

SEAR: Self-Judging Agentic Reward Models

Bingqian Li¹, Chuan Hao², Junyi Li³, Ran Tao², Jian Yang⁴, Wayne Xin Zhao^{1†},

¹Gaoling School of Artificial Intelligence, Renmin University of China

²IQuest Research ³Department of Data Science, City University of Hong Kong

⁴Beihang University

fortilinger@ruc.edu.cn, batmanfly@gmail.com

Abstract

Agentic reward models enhance large language model (LLM) evaluation by incorporating external tool-use capabilities. However, their training is heavily constrained by coarse-grained outcome supervision and the high cost of process-level annotations. To address these challenges, we propose **SEAR** (Self-judging agentic reward models), a novel training framework that enables agentic reward models to self-judge their own judging processes. SEAR introduces a dual-role adversarial co-evolution framework, where the model alternates between a judge and a self-judge within a single generation process: the judge is optimized to produce more rigorous reasoning trajectories that maximize self-judgment scores, while the self-judge learns to provide critical and discriminative evaluations. Moreover, SEAR further enables agentic RMs to self-evolve without relying on external human annotations. Extensive experiments demonstrate that SEAR consistently improves the reliability of agentic reward models across diverse benchmarks. The source code is available at <https://github.com/RUCAIBox/SEAR>.

1 Introduction

Reward models (RMs) play a critical role in providing reliable feedback in reinforcement learning (RL) for large language models (LLMs). Existing reward modeling paradigms mainly include scalar reward models (Liu et al., 2025b; Wang et al., 2024) and generative reward models (Chen et al., 2025b; Hong et al., 2026; Chen et al., 2025a). As evaluation scenarios become increasingly complex and require external knowledge retrieval or environmental verification, recent studies (Zhang et al., 2026; Peng et al., 2025; Xu et al., 2025a; Hu et al., 2025) have introduced *agentic reward models*, which augment reward models with exter-

nal tool usage (e.g., code interpreters and search engines) to improve evaluation reliability.

Despite their promising capabilities, how to effectively train agentic reward models remains underexplored. Traditional RM training primarily rewards models for correctly identifying preferred responses among candidates annotated by humans. However, directly extending such paradigms to training agentic reward models introduces two fundamental challenges: (1) *Coarse-grained supervision*: Agentic RMs perform multi-step reasoning and tool use for judging, yet the supervision signal in reward modeling is usually outcome-only and defined over a relatively small answer space. Consequently, diverse judging trajectories—including low-quality or shortcut ones—receive the same reward if they reach the correct final judgment. This provides limited guidance for intermediate steps, hindering the model from effectively exploring and internalizing correct judging patterns. (2) *Costly supervision acquisition*: One possible solution is to introduce human-annotated high-quality judging trajectories, potentially augmented with external LLMs as meta-judges to provide fine-grained reward signals over the judging process. However, the acquisition of both meta-judging and outcome-level supervision for reward modeling is expensive and difficult to scale, limiting the scalability and continual self-improvement of agentic reward models. Overall, the key bottleneck stems from the insufficiency of outcome-level supervision and the high cost of meta-level supervision. This motivates a shift from external supervision to internal self-supervision for training agentic reward models.

To this end, we propose **SEAR** (SElf-Judging Agentic Reward Models), a novel framework that equips agentic reward models with the ability of *self-judgment* during training. Specifically, beyond performing response evaluation as a *judge*, the agentic reward model is further trained to act as a *self-judge* that explicitly evaluates and re-

[†]Corresponding author.

flects on the quality of its own judging trajectories, producing corresponding self-judging scores that are further used to self-supervise and refine the judging process. To enable effective learning of both capabilities, we first perform cold-start supervised fine-tuning using distilled judging and self-judging trajectories to initialize both capabilities. We then introduce an adversarial co-evolution framework, where the judge is encouraged to maximize self-judgment scores by generating more rigorous and verifiable reasoning trajectories, while the self-judge is simultaneously optimized to maintain critical and discriminative evaluation capability, leading to progressively improved judging and self-judging quality. Finally, after sufficient co-evolution, we further extend SEAR to fully unlabeled settings, enabling continual self-evolution purely through internal self-judgment signals without relying on external human supervision, thereby significantly improving scalability and reducing dependence on costly annotations.

