathbb{E}_{(c,x,y_a,y_b,l)\sim D_r}\big[\log \pi_\phi(w = l|s)\big] \tag{2}$$

where s denotes the string $[c, x, y_a, y_b]$, and $\pi_\phi(\cdot)$ denotes the probability of token prediction.

Building on this generative formulation, recent studies further showed that framing reward prediction as a reasoning task can better exploit the reasoning capabilities of LLMs, leading to improved reward modeling performance (Chen et al., 2025; Guo et al., 2025b; Wang et al., 2025c). In these approaches, the model is trained to generate explicit CoT.

2.2 Applying Reward Models to RL

In RL, the policy optimization objective can be written as $\max_\theta \mathbb{E}_{x\sim\mathcal{D},o\sim\pi_\theta(\cdot|x)}[r_\phi(x, o)]$, where x denotes an input prompt sampled from the dataset $\mathcal{D}$, o denotes a response generated by the current policy $\pi_\theta(\cdot|x)$, and θ denotes the parameters of the policy model. A commonly used algorithm to optimize this objective is proximal policy optimization (PPO) (Schulman et al., 2017). However, PPO is often complex and unstable due to its reliance on an additional critic model to estimate advantages. To address this issue, GRPO (Shao et al., 2024) replaces the learned value function with relative rewards computed within a group of sampled responses $\{o_1, o_2, \ldots, o_m\}$, where m denotes the group size. Specifically, the advantage for a sampled response o_t is computed as $A_t = \frac{r_\phi(x,o_t)-\mu}{\sigma}$, where μ and σ denote the mean and standard deviation of reward values within the sampled group, respectively. Under this formulation, reward models are required to provide scalar-valued signals for each sampled response. Discriminative reward models naturally satisfy this requirement, as they are explicitly designed to assign scalar reward scores to individual responses.

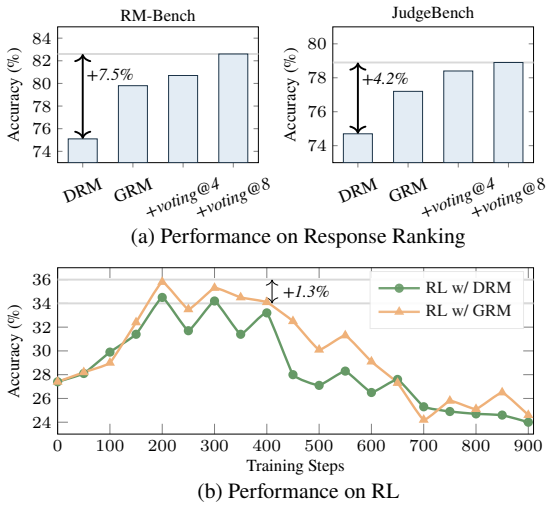
When applying generative reward models to RL, reward scores are typically constructed from the predicted probabilities of designated preference tokens (Zhang et al., 2025; Wang et al., 2025b). Specifically, the reward score is computed for a single input–response pair (x, o). During the RL training, we first obtain a reference response $o_{\text{ref}} = \arg\max_o \pi_\theta(o|x)$ via greedy decoding from the current policy. We then concatenate the context prompt c, the input x, the sampled response o, and the reference response o_{ref} into a single sequence $s_o = [c, x, o, o_{\text{ref}}]$. The generative reward model $\pi_\phi(\cdot)$ is prompted to predict which response is preferred, and the final reward for the pair (x, o) is defined as the probability that o is preferred over o_{ref}. Concretely, if o is designated as "Response A" in the prompt, the reward score is

$$r_\phi(x, o) = \pi_\phi(w = \text{A} \mid s_o) \tag{3}$$

where w denotes the predicted preference token and the resulting reward score lies in the range $[0, 1]$.

3 Ranking vs. Scoring in Reward Models

Both discriminative and generative models are commonly adopted for reward modeling, but we found that generative models are better suited for response ranking, rather than for serving as a score-based role in RL. To study this issue, we trained both a discriminative and a generative reward model using the same labeled preference dataset with LLaMA-3.2-3B-Instruct serving as the backbone. We then evaluated these models on pairwise ranking benchmarks, including RM-Bench (Liu et al., 2025) and JudgeBench (Tan et al., 2025), as well as in an RL setting where reward scores are constructed from the predicted probabilities of preference labels. As shown in Figure 1, generative reward models significantly outperform discriminative ones on response ranking tasks. However, this advantage largely diminishes in the RL setting. Although preliminary, these results are consistent with prior findings reported by Wang et al. (2025b) and Guo et al. (2025b): when trained on the same preference data, generative reward models significantly outperform discriminative models on response ranking, but yield only marginal improvements when used as a score-based role in RL.



(a) Performance on Response Ranking

(b) Performance on RL

Figure 1: Comparison between discriminative and generative reward models under response ranking and RL settings. RL performance is evaluated using AlpacaEval2.

We attribute this discrepancy to a mismatch between the optimization objective of generative reward models and the requirements of scalar reward scoring. Specifically, discriminative reward models are designed to directly predict a scalar reward score for each response, where the score represents the quality of the response and can be easily used as a learning signal in RL. In contrast, generative reward models are typically optimized to express preferences in the form of comparative judgments. In practice, when applying generative reward models in RL, we use the probability of predicted preference tokens as reward signals, thereby bridging textual preference prediction and scalar scoring. However, this suffers from inherent limitations, as illustrated in Figure 2(a). On one hand, when explicit reward reasoning is employed, preference token probabilities often collapse, making them poorly calibrated as reward scores. On the other hand, the predicted probabilities also reflect the model's confidence in its judgment, which is not necessarily aligned with human preference and might be noise. Therefore, the potential of generative reward models in preference modeling is weakened when they are directly used as a score-based role.

4 Proposed Method

We now present our approach for better leveraging the comparative strengths of generative reward models. We first discuss the desiderata for reward construction in RL and then introduce RRC.

4.1 Desiderata for Reward Construction in RL

Given an input x and a sampled response o, an effective reward function $r(x, o)$ for LLM reinforcement learning should satisfy the following two properties.

Property 1. *For any two responses (o_i, o_j) sampled under the same input x, the reward should preserve the preference ordering, e.g., $o_i \succ o_j \rightarrow r(x, o_i) > r(x, o_j)$.*

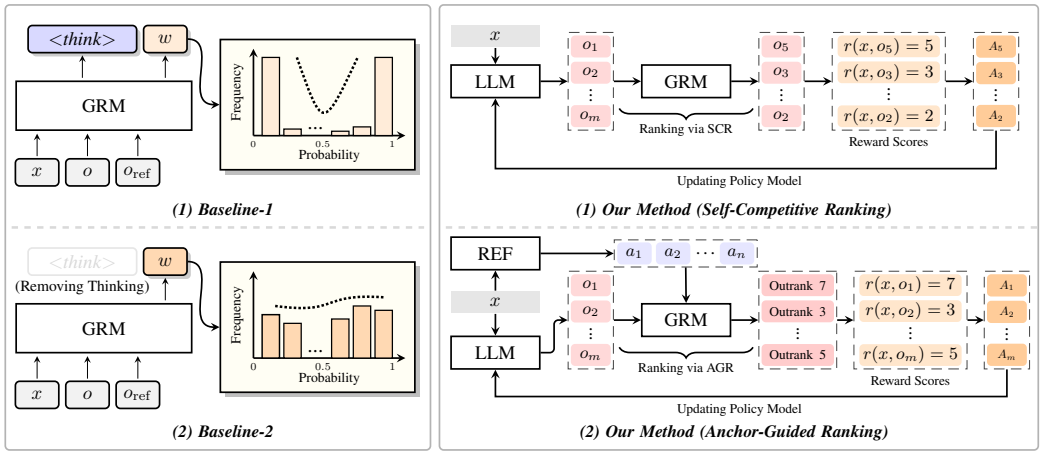
We refer to this property as *order preservation*. It ensures that policy updates are directionally consistent with preference judgments, so that responses more aligned with human preferences are consistently reinforced.

Property 2. *The reward difference should reflect the relative quality gap between responses: $r(x, o_i) - r(x, o_j) \propto \Delta q_{ij}$, where Δq_{ij} denotes the latent oracle preference margin between responses o_i and o_j.*

We refer to this property as *margin awareness*. This property enables meaningful advantage estimates, preventing all preference differences from being treated as equally informative during optimization. As shown in Appendix A, we further provide a proof from a policy gradient perspective, showing how this property directly affects the magnitude and effectiveness of optimization updates.

4.2 Ranking-Based Reward Construction

When applying reward models in RL, optimization performance depends on whether the reward scores they provide satisfy both order preservation and margin awareness. Guided by these two properties, we redesign the reward construction process when using generative reward models. As illustrated in Figure 2(b), we propose an RRC approach. Unlike previous work, we do not utilize preference token probabilities to construct reward scores. Instead, we construct reward scores solely



(a) Probability-Based Reward Construction (b) Ranking-Based Reward Construction

Figure 2: **(a) Probability-Based Reward Construction.** We use the predicted probability of the preference token as the scalar reward signal (denoted as *Baseline-1*). We also remove explicit reasoning to encourage output diversity and improve margin awareness (denoted as *Baseline-2*). **(b) Ranking-Based Reward Construction.** We obtain reward scores from relative rankings obtained either among sampled responses or between sampled responses and anchor responses. Here, "REF" denotes the reference policy used to generate anchor responses. The Python-style ranking algorithms for SCR (Alg. 1) and AGR (Alg. 2) are provided in the Appendix.

from response rankings. Specifically, RRC is implemented through two complementary ranking mechanisms: *self-competitive ranking* and *anchor-guided ranking*.

Self-Competitive Ranking. The goal of self-competitive ranking (SCR) is to construct reward signals purely from the relative ordering among sampled responses, rather than relying on absolute scoring. In RL, responses generated under the same input naturally form a comparable set, and their relative quality provides a more reliable basis for reward construction than isolated scalar scores (Yuan et al., 2023; Shao et al., 2024). This motivates us to treat these responses as competitors in a self-contained ranking process. Formally, given an input x, we sample a group of responses $\{o_1, o_2, \ldots, o_m\}$ from the current policy. We then query the generative reward model to perform pairwise preference judgments within this set. For each pair (o_i, o_j), the generative reward model predicts a preference relation:

$$o_i \succ o_j \quad \text{or} \quad o_j \succ o_i \tag{4}$$

These pairwise preferences induce a tournament graph over the response set. Based on the pairwise comparisons, we assign each response a reward score via the number of wins:

$$r(x, o_i) = \alpha \times \sum_{j \neq i} \mathbf{1}[o_i \succ o_j] \tag{5}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function and α denotes a scaling factor. Intuitively, responses that are preferred over more competitors receive higher reward scores, reflecting stronger relative quality within the sampled group.

This self-competitive construction satisfies both desiderata for reward construction in RL. Order preservation is guaranteed since higher-ranked responses receive larger rewards. Additionally, margin awareness emerges naturally, as reward differences reflect the number of relative wins against other responses, providing graded learning signals.

Majority Voting for Robust Ranking. Pairwise preference predictions from generative reward models can be stochastic, especially when the reward reasoning is involved, which may introduce noise into individual comparisons. To improve robustness, we allocate additional inference-time budget to stabilize preference estimation (Muennighoff et al., 2025; Chang et al., 2025). Specifically, we adopt a majority voting scheme when aggregating pairwise preferences. For each response pair, we query the generative reward model multiple times and collect a set of preference predictions. The final preference relation is then determined by majority vote across these predictions.

Conflict-Aware Ranking Adjustment. Pairwise preference predictions produced by generative reward models are not guaranteed to be globally consistent. In practice, cyclic inconsistencies may arise, such as

$$o_i \succ o_j, \quad o_j \succ o_k, \quad \text{but} \quad o_k \succ o_i \tag{6}$$

which violate transitivity and induce conflicts in the induced ranking. To mitigate this issue, we perform conflict-aware ranking adjustment by treating pairwise preferences as a directed weighted graph, as shown in Property 3.

Property 3. *Given inconsistent pairwise preferences produced by a generative reward model, a Kemeny-rule-based aggregation can recover a globally consistent ranking that maximizes the weighted agreement with all pairwise preferences, where the weight of each preference is determined by its support degree (e.g., voting count) from the model.*

The proof of Property 3 is provided in Appendix A. Specifically, given a sampled set of responses, we construct a directed weighted graph $G = (V, E)$, where each node $v_i \in V$ corresponds to a response o_i, and each directed edge $e_{ij} = (o_i \to o_j) \in E$ denotes a predicted preference $o_i \succ o_j$. The weight of each edge is defined as the number of times the preference $o_i \succ o_j$ is predicted by the generative reward model across multiple stochastic queries. Following the Kemeny rule's core idea of minimizing global pairwise disagreements, we transform the conflict resolution problem into finding the minimum weight feedback edge set of G, where removing edges with the smallest total weight eliminates cycles while retaining as many high-weight preferences as possible. In practice, we adopt the greedy heuristic of Davenport & Kalagnanam (2004). Specifically, at each step, we prioritize the preference with the largest weight difference between the two directions and enforce its majority-supported order. We then propagate this ordering through transitive closure to maintain an acyclic graph.

Anchor-Guided Ranking. While SCR provides a principled way to construct rewards from pairwise comparisons, it requires $\mathcal{O}(m^2)$ preference queries for a group of m sampled responses, which becomes computationally prohibitive when using generative reward models. This quadratic cost limits scalability, especially in modern RL algorithms that rely on large-scale sampling to improve exploration and stability. To solve this problem, we propose an anchor-guided ranking (AGR) mechanism that reduces the number of required comparisons while preserving relative ordering information. The basic idea is to introduce a small set of reference responses, referred to as *anchors*, which serve as stable landmarks for estimating the relative quality of sampled responses.

Formally, given an input x, we collect a small set of anchor responses $\{a_1, a_2, \cdots, a_n\}$, where n denotes the size of the anchor responses. Our experimental results show that even a small value of n is sufficient to achieve strong performance (see Figure 3). For each sampled response o_i and each anchor a_k, we query the generative reward model to predict a pairwise preference relation, indicating whether $o_i \succ a_k$ or $a_k \succ o_i$. Based on these comparisons, we define the reward score of the i-th sampled response as

$$r(x, o_i) = \alpha \times \sum_{k=1}^{n} \mathbf{1}[o_i \succ a_k] \tag{7}$$

Intuitively, responses that outperform more anchors are considered to have higher relative quality. Furthermore, our AGR naturally supports majority voting by querying the generative reward model multiple times for each anchor comparison (o_i, a_k) and aggregating the preferences. This provides an additional inference-time scaling dimension to improve reward reliability, while still keeping the overall query complexity at $\mathcal{O}(m \cdot n)$. In this way, we provide a comparison between SCR and AGR in Appendix C.1. In this mechanism, anchor construction is key to its effectiveness. In this work, anchors are generated from a reference policy. As shown in Appendix A, our theoretical analyses indicate that, compared to alternative designs such as selecting anchors from the current policy's samples, this strategy avoids coupling anchor quality with the evolving policy, thereby providing a more stable basis for comparison.

5 Experiments

We evaluated RRC in open-ended RL settings, where reward models rather than verifiable rules must provide rewards.

Method	*w/ Thinking*					*w/o Thinking*				
	AE2	AH2	WiB	MM_R	MATH	AE2	AH2	WiB	MM_R	MATH
SFT (*w/ LLaMA-3.1-8B-Instruct*)	27.4	6.4	46.6	55.4	38.8	23.7	6.2	44.5	50.4	36.4
+ DPO	29.8	7.0	48.8	52.1	39.6	26.1	7.5	45.3	51.4	36.8
+ SimPO	31.2	7.2	50.3	54.7	40.4	28.2	7.0	48.4	50.2	38.6
RL Training with 3B-Scale Reward Models										
+ DRM	30.5	7.0	48.7	52.3	42.6	27.6	7.2	46.3	50.2	40.4
+ GRM w/ PRC	31.4	7.8	50.2	52.4	44.8	28.4	6.8	48.1	51.7	41.2
+ *Removing Thinking*	32.4	7.2	51.7	53.6	43.0	28.8	8.0	49.2	50.0	41.2
+ GRM w/ RRC-SCR	36.4	8.8	55.6	54.4	45.6	33.1	8.2	53.0	52.4	42.0
+ *voting@8*	**37.8**	9.8	56.7	**56.7**	47.2	33.8	9.0	53.2	53.7	44.8
+ GRM w/ RRC-AGR	35.8	9.8	56.8	55.0	46.4	32.8	8.8	52.9	52.7	45.4
+ *voting@8*	37.5	**10.2**	**58.2**	56.4	**48.4**	**34.9**	**9.4**	**54.7**	**54.5**	**45.8**
RL Training with 8B-Scale Reward Models										
+ DRM	33.2	7.4	49.8	52.7	42.0	30.1	7.0	47.2	51.4	40.2
+ GRM w/ PRC	35.8	7.8	48.3	51.4	41.2	32.3	7.2	47.5	50.6	38.8
+ *Removing Thinking*	35.1	8.0	50.1	52.9	43.6	32.5	7.4	46.5	50.8	41.6
+ GRM w/ RRC-SCR	38.7	9.0	56.6	55.1	46.4	34.6	8.8	53.3	53.6	44.8
+ *voting@8*	40.0	10.4	57.8	**57.3**	47.4	34.8	8.2	54.5	54.8	45.2
+ GRM w/ RRC-AGR	39.4	11.0	58.1	55.4	47.8	**35.2**	9.0	55.1	53.4	45.0
+ *voting@8*	**41.3**	**11.2**	**59.8**	56.9	**48.6**	34.8	**9.8**	**55.8**	**56.2**	**46.8**

Table 1: Performance of models fine-tuned with explicit reasoning (*w/ Thinking*) and without explicit reasoning (*w/o Thinking*). Best results within each group are shown in **bold**. A dotted line separates baselines from our RRC. +*voting@8* is that we use a majority voting of 8 to enhance RRC, together with conflict-aware ranking adjustment in the SCR. 3B-scale and 8B-scale reward models denote that models are trained based on LLaMA-3.2-3B-Instruct and LLaMA-3.1-8B-Instruct, respectively. DRM: Discriminative Reward Model; GRM: Generative Reward Model.

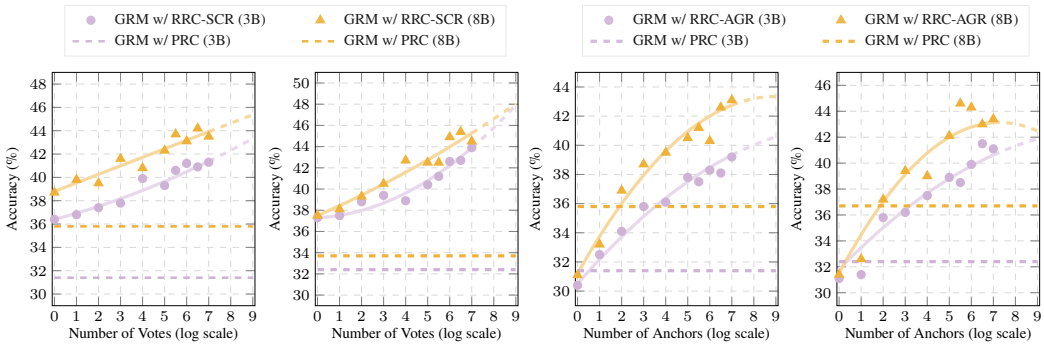
5.1 Experimental Setups

Model Backbones. For reward model training, we employed LLaMA-3.1-8B-Instruct and LLaMA-3.2-3B-Instruct (Grattafiori et al., 2024) as backbone models. For RL training, we used LLaMA-3.1-8B-Instruct as the policy model. We also evaluated RRC on Qwen2.5-7B-Instruct (Team et al., 2024) to validate the effectiveness of RRC, as shown in Table 2 in the Appendix.