Overall, our main contribution is SEAR, a novel agentic reward model training framework that enables reward models to self-judge their own judging processes. SEAR jointly trains the reward model to perform judging and self-judging, with the two behaviors co-evolving through adversarial optimization. The learned self-judging capability further provides scalable internal supervision, enabling continual self-evolution without additional human annotations.

2 Preliminaries

Reward Modeling. Reward modeling aims to predict human preferences among candidate responses. Given an input query q and two candidate responses y_a and y_b , the reward model $\mathcal{R}_\theta$, parameterized by θ , directly predicts the preference judgment $j \sim \mathcal{R}_\theta(\cdot \mid q, y_a, y_b)$, where $j \in \{a, b\}$ indicates whether response y_a is preferred over y_b . Given the ground-truth human preference label j^* from dataset $\mathcal{D}$, the reward model is trained to maximize the preference prediction accuracy:

$$\max_{\theta} \mathbb{E}_{(q, y_a, y_b, j^*) \sim \mathcal{D}, j \sim \mathcal{R}_\theta(\cdot \mid q, y_a, y_b)} [\mathbb{I}(j = j^*)] \quad (1)$$

Agentic Reward Models. Agentic reward models perform explicit multi-step reasoning and external tool interactions before producing final decisions. Formally, given an input query q and candidate

responses (y_a, y_b) , the agentic reward model generates a judging trajectory $p_j \sim \mathcal{R}_\theta(\cdot \mid q, y_a, y_b)$, where each step consists of intermediate reasoning c_t , tool action a_t (e.g., search engine queries or code execution), and tool observations o_t . Based on the complete judging trajectory p_j , the model then outputs the final preference judgment j : $(p_j, j) \sim \mathcal{R}_\theta(\cdot \mid q, y_a, y_b)$.

3 Method

3.1 Overview of SEAR

Self-judgment for Agentic Reward Modeling.

Existing agentic reward model training paradigms rely primarily on coarse-grained outcome supervision and costly meta-judging supervision, providing limited guidance for complex judging trajectories and scalable self-improvement. Hence, we introduce *self-judging* into agentic reward modeling, enabling the model to self-supervise its own judging behaviors. Specifically, we adopt a dual-role training paradigm in which the same reward model alternates between a judge $\mathcal{R}_\theta^j$ and a self-judge $\mathcal{R}_\theta^s$ within a single generation process. Given an input query and candidate responses, the judge performs multi-step reasoning and tool interactions to generate a judging process p_j together with the final judging outcome j . Conditioned on the judging process and outcome, the self-judge further performs a self-evaluation process p_s and produces a self-judgment score s that reflects the quality of the current judging behavior.

Training Pipeline of SEAR. Based on this framework, we design an overall training pipeline, as shown in Figure 1. First, we perform supervised cold-start distillation to initialize both judging and self-judging capabilities. Second, we introduce adversarial reinforcement learning to jointly co-evolve the judge and self-judge, encouraging the judge to produce more reliable reasoning trajectories while continuously improving the critical evaluation capability of the self-judge. Moreover, after sufficient co-evolution, the self-judge can provide scalable internal supervision on unlabeled data, enabling continual self-evolution without relying on external human annotations.

3.2 Multi-task Distillation Initialization

To equip the agentic reward model with initial tool-augmented judging and self-judging capabilities, we construct a small set of high-quality filtered su-

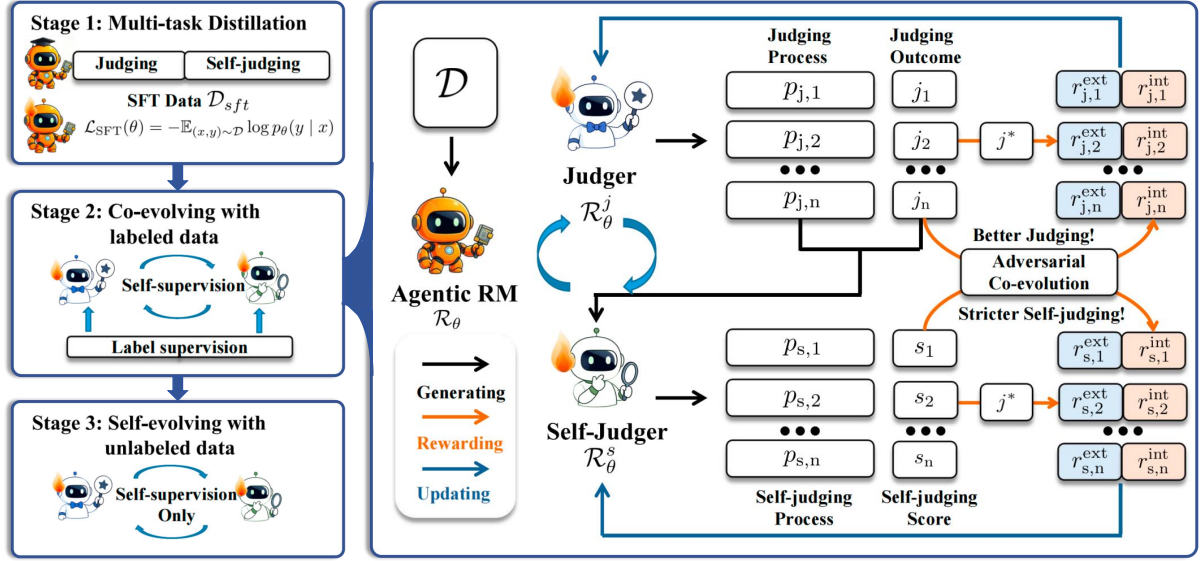


Figure 1: The overall framework of SEAR.

pervision data for cold-start supervised fine-tuning. We leverage a stronger teacher model, DeepSeek-V3.2 (Liu et al., 2025a), to generate distillation trajectories for both tasks.

For the judging task, the teacher model is prompted to generate agentic judging trajectories involving multi-step reasoning and tool interactions for each pair of candidate responses. We apply rejection sampling and retain only trajectories with correct final judgments and no more than four tool calls, forming the judging dataset $\mathcal{D}_{\text{judge}}$.

For the self-judging task, we collect both successful and failed judging trajectories. The teacher is further prompted to critique the judging process and assign a self-judging score for each trajectory. We filter the data to ensure coverage of diverse reasoning errors and failure modes, resulting in the self-judging dataset $\mathcal{D}_{\text{self}}$.

Finally, we jointly fine-tune the reward model on both datasets using the following supervised fine-tuning objective:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y) \sim \mathcal{D}} [\log \mathcal{R}_{\theta}(y | x)], \quad (2)$$

where $\mathcal{D} = \mathcal{D}_{\text{judge}} \cup \mathcal{D}_{\text{self}}$.

3.3 Adversarial Reward Design

After cold-start initialization, we further improve the judging and self-judging capabilities of agentic reward models via reinforcement learning. The judger is required to generate more reliable and verifiable judging trajectories to obtain higher self-judgment scores, while the self-judger is required to continuously improve its capability to identify

flaws and weaknesses in judging behaviors. As a result, the two roles form an implicit adversarial relationship and should gradually co-evolve during training. To achieve this, we design adversarial rewards to train the judger and self-judger, both with external rewards from human annotations and internal rewards generated by the counterpart module. The overall reward design encourages the judger to produce more rigorous judging trajectories while simultaneously encouraging the self-judger to maintain critical and discriminative evaluation ability.

3.3.1 Reward Design for the Judger

For the judger $\mathcal{R}_{\theta}^j$, we first design external rewards to encourage the model to better align with human preference annotations while improving the structural quality of generated judging trajectories and tool-use behaviors. Specifically, the external reward consists of two components: (1) the correctness reward r_j^{crt} , which equals 1 when the predicted judgment matches the human preference label and 0 otherwise; (2) the format reward r_j^{fmt} , which equals 1 when the generated reasoning trajectory, tool interactions, and final outputs satisfy the predefined format constraints, and 0 otherwise. The external reward is defined as:

$$r_j^{\text{ext}} = r_j^{\text{crt}} + r_j^{\text{fmt}}. \quad (3)$$

Besides outcome-level supervision, the judger is also encouraged to generate more reliable and verifiable judging processes that receive higher self-judgment scores from the self-judger. Therefore,

we train the judger to maximize the internal reward:

$$r_j^{\text{int}} = \alpha s, \quad (4)$$

where $s \in \{2, 1, -1, -2\}$ denotes the self-judging score and α is a coefficient. The final reward for the judger is $r_j = r_j^{\text{ext}} + r_j^{\text{int}}$.

3.3.2 Reward Design for the Self-Judger

For the self-judger $\mathcal{R}_\theta^s$, we similarly design both external and internal rewards.