Training Datasets. For reward model training, we used the HelpSteer3 (Wang et al., 2025f), which comprises 40.5K labeled preference examples. Each example includes human-written feedback and comparative analysis, which we treated as rationales when training generative reward models. For discriminative reward models, we used the same binary preference labels (*i.e.*, which response is preferred). Still, we did not provide access to the textual rationales to ensure a controlled and fair comparison between the two reward modeling approaches. For RL training, we followed the same supervised fine-tuning (SFT) and RL datasets used in Bhaskar et al. (2025). The SFT dataset consisted of 6K examples, while the RL dataset contained 7.5K examples. These datasets were designed for open-ended RL and covered a diverse set of tasks.

Settings. We trained both discriminative and generative reward model baselines for one epoch using a learning rate of 1e-5 and a batch size of 256. For all RL training runs in our main experiments, we used the GRPO algorithm, with a learning rate of 1e-6 and a batch size of 128, while keeping all other hyperparameters at their default values. To evaluate the robustness of our RRC, we conducted RL training under two policy settings. In the first setting, the policy model generated explicit CoT reasoning before producing the final response (denoted as *reasoning-augmented policy*). In the second setting, the policy model generated only the final response without intermediate CoT reasoning. In the former setting, the generated reasoning paths were not provided to the reward models; only the final responses were evaluated. Additional training details are provided in Appendix B.

Baselines. Our main baseline was the probability-based reward construction approach (denoted as *PRC*), where the predicted probability of the preference token was used as the scalar reward score (Zhang et al., 2025; Wang et al., 2026b). Additionally, we considered a variant that removed explicit reward reasoning to encourage output diversity and improve margin awareness (Wang et al., 2025b)



(a) Scaling the Number of Majority Votes

(b) Scaling the Number of Anchors

Figure 3: Scaling behavior of RRC.

(denoted as *PRC+Removing Reasoning*). We also included a discriminative reward model trained on the same preference dataset as the generative reward models. Furthermore, we compared RRC with several offline preference optimization approaches, including *DPO* (Rafailov et al., 2023) and *SimPO* (Meng et al., 2024), trained on the same preference data.

Evaluation. We focused on three open-ended chat benchmarks: AlpacaEval2 (Li et al., 2023), ArenaHardV2 (Li et al., 2025), and WildBench (Lin et al., 2025), which are commonly used to assess the performance of LLMs trained with open-ended RL. We also evaluated our methods on knowledge and math reasoning benchmarks, including MMLU-Redux (Gema et al., 2025) and MATH-500 (Lightman et al., 2024).

5.2 Main Results

Table 1 reports the performance of models fine-tuned from the LLaMA-3.1-8B-Instruct model. First, comparing DRM and GRM w/ PRC, we observe trends consistent with those in Figure 1: using probability-based reward construction does not effectively translate the ranking advantage of generative reward models into RL. For example, under 3B-scale reward models with thinking, GRM w/ PRC achieves 31.4% on AlpacaEval2 and 7.8% on ArenaHardV2, only marginally better than DRM (30.5%/7.0%). Similar gaps are observed in the non-thinking setting (28.4% vs. 27.6% on AlpacaEval2). Second, across both 3B- and 8B-scale reward models, our proposed RRC approaches consistently outperform all baselines by a clear margin. The results show that constructing rewards from rankings rather than token probabilities is substantially more effective for generative reward models in RL. Third, we observe that the proposed majority voting mechanism further boosts performance. For example, under 3B reward models with thinking, RRC-SCR improves from 36.4% to 37.8% on AlpacaEval2 after applying *voting@8*. This suggests that aggregating multiple stochastic preference judgments helps reduce noise in pairwise comparisons and yields more robust rankings.

We further investigate the performance gain on two different ranking approaches in RRC. From the results, we find that AGR generally achieves stronger or comparable performance, especially when combined with voting. We attribute this advantage to the use of anchors generated from a fixed reference policy, which provides more stable and consistent comparison baselines and avoids coupling reward estimation to the rapidly evolving policy distribution.

6 Analysis

6.1 Scaling Behavior of RRC

We investigate the scaling behavior of RRC along two key dimensions: 1) the number of votes used by the generative reward model, and 2) the number of anchors employed in AGR. All experiments are conducted using 3B-scale reward models, under two different sampling settings (sampling 8 and sampling 16 responses per prompt). The results are summarized in Figure 3. From the results, we see a clear and consistent scaling trend with respect to both the number of votes and the number of anchors. Specifically, increasing the number of votes leads to monotonic improvements in accuracy,

indicating that more reliable preference aggregation from the generative reward model yields better-aligned reward signals and thus more effective policy optimization. Similarly, increasing the number of anchors in AGR consistently improves performance, suggesting that denser anchor comparisons provide finer-grained relative quality estimates, which translate into more informative advantage signals in RL. Notably, the performance gains exhibit diminishing returns as the number of votes or anchors continues to increase, which is characteristic of scaling-law behavior (Kaplan et al., 2020). This shows that while larger comparison budgets improve reward calibration and margin estimation, the marginal benefit gradually decreases beyond a certain scale, indicating a trade-off between computational cost and performance.

From the scaling behavior of the AGR, we can see that a relatively small number of anchors (*e.g.*, 8 and 16) is already sufficient to yield substantial performance gains. Interestingly, when the number of anchors becomes very large (*e.g.*, 256), performance tends to saturate or even slightly degrade. We hypothesize that under large-scale anchor sampling, it becomes increasingly difficult to maintain sufficient diversity among anchor responses, leading to redundant or low-informative comparisons.

7 Related Work

Reward Modeling. Reward models play a central role in RLHF and other alignment strategies, such as direct preference optimization (DPO) and rejection sampling (Lee et al., 2021; Rafailov et al., 2023; Wang et al., 2024b; Zhou et al., 2024). Recent works on improving reward models can be classified into two groups. The first group focused on addressing the scarcity of preference data. For example, several studies constructed large-scale training datasets, either in task-specific settings (Stiennon et al., 2020; Xu et al., 2024) or as general-purpose preference datasets (Cui et al., 2023). Additionally, GRAM (Wang et al., 2025b) and POLAR (Dou et al., 2025) explored the use of unlabeled data for pre-training reward models. The second group explored stronger reward modeling approaches (Coste et al., 2024; Min et al., 2024). A major development in this line of work was the transition from discriminative to generative reward modeling. Building on this shift, recent studies showed that incorporating explicit reward reasoning (*i.e.*, CoT) into generative reward models was crucial for improving reward modeling (Chen et al., 2025; Wang et al., 2026b; Chen et al., 2025; Guo et al., 2025b; Wang et al., 2025e;d).

RL with Reward Models. In RLHF, one straightforward way to construct reward signals is to design task-specific rules, an approach commonly referred to as reinforcement learning with verifiable rewards (RLVR), which has been widely used to improve the reasoning capabilities of LLMs (Guo et al., 2025a; Mu et al., 2025; Mroueh, 2025). While such rule-based rewards can provide accurate supervision, their applicability is typically limited to narrowly defined tasks and does not generalize well to open-ended, real-world tasks (Bhaskar et al., 2025; Liang et al., 2026). As a result, the most widely adopted approach is to train reward models to provide learning signals (Ouyang et al., 2022; Wang et al., 2024a). In practice, this is typically implemented by training discriminative reward models that assign scalar scores to sampled responses. Motivated by the recent success of generative reward models, more recent studies have also explored using generative models in this score-based role (Zhang et al., 2025; Wang et al., 2025b). Different from these approaches, we argue that treating generative reward models solely as scalar score predictors underutilizes their comparative modeling capabilities, which motivates us to propose RRC. Recent efforts, such as Song et al. (2025), have also investigated reference-based reward construction for generative reward models. However, these approaches still focus on deriving scalar rewards from reference comparisons, whereas RRC explores ranking-based reward construction.

8 Conclusions

We proposed Ranking-based Reward Construction (RRC) to better leverage the comparative capabilities of generative reward models in RL. Instead of directly using preference probabilities as scalar rewards, RRC derives learning signals from relative response rankings, better matching the natural comparative formulation of generative reward modeling. We introduced two complementary mechanisms, self-competitive ranking and anchor-guided ranking, to support accurate within-group comparison and scalable reward construction, respectively. Extensive experiments on six open-ended chat and reasoning benchmarks demonstrate that RRC consistently improves RL performance over existing reward construction approaches.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing, readability improvements, and formatting checks during the preparation of this manuscript. They were not used to generate experimental results, derive technical claims, or make decisions about the paper's scientific conclusions. We reviewed all AI-assisted edits and take full responsibility for the final content of the paper.

REFERENCES

Nadja Betzler, Robert Bredereck, and Rolf Niedermeier. Theoretical and empirical evaluation of data reduction for exact kemeny rank aggregation. *Autonomous Agents and Multi-Agent Systems*, 28(5):721–748, 2014.

Adithya Bhaskar, Xi Ye, and Danqi Chen. Language models that think, chat better. *ArXiv preprint*, 2025.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 1952.

Kaiyan Chang, Yonghao Shi, Chenglong Wang, Hang Zhou, Chi Hu, Xiaoqian Liu, Yingfeng Luo, Yuan Ge, Tong Xiao, and JingBo Zhu. Step-level verifier-guided hybrid test-time scaling for large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proc. of EMNLP*, pp. 18462–18477, Suzhou, China, 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. *ArXiv preprint*, 2025.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *Proc. of ICLR*. OpenReview.net, 2024.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. 2023.

Andrew Davenport and Jayant Kalagnanam. A computational study of the kemeny rule for preference aggregation. In *AAAI*, pp. 697–702, 2004.

Shihan Dou, Shichun Liu, Yuming Yang, Yicheng Zou, Yunhua Zhou, Shuhao Xing, Chenhao Huang, Qiming Ge, Haijun Lv, Demin Song, Songyang Gao, Chengqi Lyu, Enyu Zhou, Honglin Guo, Zhiheng Xi, Qipeng Guo, Wenwei Zhang, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, and Kai Chen. Pre-trained policy discriminators are general reward models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Proc. of NeurIPS*, 2025.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *Proc. of COLM*, 2024.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. Are we done with MMLU? In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proc. of NAACL*, pp. 5069–5096, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv preprint*, 2025a.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Reward reasoning model. *ArXiv preprint*, 2025b.

Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Xu, Haoming Jiang, Qingru Zhang, Qin Lu, Xin Liu, Chao Zhang, and Tuo Zhao. Think-rm: Enabling long-horizon reasoning in generative reward models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Proc. of NeurIPS*, 2025.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, Sirui Han, Yike Guo, and Yaodong Yang. PKU-SafeRLHF: Towards multi-level safety alignment for LLMs with human preference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proc. of ACL*, pp. 31983–32016, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Ann Lee, Michael Auli, and Marc'Aurelio Ranzato. Discriminative reranking for neural machine translation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proc. of ACL*, pp. 7250–7264, Online, 2021. Association for Computational Linguistics.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proc. of ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

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

Xiaobo Liang, Haoke Zhang, Juntao Li, Kehai Chen, Qiaoming Zhu, and Min Zhang. Generative reward modeling via synthetic criteria preference learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proc. of ACL*, pp. 26755–26769, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0.

Xiaobo Liang, Wanfu Wang, Qipeng Huang, Yuyang Ding, Zecheng Tang, Yixin Ji, Qianben Chen, Zhe Zhao, Kehai Chen, Juntao Li, and Min Zhang. DUAL RM: Beyond rule-based preference reward modeling via meta-reward. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37281–37296, San Diego, California, United States, 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proc. of ICLR*. OpenReview.net, 2024.

Bill Yuchen Lin, Yuntian Deng, Khyathi Raghavi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In *Proc. of ICLR*. OpenReview.net, 2025.

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 *Proc. of ICLR*. OpenReview.net, 2025.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Proc. of NeurIPS*, 2024.

Do June Min, Veronica Perez-Rosas, Ken Resnicow, and Rada Mihalcea. Dynamic reward adjustment in multi-reward reinforcement learning for counselor reflection generation. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 5437–5449, Torino, Italia, 2024. ELRA and ICCL.

Youssef Mroueh. Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification. *ArXiv preprint*, 2025.

Yongyu Mu, Jiali Zeng, Bei Li, Xinyan Guan, Fandong Meng, Jie Zhou, Tong Xiao, and Jingbo Zhu. Dissecting long reasoning models: An empirical study. *ArXiv preprint*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proc. of EMNLP*, pp. 20275–20321, Suzhou, China, 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. 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. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Proc. of NeurIPS*, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Proc. of NeurIPS*, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, 2024.

Chufan Shi, Yixuan Su, Cheng Yang, Yujiu Yang, and Deng Cai. Specialist or generalist? instruction tuning for specific NLP tasks. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proc. of EMNLP*, pp. 15336–15348, Singapore, 2023. Association for Computational Linguistics.

Narges Simjour. Improved parameterized algorithms for the kemeny aggregation problem. In *International Workshop on Parameterized and Exact Computation*, pp. 312–323. Springer, 2009.

Kefan Song, Amir Moeini, Peng Wang, Lei Gong, Rohan Chandra, Shangtong Zhang, and Yanjun Qi. Reward is enough: Llms are in-context reinforcement learners, 2025.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Proc. of NeurIPS*, 2020.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca A. Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. In *Proc. of ICLR*. OpenReview.net, 2025.

Qwen Team et al. Qwen2 technical report, 2024.

Binghai Wang, Runji Lin, Keming Lu, Le Yu, Zhenru Zhang, Fei Huang, Chujie Zheng, Kai Dang, Yang Fan, Xingzhang Ren, et al. Worldpm: Scaling human preference modeling, 2025a.

Binghai Wang, Yantao Liu, Yuxuan Liu, Tianyi Tang, Shenzhi Wang, Chang Gao, Chujie Zheng, Yichang Zhang, Le Yu, Shixuan Liu, Tao Gui, Qi Zhang, Xuanjing Huang, Bowen Yu, Fei Huang, and Junyang Lin. Outcome accuracy is not enough: Aligning the reasoning process of reward models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41482–41508, San Diego, California, United States, 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6.

Chenglong Wang, Hang Zhou, Kaiyan Chang, Bei Li, Yongyu Mu, Tong Xiao, Tongran Liu, and JingBo Zhu. Hybrid alignment training for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proc. of Findings*, pp. 11389–11403, Bangkok, Thailand, 2024a. Association for Computational Linguistics.

Chenglong Wang, Hang Zhou, Yimin Hu, Yifu Huo, Bei Li, Tongran Liu, Tong Xiao, and Jingbo Zhu. ESRL: efficient sampling-based reinforcement learning for sequence generation. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19107–19115. AAAI Press, 2024b.

Chenglong Wang, Yang Gan, Yifu Huo, Yongyu Mu, Qiaozhi He, Murun Yang, Bei Li, Tong Xiao, Chunliang Zhang, Tongran Liu, and JingBo Zhu. GRAM: A generative foundation reward model for reward generalization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proc. of ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b.

Chenglong Wang, Yang Gan, Hang Zhou, Chi Hu, Yongyu Mu, Kai Song, Murun Yang, Bei Li, Chunliang Zhang, Tongran Liu, JingBo Zhu, Zhengtao Yu, and Tong Xiao. MRO: enhancing reasoning in diffusion language models via multi-reward optimization. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Proc. of NeurIPS*, 2025c.

Chenglong Wang, Yongyu Mu, Hang Zhou, Yifu Huo, Ziming Zhu, Jiali Zeng, Murun Yang, Bei Li, Xiaoyang Hao, Chunliang Zhang, Fandong Meng, Jingbo Zhu, and Tong Xiao. Gram-r^2: Self-training generative foundation reward models for reward reasoning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 33395–33403. AAAI Press, 2026b.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proc. of ACL*, pp. 9440–9450, Bangkok, Thailand, 2024c. Association for Computational Linguistics.

Yibin Wang, Zhimin Li, Yuhang Zang, Chunyu Wang, Qinglin Lu, Cheng Jin, and Jiaqi Wang. Unified multimodal chain-of-thought reward model through reinforcement fine-tuning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Proc. of NeurIPS*, 2025d.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation, 2025e.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Proc. of NeurIPS*, 2025f.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Proc. of NeurIPS*, 2022.

Tong Xiao and Jingbo Zhu. Foundations of large language models. *ArXiv preprint*, 2025.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Proc. of ICML*. OpenReview.net, 2024.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng-Ann Heng, and Wai Lam. Unveiling the generalization power of fine-tuned large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 884–899, Mexico City, Mexico, 2024. Association for Computational Linguistics.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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 Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Proc. of NeurIPS*, 2025.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *ArXiv preprint*, 2023.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *Proc. of ICLR*. OpenReview.net, 2025.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. Genprm: Scaling test-time compute of process reward models via generative reasoning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 34932–34940. AAAI Press, 2026.

Hang Zhou, Chenglong Wang, Yimin Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. Prior constraints-based reward model training for aligning large language models. In Sun Maosong, Liang Jiye, Han Xianpei, Liu Zhiyuan, and He Yulan (eds.), *Proceedings of the 23rd Chinese National Conference on Computational Linguistics (Volume 1: Main Conference)*, pp. 1395–1407, Taiyuan, China, 2024. Chinese Information Processing Society of China.

A PROOFS FOR THEORETICAL RESULTS

In this section, we present proofs and additional analysis for the theoretical results used in the main paper. Property 1 follows directly from the monotonic reward assignment induced by response rankings. We therefore focus on Property 2, which establishes the importance of margin awareness in reward construction for RL, and Property 3, which shows that Kemeny-rule-based aggregation can recover a globally consistent ranking that maximizes the weighted agreement with all pairwise preferences. We also provide an additional proposition explaining the advantage of using a reference policy to generate anchor responses.

Property 2. *The reward difference should reflect the relative quality gap between responses: $r(x, o_i) - r(x, o_j) \propto \Delta q_{ij}$, where Δq_{ij} denotes the latent oracle preference margin between responses o_i and o_j.*

Proof: For any two responses o_i and o_j sampled under the input x, the reward gap should reflect the underlying quality gap:

$$r(x, o_i) - r(x, o_j) \;\propto\; \Delta q_{ij}, \qquad \Delta q_{ij} \triangleq q(x, o_i) - q(x, o_j) \tag{8}$$

where $q(x, o)$ denotes the latent (oracle) preference quality, and Δq_{ij} is the latent oracle preference margin. Assume there exists an oracle quality function $q(x, o)$ that characterizes the true preference intensity for response o given input x. The ideal RL objective is

$$J_q(\theta) \;\triangleq\; \mathbb{E}_{x\sim\mathcal{D}, o\sim\pi_\theta(\cdot|x)}\big[q(x, o)\big] \tag{9}$$

By the policy gradient theorem, its gradient can be written as

$$\nabla_\theta J_q(\theta) = \mathbb{E}_{x,o}\Big[q(x, o)\nabla_\theta \log \pi_\theta(o|x)\Big] \tag{10}$$

In practice, GRPO-style RL algorithms typically optimize a surrogate objective using a *group* of sampled responses $\{o_1, \cdots, o_m\}$ for the same x, and define a normalized group-relative advantage:

$$A_i^{(r)} = \frac{r_i - \mu_r}{\sigma_r}, \qquad r_i \triangleq r(x, o_i), \quad \mu_r \triangleq \frac{1}{m}\sum_{k=1}^{m} r_k, \quad \sigma_r^2 \triangleq \frac{1}{m}\sum_{k=1}^{m}(r_k - \mu_r)^2 \tag{11}$$

Ignoring clipping and other stabilizers for clarity, the corresponding RL update direction is

$$g_r(\theta) \;\triangleq \mathbb{E}_x\left[\sum_{i=1}^{m} A_i^{(r)}\nabla_\theta \log \pi_\theta(o_i|x)\right] \tag{12}$$

Hence, the relative magnitudes of policy updates across sampled responses are governed by the normalized reward gaps $A_i^{(r)}$. This demonstrates that not only the ordering of rewards, but also the proper calibration of reward differences, is crucial for effective optimization.