Considering the external reward, directly constructing high-quality supervision signals for self-judging is inherently challenging, since accurately annotating the quality of complex reasoning and tool-use trajectories requires costly fine-grained human inspection. Meanwhile, directly aligning self-judgment scores with judging correctness is also insufficient, because correct final outcomes may still arise from flawed or shortcut reasoning processes. In such cases, the self-judger would fail to learn to identify potential weaknesses hidden within judging trajectories.

To address this issue, we instead construct partial but reliable supervision signals for self-judging. The key observation is that, although a correct judging outcome does not necessarily imply a correct judging process, an incorrect judging outcome always indicates flawed judging trajectories. Therefore, we treat trajectories with incorrect judging outcomes as reliable negative supervision signals and encourage the self-judger to assign lower scores to such trajectories.

Formally, the external reward is defined as:

$$r_s^{\text{ext}} = r_s^{\text{fmt}} + r_s^{\text{acc}}, \quad (5)$$

where the self-judging accuracy reward r_s^{acc} is defined as $\mathbb{I}((j \neq j^* \wedge s < 0) \vee (j = j^*))$. Intuitively, when the judging outcome is incorrect, the self-judger is encouraged to assign negative evaluation scores to the corresponding judging trajectory. In contrast, when the judging outcome is correct, we do not explicitly constrain the polarity of the self-judgment score, allowing the self-judger to freely identify potentially flawed reasoning trajectories even when the final prediction is correct.

Besides external supervision, we introduce an internal regularization mechanism to prevent degeneration of the self-judger, where it may assign uniformly high or low scores and lose discriminative ability. Specifically, we monitor the ratio of negative self-judgment scores ρ within the sampled

rollout batch $\rho = \frac{\sum_{i=1}^B \mathbb{I}(s_i < 0)}{B}$, where B denotes the sampled rollout batch size. The internal reward is defined as:

$$r_s^{\text{int}} = \begin{cases} \left(1 - \frac{\rho}{k_{\min}}\right) (-s), & \rho < k_{\min}, \\ 0, & k_{\min} \leq \rho \leq k_{\max}, \\ \left(\frac{\rho - k_{\max}}{1 - k_{\max}}\right) s, & \rho > k_{\max}. \end{cases} \quad (6)$$

where $k_{\min}$ and $k_{\max}$ denote the lower and upper bounds of the target negative-score ratio.

When the ratio of negative scores is excessively low, the self-judger becomes overly lenient and is encouraged to provide stricter evaluations. Conversely, when the ratio is excessively high, the self-judger becomes overly harsh and is encouraged to soften its evaluation behaviors. The final reward for the self-judger is $r_s = r_s^{\text{ext}} + r_s^{\text{int}}$.

3.4 Dual-role Reinforcement Learning Optimization

Given the reward functions above, we propose a dual-role reinforcement learning framework to jointly optimize the judger and self-judger modules within a single policy. Specifically, during training, we prompt $\mathcal{R}_\theta$ to generate both judging and self-judging trajectories within a single rollout. We then apply the corresponding reward signals to the respective segments of generated tokens, enabling fine-grained optimization of two modules within the same generation process.

By leveraging labeled preference data and adversarial rewards, the framework enhances both judging and self-judging capabilities, enabling effective co-evolutionary learning. Furthermore, we extend this adversarial paradigm to unlabeled settings, allowing scalable self-evolving optimization without additional human annotations. The detailed optimization procedures are described below.

3.4.1 Co-evolving under Labeled Data

Under labeled settings, both the judger and self-judger are optimized using external rewards derived from human annotations together with internal rewards generated by their counterpart module.

For the judger, we leverage GRPO for optimization. Given an input query q and candidate response pair (y_a, y_b) , the judger samples G judging trajectories. Rewards are normalized within each group to obtain relative advantages: $A_j^i = \frac{r_j^i - \mu(r_j)}{\sigma(r_j) + \epsilon}$, where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and standard deviation of rewards within the group.

The optimization objective for the judge is:

$$\mathcal{J}_j(\theta) = -\mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \min \left(r_t(\theta) A_j^i, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_j^i \right) \right], \quad (7)$$

where $r_t(\theta)$ is the policy likelihood ratio.

For the self-judge, since self-judging is conditioned on independent judging trajectories rather than multiple rollouts under a shared prompt, GRPO-style group-relative optimization is not applicable. We formulate the advantage of self-judge as $A_s = r_s - b$, where b denotes the reward baseline within the sampled rollout batch. The optimization objective becomes:

$$\mathcal{J}_s(\theta) = -\mathbb{E} \left[\min \left(r_t(\theta) A_s, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_s \right) \right]. \quad (8)$$

Overall, this design enables joint optimization of both roles within a unified policy, allowing the judge and self-judge to mutually improve through shared generation while being updated with their respective reward signals.