Furthermore, assume that for any fixed x, reward scores from reward models are an affine transformation of oracle qualities:

$$r(x, o) = bq(x, o) + d, \qquad b > 0 \tag{13}$$

Then the advantage computed from r coincides with the advantage computed from q:

$$A_i^{(r)} = A_i^{(q)} \;\triangleq\; \frac{q_i - \mu_q}{\sigma_q}, \quad q_i \triangleq q(x, o_i), \quad \mu_q \triangleq \frac{1}{m}\sum_{k=1}^{m} q_k, \quad \sigma_q^2 \triangleq \frac{1}{m}\sum_{k=1}^{m}(q_k - \mu_q)^2 \tag{14}$$

Under the affine assumption,

$$\mu_r = \frac{1}{m}\sum_{k=1}^{m}(bq_k + d) = b\mu_q + d \tag{15}$$

and

$$\sigma_r^2 = \frac{1}{m}\sum_{k=1}^{m}\big(bq_k + d - (b\mu_q + d)\big)^2 = \frac{1}{m}\sum_{k=1}^{m}(b(q_k - \mu_q))^2 = b^2\sigma_q^2 \tag{16}$$

so $\sigma_r = b\sigma_q$ since $b > 0$. Therefore,

$$A_i^{(r)} = \frac{bq_i + d - (b\mu_q + d)}{b\sigma_q} = \frac{q_i - \mu_q}{\sigma_q} = A_i^{(q)} \tag{17}$$

Based on this equation, the RL update direction induced by r becomes

$$g_r(\theta) = \mathbb{E}_x \left[\sum_{i=1}^{m} A_i^{(q)} \nabla_\theta \log \pi_\theta(o_i|x) \right] \tag{18}$$

which is precisely the group-normalized counterpart of the oracle policy gradient. In particular, the *relative weighting* assigned to different sampled responses is governed by $(q_i - \mu_q)$, *i.e.*, the oracle quality margins within the sampled group. Hence, to maintain oracle-consistent update geometry, reward differences must scale proportionally with oracle differences:

$$r_i - r_j = b(q_i - q_j) = b\Delta q_{ij} \tag{19}$$

This is exactly the desired margin-awareness property:

$$r(x, o_i) - r(x, o_j) \;\propto\; \Delta q_{ij} \tag{20}$$

Building on the above proof, we further analyze how our SCR and AGR mechanisms satisfy the property of margin awareness. In SCR, reward scores are constructed from pairwise comparisons among sampled responses:

$$r(x, o_i) = \alpha \sum_{j \neq i} \mathbf{1}[o_i \succ o_j] \tag{21}$$

Thus, the reward difference between the two responses satisfies

$$r_i - r_j = \alpha \sum_{k} \big(\mathbf{1}[o_i \succ o_k] - \mathbf{1}[o_j \succ o_k] \big) \tag{22}$$

This difference equals the net number of wins of o_i over the rest of the group relative to o_j. Intuitively, if o_i is substantially better than o_j, it will defeat more competitors and lose to fewer, yielding a larger reward gap. Conversely, if two responses are close in quality, their win counts will also be similar.

In AGR, rewards are constructed via comparisons with a fixed set of anchor responses:

$$r(x, o_i) = \alpha \sum_{k=1}^{n} \mathbf{1}[o_i \succ a_k] \tag{23}$$

where anchors a_k are generated from a reference policy. Here, the reward difference becomes

$$r_i - r_j = \alpha \sum_{k=1}^{n} \big(\mathbf{1}[o_i \succ a_k] - \mathbf{1}[o_j \succ a_k] \big) \tag{24}$$

which measures how much better o_i performs than o_j relative to the same quality benchmarks. If o_i consistently outperforms anchors that o_j fails against, the reward gap increases accordingly. This mechanism effectively estimates the relative position of each response along a shared quality scale defined by the anchors. Thus, reward differences reflect how much higher one response ranks than another with respect to common reference points, providing a stable and scalable approximation of oracle margin differences.

This completes the proof of this property.

Property 3. *Given inconsistent pairwise preferences produced by a generative reward model, a Kemeny-rule-based aggregation can recover a globally consistent ranking that maximizes the weighted agreement with all pairwise preferences, where the weight of each preference is determined by its support degree (e.g., voting count) from the model.*

Proof: Let $O = \{o_1, \ldots, o_m\}$ be the response set. Suppose the generative reward model outputs (possibly inconsistent) pairwise preferences with nonnegative weights $w_{ij}{}_{i \neq j}$, where w_{ij} represents

the support for the preference $o_i \succ o_j$ (*e.g.*, the number of votes in favor of $o_i \succ o_j$). This defines a weighted directed graph $G = (V, E)$ with $V = O$ and directed edges $i \to j$ of weight w_{ij}. A globally consistent ranking is a total order π over O. Any total order π induces a transitive tournament orientation: we say π agrees with an edge $i \to j$ if i is ranked ahead of j under π (denoted $i \prec_\pi j$ or equivalently $o_i \succ_\pi o_j$). Define the weighted agreement of π with the pairwise preferences as

$$\mathrm{Agree}(\pi) := \sum_{i \prec_\pi j} w_{ij} \tag{25}$$

i.e., the total weight of directed preferences that are consistent with the total order π. Equivalently, define the weighted disagreement (*a.k.a*, feedback weight) as

$$\mathrm{Disagree}(\pi) := \sum_{i \prec_\pi j} w_{ji} \tag{26}$$

which corresponds to the total weight of edges that point against the ordering induced by π. Note that for any unordered pair i, j, exactly one of $i \prec_\pi j$ or $j \prec_\pi i$ must hold. Hence,

$$\mathrm{Agree}(\pi) + \mathrm{Disagree}(\pi) = \sum_{i<j} \big(w_{ij} + w_{ji}\big) =: C \tag{27}$$

where C is a constant independent of π. Therefore,

$$\arg\max_\pi \mathrm{Agree}(\pi) = \arg\min_\pi \mathrm{Disagree}(\pi) \tag{28}$$

This is precisely the Kemeny optimal aggregation objective in the weighted setting: find the total order π^* that minimizes the total weight of pairwise disagreements (Simjour, 2009; Betzler et al., 2014). Finally, because π^* is a total order, it is globally consistent by construction: if $i \prec_{\pi^*} j$ and $j \prec_{\pi^*} k$, then necessarily $i \prec_{\pi^*} k$. Thus, the Kemeny-rule solution π^* yields a globally consistent ranking that maximizes weighted agreement with the pairwise preferences, with weights determined by support degrees (*e.g.*, voting counts).

This completes the proof of this property.

Proposition 1. *Compared to selecting anchors from samples of the current policy $\pi_\theta(\cdot)$, using a reference policy $\pi_{\mathrm{ref}}(\cdot)$ to generate anchor responses decouples anchor quality from policy updates. This mitigates feedback loops between policy improvement and anchor quality, reducing distributional drift and yielding more stable reward signals during training.*

Proof: In anchor-guided ranking (see Section 4.2), the reward for a sampled response $o_i \sim \pi_\theta(\cdot|x)$ is constructed as

$$r_\theta(x, o_i) = \alpha \sum_{k=1}^{n} \mathbf{1}[o_i \succ a_k], \quad a_k \sim q_\theta(\cdot|x) \tag{29}$$

where q_θ denotes the anchor-generating distribution. Here, we consider two strategies for generating anchors: 1) On-policy anchors: $q_\theta(\cdot|x) = \pi_\theta(\cdot|x)$ and 2) Reference anchors: $q_\theta(\cdot|x) = \pi_{\mathrm{ref}}(\cdot|x)$, independent of θ.

Here, we call the gradient of the RL optimization objective *w.r.t* θ:

$$\nabla_\theta J(\theta) = \underbrace{\mathbb{E}_{x,o}\left[r_\theta(x, o) \nabla_\theta \log \pi_\theta(o|x)\right]}_{\text{policy improvement term}} + \underbrace{\mathbb{E}_{x,o}\left[\nabla_\theta r_\theta(x, o)\right]}_{\text{baseline drift term}} \tag{30}$$

The second term is non-zero because anchors are also drawn from π_θ:

$$\nabla_\theta r_\theta(x, o) = \alpha \sum_{k=1}^{n} \mathbb{E}_{a_k \sim \pi_\theta}\big[\mathbf{1}[o \succ a_k] \nabla_\theta \log \pi_\theta(a_k|x)\big] \tag{31}$$

Therefore, each policy update simultaneously: 1) shifts probability mass toward better responses o, and 2) improves the anchor set used as the comparison baseline. As a result, the reward assigned to a

Method	*w/ Thinking*					*w/o Thinking*				
	AE2	AH2	WiB	MM_R	MATH	AE2	AH2	WiB	MM_R	MATH
SFT (*w/ Qwen2.5-7B-Instruct*)	27.1	5.6	44.6	60.3	45.6	25.6	5.2	46.2	62.7	47.8
+ DPO	29.6	6.2	48.4	58.4	42.4	28.7	7.6	47.8	60.4	45.6
+ SimPO	30.4	7.0	50.3	56.7	45.8	29.1	7.8	48.9	58.2	46.2
RL Training via 3B-Scale Reward Models										
+ DRM	27.8	6.8	46.6	58.2	46.8	26.8	7.4	46.0	59.1	48.8
+ GRM w/ PRC	30.9	7.2	48.3	56.4	48.8	28.1	7.2	47.8	60.4	48.0
+ *Removing Thinking*	28.4	6.4	46.6	55.3	45.2	27.6	6.8	46.8	61.7	48.4
+ GRM w/ RRC-SCR	33.2	7.8	51.2	58.4	48.0	**34.8**	7.6	48.6	62.2	49.8
+ *voting@8*	35.6	8.4	**52.8**	60.5	50.0	35.7	8.2	49.2	64.7	48.6
+ GRM w/ RRC-AGR	34.5	8.6	52.8	62.6	51.8	32.8	8.0	49.6	65.7	51.4
+ *voting@8*	**35.8**	**9.4**	52.4	**63.4**	**52.6**	33.8	**8.6**	**50.6**	**67.6**	**52.8**

Table 2: Performance of models fine-tuned from Qwen2.5-7B-Instruct model. All experiments are conducted using 3B-scale reward models.

fixed response o may decrease even if its absolute quality does not, simply because anchors become stronger. Hence, the optimization target itself changes with θ, forming a moving-target objective:

$$\theta_t \Rightarrow \pi_{\theta_t} \Rightarrow \text{stronger anchors} \Rightarrow \text{shifted rewards} \Rightarrow \nabla_\theta J(\theta_t)\text{changes} \tag{32}$$

This violates the stationarity assumption implicitly required by stochastic gradient ascent and increases both gradient variance and training instability.

When anchors are sampled from a fixed reference policy,

$$r(x, o) = \alpha \sum_{k=1}^{n} \mathbf{1}[o \succ a_k], \quad a_k \sim \pi_{\text{ref}} \tag{33}$$

which is independent of θ. Thus,

$$\nabla_\theta r(x, o) = 0 \tag{34}$$

and the gradient reduces to

$$\nabla_\theta J(\theta) = \mathbb{E}\left[r(x, o)\nabla_\theta \log \pi_\theta(o|x)\right] \tag{35}$$

which corresponds to optimizing a fixed reward landscape. In this case, policy updates only affect the sampling distribution of candidate responses, while the evaluation baseline remains stationary. Hence, the optimization objective does not drift across iterations, satisfying the standard assumptions of stochastic optimization and yielding more stable training dynamics.

This completes the proof of this property.

B Additional Experimental Details and Results

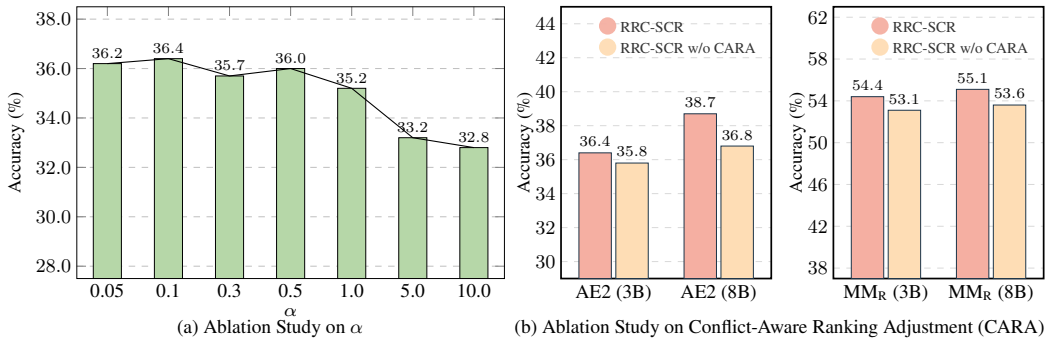
In this section, we provide additional experimental details and present the experimental results of our RRC approach.

B.1 Experimental Details

Training Setups. For the experiments reported in Figure 1 and Table 1, we set the group sampling size to $m = 8$ and the number of anchor responses to $n = 8$. We also evaluated other group and anchor sizes in Figure 3 to study the impact of these hyperparameters. For RRC, we set the scaling factor α to 0.1. Following Bhaskar et al. (2025), we additionally incorporated a format reward to encourage structured outputs in the reasoning-augmented setting. The final reward is computed as a combination of the ranking-based reward and the format reward:

$$r(o, x) = r_\phi(o, x) + r_{\text{format}}(o) \tag{36}$$

Here, we define the format reward $r_{\text{format}}(\cdot)$ as a binary indicator: it equals 1 if the sampled response follows the required structure with explicit `<think>` and `<answer>` tags, and 0 otherwise. When



(a) Ablation Study on α (b) Ablation Study on Conflict-Aware Ranking Adjustment (CARA)

Figure 4: Ablation Study

querying the generative reward model, we account for the positional bias reported in Wang et al. (2024c) by randomly assigning responses to the "Response A" and "Response B" slots in each pairwise comparison. Additionally, during RL training, we saved a checkpoint every 100 steps. We then selected the best checkpoint based on performance on a validation set. The validation set was sampled from the training distribution and consists of 500 examples. For validation, we used `GPT-4o` to generate reference responses and evaluated model outputs using the AlpacaEval2 evaluation protocol to compute validation scores. The checkpoint with the highest validation score was selected as the final model.

Evaluation. For AlpacaEval2 and ArenaHardV2, we adopted `GPT-4-1106-preview` as the reference model to generate baseline responses paired with each evaluation prompt. AlpacaEval2 consists of 805 user prompts. Evaluation was conducted in a head-to-head setting, where model outputs were compared against the reference responses and judged by a generative evaluator, resulting in win rates ranging from 0% to 100%. We followed the length-controlled win-rate protocol recommended by Dubois et al. (2024) to mitigate length bias. In our implementation, we replaced the default GPT-4 judge with `GPT-4o`. Notably, we further employed a stronger `GPT-5` as the final evaluator to ensure robust and reliable assessment. ArenaHardV2 comprises 500 challenging real-world user queries, and we employed the same evaluation protocol as AlpacaEval2. For WildBench, which contains 1,024 user prompts with some multi-turn interactions, we followed the official evaluation protocol and used point-wise scoring instead of head-to-head comparison[1]. The final scores are reported on a 0-100 scale. For the reasoning benchmarks, we used `evalscope` to conduct automatic evaluation[2].

B.2 Ablation Study

We conduct a comprehensive ablation study to investigate the sensitivity of our method to the reward scaling factor α in SCR. Specifically, as shown in Figure 4(a), we vary α over a wide range from 0.05 to 10.0, and evaluate performance using SCR with 3B-scale reward models. The results indicate that the performance remains relatively stable when α lies in the range of 0.05 to 0.5, suggesting that our method is not highly sensitive to moderate changes in reward scaling. However, when α increases beyond 1.0, we observe a gradual performance degradation. This behavior can be attributed to the normalization mechanism in GRPO-style RL. Since advantages are normalized within each batch, moderate rescaling of reward values does not significantly affect the relative magnitudes of policy gradients. In contrast, overly large scaling factors may amplify noise and lead to unstable advantage normalization, which in turn degrades training stability and final performance.

We further investigate the effectiveness of Conflict-Aware Ranking Adjustment (namely CARA) in SCR. As shown in Figure 4(b), removing CARA consistently leads to performance drops across different backbone scales and benchmarks. Specifically, on AE2, SCR with CARA outperforms the variant without CARA by 0.6% and 1.9% on the 3B and 8B backbones, respectively. Similarly,

[1] `https://github.com/allenai/WildBench/blob/main/evaluation/eval_template.score.v2.md`

[2] `https://github.com/modelscope/evalscope`

Dimension	SCR	AGR
Structure	Fully Connected	Bipartite Graph
Preference Query	$\mathcal{O}(m^2)$	$\mathcal{O}(m \cdot n)$
Scalability	Limited	Better

Table 3: Comparison between SCR and AGR in terms of comparison structure, preference query complexity, and scalability.

on MM_R, we observe improvements of 1.3% (3B) and 1.5% (8B) when CARA is enabled. These results confirm that CARA plays an important role in mitigating the adverse effects of conflicting pairwise preferences produced by generative reward models.

B.3 Results on Additional Backbone Models

To evaluate the generality of our approach across different policy backbones, we further conduct experiments on Qwen2.5-7B-Instruct. As shown in Table 2, we observe trends highly consistent with those on Qwen2.5-7B-Instruct. In particular, RRC-based methods, including both SCR and AGR variants, consistently outperform DPO, SimPO, and our RL baselines, demonstrating that the effectiveness of RRC is not tied to a specific backbone architecture or model scale. An interesting observation is that removing explicit `<think>` traces (*i.e.*, *w/o Thinking*) does not lead to noticeable performance degradation on certain benchmarks, especially on MM_R and MATH. In some cases, the performance is even slightly improved. We hypothesize that this phenomenon is related to the data distribution used during SFT, where a large portion of reasoning-related samples are embedded within open-ended chat formats rather than strictly structured chain-of-thought annotations. Similar findings have also been reported in Bhaskar et al. (2025).

C Additional Analyses

C.1 Behavioral Differences Between SCR and AGR

As summarized in Table 3, SCR and AGR mainly differ in their comparison structures and in the number of preference queries to the generative reward model. SCR performs pairwise comparisons among all sampled responses, which induces a fully connected tournament over the response set. In contrast, AGR compares each response only against a set of anchors, forming a bipartite graph between responses and anchors. This structural difference leads to substantially different computational costs. We further compare the number of queries to the generative reward model required by the two approaches. SCR requires pairwise comparisons between all response pairs, resulting in

$$N_{\text{SCR}} = \frac{m(m-1)}{2} = \mathcal{O}(m^2) \tag{37}$$

while AGR only performs response–anchor comparisons,

$$N_{\text{AGR}} = m \cdot n = \mathcal{O}(mn) \tag{38}$$

The difference in query cost is therefore

$$N_{\text{SCR}} - N_{\text{AGR}} = m\big(\tfrac{m-1}{2} - n\big) \tag{39}$$

which is positive whenever $n < \frac{m-1}{2}$, a typical setting in practice. Up to constant factors, this gap scales on the order of $m(m-n)$ when $n < m$, highlighting the improved scalability of AGR under larger sampling budgets. We also provide the corresponding ranking algorithms in Algorithms 1 and 2.

C.2 Performance vs. Time Cost

We further analyze the computational overhead introduced by ranking-based rewards and study the trade-off between performance gains and training efficiency.

Efficiency Optimization. To reduce the practical overhead of generative reward models, we optimize reward computation from two aspects. First, we deploy generative reward models in parallel

Method	Training Time	AlpacaEval2	Gain
Discriminative RM	8.2h	33.2	-
Baseline RL (GRM w/ PRC)	10.5h	35.8	-
RRC-SCR	13.2h	38.7	+2.9
RRC-AGR	10.6h	39.4	+3.6
RRC-SCR + voting@8	15.2h	40.0	+4.2
RRC-AGR + voting@8	11.8h	41.3	+5.5

Table 4: Comparison of end-to-end training cost and performance on AlpacaEval2 (8B-scale models with Thinking). Improvements are computed over the baseline RL pipeline.

Method	AE2	WiB	MMR	MATH
GRM w/ PRC + Majority Voting	32.8	52.1	52.8	45.2
GRM w/ PRC + Soft Aggregation	30.1	51.4	48.2	44.8
GRM w/ RRC-SCR	**36.4**	55.6	54.4	45.6
GRM w/ RRC-AGR	35.8	**56.8**	**55.0**	**46.4**

Table 5: Compute-matched comparison between PRC and RRC (3B-scale models with Thinking). All compared methods use the same number of GRM queries as RRC under the same sampling configuration. Note that discriminative reward models are not included because they do not support the same computation scaling mechanism based on repeated generative preference judgments.

using `vLLM`, where 8 GPUs are used to host 8 model instances for concurrent preference evaluation. Second, we adopt asynchronous reward querying to overlap reward computation with policy optimization. With a sampling size of 8, the additional ranking computation in SCR takes approximately 0.5ms per batch, compared with 0.2ms for discriminative reward models under the same setting. This demonstrates that the overhead introduced by ranking operations remains small at the reward computation level. For AGR, we further improve efficiency by precomputing anchor responses. Specifically, anchor generation requires approximately 1 hour using 16 GPUs. After anchor construction, AGR training takes around 9 hours, leading to an overall training cost comparable to the baseline RL pipeline.

Performance-Efficiency Analysis. Table 4 summarizes the end-to-end training cost and performance improvements under a fixed hardware budget. Compared with the baseline RL pipeline (GRM w/ PRC), SCR increases the training time from 10.5h to 13.2h, while improving AlpacaEval2 performance from 35.8% to 38.7%. When combined with voting@8, SCR further improves performance to 40.0% with a moderate increase in training cost. In contrast, AGR provides a more favorable performance-efficiency trade-off by leveraging a small set of precomputed anchor responses. With nearly identical training time to the baseline RL pipeline (10.6h vs. 10.5h), AGR improves AlpacaEval2 from 35.8% to 39.4%. Moreover, AGR with voting@8 achieves the best performance of 41.3 with only a modest increase in training cost. These results demonstrate that ranking-based reward construction can effectively improve RL training with generative reward models while maintaining practical efficiency.

Compute-Matched Comparison. To investigate whether the improvements of RRC mainly come from additional reward model computation, we conduct a compute-matched comparison with a strengthened PRC baseline. Specifically, we replace majority voting in GRM w/ PRC with soft aggregation, where preference probabilities from multiple CoT-enabled GRM queries are averaged to construct continuous rewards. This variant uses a comparable number of GRM evaluations while providing a finer-grained reward signal. As shown in Table 5, soft aggregation does not consistently improve over standard majority voting and remains inferior to RRC across all evaluated benchmarks. For example, on AlpacaEval2, the soft aggregation can achieve 30.1%, compared with 36.4% from RRC-SCR and 35.8% from RRC-AGR. These results suggest that allocating additional inference computation alone is insufficient. The improvement mainly comes from how comparison signals are aggregated: RRC preserves global preference structures and relative margins, whereas score aggregation treats each comparison independently.

Probability Range	[0,0.1]	[0.1,0.9]	[0.9,1.0]
Percentage (%)	42.3	11.7	46.0

Table 6: Distribution of preference token probabilities from a point-wise generative reward model trained on HelpSteer3.

C.3 Point-wise Generative Reward Models

In this subsection, we further investigate point-wise generative reward models, which directly optimize scalar preference prediction. Specifically, given an input-response pair (x, y), the model is trained to generate a preference label z indicating the quality of the response (*e.g.*, 1, 2, and 3). The training objective is formulated as:

$$\mathcal{L}_{\text{point}} = -\mathbb{E}_{(x,y,z)\sim\mathcal{D}} \log \pi_\theta(z|x, y) \tag{40}$$

During inference, we use the probability of the positive preference token as the scalar reward:

$$r(x, y) = \pi_\theta(z^+|x, y) \tag{41}$$

where z^+ denotes the token corresponding to a positive preference. Although point-wise generative reward models can directly produce scalar preference scores, we find that the resulting reward signals suffer from limited resolution.

To analyze this issue, we train a point-wise generative reward model on the HelpSteer3 dataset and examine the distribution of the final preference label token probabilities under CoT-based generation. As shown in Table 6, the predicted probabilities are highly concentrated near extreme values, with only a small fraction of samples falling into the middle range. This phenomenon indicates scalar compression, where the reward model tends to produce near-deterministic preference predictions and provides limited resolution for distinguishing responses with similar quality. We attribute this behavior to the fact that CoT generation often determines the final preference token, resulting in saturated probability distributions. In contrast, RRC constructs rewards from structured comparisons among responses, explicitly preserving relative ranking and margin information even when individual preference probabilities become saturated. Therefore, the advantage of RRC does not simply come from producing finer-grained scalar rewards, but from better leveraging the comparative nature of generative reward models to provide more informative learning signals for RL. Importantly, the limitation is not caused by the use of token probabilities itself, but by using them as absolute rewards. Even when the underlying preference prediction is accurate, saturated probabilities discard relative differences among responses. RRC avoids this issue by aggregating multiple comparative judgments into structured rankings.

Algorithm 1: Self-Competitive Ranking (SCR) with Majority Voting and CARA

Require: Prompt x; responses $O = \{o_1, \ldots, o_m\}$; generative reward model GRM; voting budget V; scaling factor α; flag `use_CARA`
Ensure: Rewards $\{r(x, o_i)\}_{i=1}^{m}$ and total order π

Initialize $w_{ij} \leftarrow 0$ for all $i \neq j$
for each unordered pair $\{i, j\}$ with $i < j$ **do**
  $c_{i \succ j} \leftarrow 0$, $c_{j \succ i} \leftarrow 0$
  for $t = 1$ to V **do**
    Query GRM on (x, o_i, o_j) and obtain preference
    if $o_i \succ o_j$ **then**
      $c_{i \succ j} \leftarrow c_{i \succ j} + 1$
    else
      $c_{j \succ i} \leftarrow c_{j \succ i} + 1$
    end if
  end for
  if $c_{i \succ j} \geq c_{j \succ i}$ **then**
    $w_{ij} \leftarrow c_{i \succ j}$, $w_{ji} \leftarrow c_{j \succ i}$
  else
    $w_{ji} \leftarrow c_{j \succ i}$, $w_{ij} \leftarrow c_{i \succ j}$
  end if
end for
if `use_CARA` = 1 **then**
  Initialize partial order Ω with an empty set
  while there exists an undecided pair (i, j) **do**
    Select (i, j) with the largest $|w_{ij} - w_{ji}|$
    if $w_{ij} \geq w_{ji}$ **then**
      Add relation $i \prec j$ into Ω
    else
      Add relation $j \prec i$ into Ω
    end if
    Enforce transitive closure and discard relations forming cycles
  end while
  Obtain total order π by topological sorting of Ω
else
  Sort responses by win counts $\sum_{j \neq i} \mathbf{1}[w_{ij} > w_{ji}]$ to obtain π
end if
for each $o_i \in O$ **do**
  $r(x, o_i) \leftarrow \alpha \cdot |\{o_j \neq o_i : i \prec_\pi j\}|$
end for
return $\{r(x, o_i)\}_{i=1}^{m}$, π

Algorithm 2: Anchor-Guided Ranking (AGR) with Majority Voting

Require: Prompt x; responses $O = \{o_1, \ldots, o_m\}$; anchors $A = \{a_1, \ldots, a_n\}$; generative reward model GRM; voting budget V; scaling factor α
Ensure: Rewards $\{r(x, o_i)\}_{i=1}^{m}$

for each response $o_i \in O$ **do**
  $s_i \leftarrow 0$
  for each anchor $a_k \in A$ **do**
    $c_{i\succ k} \leftarrow 0$, $c_{k\succ i} \leftarrow 0$
    for $t = 1$ to V **do**
      Query GRM on (x, o_i, a_k) and obtain preference
      if $o_i \succ a_k$ **then**
        $c_{i\succ k} \leftarrow c_{i\succ k} + 1$
      else
        $c_{k\succ i} \leftarrow c_{k\succ i} + 1$
      end if
    end for
    if $c_{i\succ k} \geq c_{k\succ i}$ **then**
      $s_i \leftarrow s_i + 1$
    end if
  end for
  $r(x, o_i) \leftarrow \alpha \cdot s_i$
end for
Rank responses by descending $r(x, o_i)$
return $\{r(x, o_i)\}_{i=1}^{m}$

Prompt	Write a rock song, with alternating rhymes [ABAB], about the following ideas below. Avoid common rhymes, jargons, expressions like "neon lights", "shadows creep", "whispers", "air is thick", "bittersweet", "forevermore", "story to be told", etc.\n——\n Compassion is the ancient stone bridge, worn smooth by centuries of rain and footsteps, that stretches across the chasm of our own pain. Each stone in its foundation is a memory, a tear, a whispered prayer–each one a testament to the moments when we have stumbled, when we have felt the sharp edges of loss and longing cut into our hearts. And yet, it is this very bridge, this fragile yet enduring connection, that allows us to reach out to others, to understand the weight of their burdens, to offer a hand in the darkness. The bridge does not erase the chasm, but it makes it possible to cross, to find solace in the shared journey, to know that we are not alone in our suffering, and that, together, we can find our way to the other side.
Baseline (PRC)	<think>\nThe user wants a rock song with alternating rhymes (ABAB) about the concept of "compassion" as described in the provided text.\n\nKey ideas to incorporate:\n1. **Compassion as a bridge:** Connecting people across pain.\n2. **The bridge is ancient, worn:** Smooth, centuries old.\n3. ....., showing we care\nA fragile, enduring span, that helps us find our way there\nThrough the chasm of our suffering, we cross, we cross, we share\n\n**Verse 2:**\n(Guitar: Adds texture, maybe a bridge riff)\nEach stone in its foundation tells a story of the fall\nOf hopes that shattered like thin ice, of prayers that echoed through the wall\nOf moments when the weight was crushing, the edges cut through all\nMemories etched into the surface, testifying to how we stood at all\n\n**Chorus:**\n(Reprise with energy)\nThis is the weight of the bridge, the weight that we bear\nConnecting the sharp edges of our hearts, showing we care\nA fragile, enduring span, that helps us find our way there\nThrough the chasm of our suffering, we cross, we cross, we share\n\n**Bridge:**\n(Different rhythm or guitar tone.... "connection worn so smooth," "span," "cross."\n * **Ancient, worn:** "ancient, weathered path," "weathered," "smooth."\n * **Stones are memories/tears/prayers:** "each stone in its foundation tells a story," "memories etched into the surface," "tears that fell behind," "prayers that echoed."\n * **Crossing the chasm:** "connecting the sharp edges of our hearts," "find our way there," "helps us find our way," "cross."\n * **It makes it possible, not erases:** "helps us find our way," "it doesn't mend the cracks," "it provides a hand," "together, we make the journey," "until the dawn is won," "shared weight."\n* **Imagery & Metaphor:** The song uses strong, evocative images like "weariness," "light is hard to find," "weathered path," "sharp edges," "stones," "tears," "prayers," "chasm," "weight," "span," "journey," "dawn." These help paint a vivid picture of the struggle and the connection.\n* **Rock Song Feel:** The lyrics are written to be sung with energy, especially in the choruses and bridge sections. The language is strong and direct, fitting a rock music style.\n</answer>"
Ours (SCR)	<think>\nThis prompt is not simply asking for a song about kindness or emotional support.\nIt establishes a **very specific philosophical structure**:\n\n* Compassion is not a feeling, but an **infrastructure**.\n* Pain is not removed, only **made traversable**.\n* The bridge is built from **accumulated suffering**, not external rescue.\n* Relief comes from **shared movement**, not from victory over pain.\n\nSo the song must avoid portraying compassion as:\n\n* a cure,\n* a rescue,\n* or a triumphant transformation.\n\nInstead, it should frame compassion as **functional, imperfect, and hard-earned**.\n\nThat is why the song repeatedly affirms that:\n\n* the canyon remains,\n* the weight remains,\n* the damage remains,\n\nbut movement becomes possible.\n\nLines like:\n\n¿ *"This bridge won't seal the drop below"\n¿ *"That canyon stays, it always will"\n\ndirectly answer the prompt's insistence that suffering is not erased by connection.\nThis preserves the realism of the metaphor instead of turning it into emotional wish-fulfillment.\n\n—\n\n## Pain as Construction Material ### ..... Verse 1\n\nA canyon cuts the ground in two,\nWe hide the breaks we cannot mend,\nThe pull of loss keeps dragging through,\nLike debts we never get to end.\n\n### Verse 2\n\nAn aging span of battered stone,\nSet down by breath and sleepless nights,\nEach block a wound we faced alone,\nEach seam still bound by failed goodbyes.\n\n### Verse 3\n\nA bent-back vow, a scraped-out plea,\nA risk we took, a truth we bent,\nThey settle in that masonry,\nWhere every hurt learns what it meant.\n\n### Verse 4\n\nThe edges bite, they mark the skin,\nThey map the cost of standing fast,\nYet every cut we carried in,\nBecomes a path across the past.\n\n### Verse 5\n\nThis bridge won't seal the drop below,\nWon't lift the weight we always keep,\nBut gives our shaking feet a row,\nOf solid ground across the deep.\n\n### Verse 6\n\nYou cross with me, I cross with you,\nNo debts to trade, no flags to raise,\nJust hands that know what hands go through,\nAnd choose to walk the same hard days.\n\n### Verse 7\n\nThe load runs light when borne in stride,\nThe fall feels less like final ground,\nWe trade the need to run and hide,\nFor steps that make a steadier sound.\n\n### Verse 8\n\nPast all the breaks we couldn't close,\nPast every word we left unsaid,\nWe reach not peace, but still we know...</answer>

Table 7: Case study from the AlpacaEval2 benchmark. Our model produces a more structured and concept-consistent rock song that better aligns with the prompt's philosophical constraints and narrative progression, whereas the baseline tends to rely on surface-level metaphors and generic expressions.

Prompt	create a simple website with a black background that isn't scrollable. There should be a video in the background for the hero and in the center is should say in large bold white text Explore Zeroh. Under the text have a button saying Join Waitlist this button should be faded out and should have a transparent background with a white outline. The text in the button should be white. Once hovering over the button is transitions to a full white button with black text over it.
Baseline (PRC)	<think>\nThe user wants a simple website with a fixed, full-screen background video that doesn't require scrolling. The content should be centered and include:\n1. Large, bold, white text saying "Explore Zeroh".\n2. A "Join Waitlist" button below the text.\n * The button should be faded out initially.\n * It should have a transparent background.\n * It should have a white outline.\n * The text on the button should be white.\n * On hover, the button should transition to a full white background.\n * The text on the button should transition to black.\n * The hover state should be smooth.\n\nPlan:\n1. **HTML Structure:** Create a basic HTML5 structure with a <!DOCTYPE html> declaration, <html>, <head> for meta tags and styles, and <body>.\n2. **CSS for Layout & Overflow:** Use CSS to make the body overflow hidden. Add position: fixed; top: 0; left: 0; width: 100%; height: 100%; to the body to make it full-screen.\n3. **CSS for Background Video:** Use a <video> element in the <body> for the background. Add position: absolute; top: 0; left: 0; width: 100%; height: 100%; to the video. Add object-fit: cover; to ensure the video fills the screen and maintains aspect ratio by covering the entire area. Add z-index: -1; so the content sits on top. The video source (<source>) should be included within the <video> tag. Use a sample video URL if none is provided.\n4. **CSS for Content:** Create a container for the exploratory text and the button. Use position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); to center it both horizontally and vertically. Add some basic styling for padding and potentially a background color (initially transparent) if needed for stacking, although z-index and position are the primary stackers here.\n5. **CSS for Header Text:**\n * Select the header text (.header-text).\n * Set font-size to a large value (e.g., 50px). You should see the black background, the centered video, the large "Explore Zeroh" text, and the faded-out "Join Waitlist" button. Hovering over the button should make it fully white with black text.\n\nThis setup provides a simple, fixed, non-scrollable layout with the requested visual effects and centering.</answer>"
Ours (AGR)	<think>1. Make the page non-scrollable ('overflow: hidden') and full-screen.\n2. Put a full-bleed **background video** behind everything ('position: fixed; object-fit: cover').\.....\n\n"'html\n<!doctype html>\n<html lang="en">\n <head>\n <meta charset="UTF-8" />\n <meta name="viewport" content="width=device-width, initial-scale=1.0" />\n <title>Explore Zeroh</title>\n <style>\n :root \n –bg: #000;\n –white: #fff;\n –black: #000;\n \n\n /* Prevent scrolling + full-screen layout */\n html, body \n width: 100%;\n height: 100%;\n margin: 0;\n overflow: hidden; /* not scrollable */\n background: var(–bg);\n font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Helvetica, Arial;\n \n\n /* Background video */\n .bg-video \n position: fixed;\n inset: 0;\n width: 100%;\n height: 100%;\n object-fit: cover;\n z-index: 0;\n background: var(–bg);\n \n\n /* Optional dark overlay to keep text readable */\n .overlay \n position: fixed;\n inset: 0;\n z-index: 1;\n background: rgba(0, 0, 0, 0.45);\n \n\n /* Centered hero content */\n .hero \n position: relative;\n z-index: 2;\n width: 100%;\n height: 100%;\n display: flex;\n align-items: center;\n justify-content: center;\n text-align: center;\n padding: 24px;\n box-sizing: border-box;\n \n\n .hero-inner \n display: flex;\n flex-direction: column;\n gap: 18px;\n align-items: center;\n \n\n h1 \n margin: 0;\n color: var(–white);\n font-weight: 800;\n letter-spacing: 0.5px;\n line-height: 1.05;\n font-size: clamp(42px, 6vw, 86px);\n text-shadow: 0 10px 30px rgba(0,0,0,0.35);\n \n\n /* Faded-out outlined button */\n .btn \n appearance: none;\n border: 2px solid rgba(255, 255, 255, 0.9);\n background: transparent;\n color: var(–white);\n font-weight: 700;\n font-size: 16px;\n padding: 12px 22px;\n border-radius: 999px;\n cursor: pointer;\n\n /* "faded out" look */\n opacity: 0.65;\n\n /* smooth hover transition */\n transition: background-color 220ms ease, color 220ms ease, opacity 220ms ease, transform 220ms ease;\n will-change: background-color, color, opacity, transform;\n \n\n .btn:hover \n background: var(–white);\n color: var(–black);\n opacity: 1;\n transform: translateY(-1px);\n \n\n .btn:active \n transform: translateY(0px);\n \n\n .btn:focus-visible \n outline: 3px solid rgba(255,255,255,0.45);\n outline-offset: 4px;\n opacity: 1;\n \n\n /* Respect reduced motion preferences */\n @media (prefers-reduced-motion: reduce) \n .btn transition: none; \n \n </style>\n </head>\n\n <body>\n <!– Replace src with your own video file URL/path –>\n <video class="bg-video" autoplay muted loop playsinline>\n ....</answer>

Table 8: Case study from the ArenaHardV2 benchmark. We use `GPT-5` as an automatic judge to evaluate the correctness of the generated code. Compared with the baseline, the AGR-trained model produces code with fewer functional and stylistic errors, and more faithfully satisfies the prompt's layout and interaction constraints.