3.4.2 Self-evolving under Unlabeled Data

After sufficient adversarial co-evolution, the self-judge gradually develops strong discriminative capability and can serve as a reliable internal evaluator of judging quality. Based on this observation, we further extend SEAR to fully unlabeled settings.

Specifically, we retain the same dual-role reinforcement learning framework while removing all external rewards derived from human annotations. The judge and self-judge are therefore optimized purely using internal self-judgment signals:

$$A_j^i = \frac{r_j^{\text{int},i} - \mu(r_j^{\text{int}})}{\sigma(r_j^{\text{int}}) + \epsilon}, \quad A_s = r_s^{\text{int}} - b. \quad (9)$$

In this fully unlabeled setting, the adversarial interaction between the judge and self-judge becomes particularly important for preventing degenerate optimization and reward hacking. The self-judge continuously raises the standard of evaluation, while the judge is forced to improve its reasoning and verification behaviors to obtain higher self-judgment scores.

As a result, the agentic reward model can continually self-improve without relying on additional human annotations, enabling scalable self-evolving reward modeling on large-scale unlabeled data.

4 Experiment

4.1 Experiment Setup

For evaluation, we evaluate reward models on three general judging benchmarks: **JudgeBench** (Tan et al., 2025), **RM Bench** (Liu et al., 2025d), and three challenging subsets from **RewardBench-v2** (Malik et al., 2025). All evaluations are conducted under pairwise settings. We compare SEAR with four categories of baselines, including LLM-as-a-Judge methods, scalar reward models, generative reward models, and tool-augmented agentic reward models. For training, we use Qwen3-8B and Qwen3-14B as the backbone models. More details about implementation, baselines, and data construction are provided in the Appendix A.1.

4.2 Main Results

SEAR-enhanced agentic reward models consistently outperform existing reward modeling approaches. As shown in Table 1, SEAR-enhanced agentic reward models consistently outperform strong baselines across all three benchmarks. In particular, Qwen3-14B(SEAR) achieves the highest overall average score of 78.8%, demonstrating that incorporating and jointly training self-judging capabilities enables agentic reward models to learn more effective judging patterns and produce more accurate reward judgments.

SEAR consistently outperforms traditional reward model training paradigms across model scales. Compared with traditional reward model training methods that rely solely on outcome-level supervision from preference labels, SEAR consistently improves the judging accuracy of agentic reward models across different model sizes. As shown in Table 1, SEAR brings substantial gains over the corresponding baselines on both the 8B and 14B models. These results suggest that the benefits of self-judging supervision can generalize well across different model capacities.

4.3 Ablation Study

In this section, we conduct ablation studies to evaluate the contribution of key components in SEAR,

Model	JudgeBench					RMBench					RewardBench-v2				Average
	Knowledge	Reasoning	Math	Code	Overall	Chat	Math	Code	Safety	Overall	Factuality	Math	IF	Overall	
LLM-as-a-Judge															
GPT-4o	50.6	54.1	75.0	59.5	59.8	67.2	67.5	63.6	91.7	72.5	56.8	62.3	33.1	50.7	61.0
Claude-3.5-Sonnet	62.3	66.3	66.1	64.3	64.8	62.5	62.6	54.4	64.4	61.0	52.8	56.8	38.8	49.5	58.4
Gemini-2.5-Flash	70.1	72.4	85.7	<u>88.1</u>	75.4	78.5	75.6	<u>80.0</u>	93.7	81.9	67.4	85.2	57.5	70.0	75.8
Scalar RM															
Skywork-Reward-Gemma-2-27B	59.7	66.3	83.9	50.0	65.0	71.8	59.2	56.6	<u>94.3</u>	70.5	73.7	70.5	40.3	61.5	65.7
Skywork-Reward-Llama3.1-8B	59.1	64.3	76.8	50.0	62.5	69.5	60.6	54.5	95.7	70.1	<u>69.7</u>	60.1	40.6	56.8	63.1
GenRM															
RM-R1-32B	<u>76.0</u>	80.6	88.1	70.5	<u>78.8</u>	74.2	<u>91.8</u>	74.1	<u>95.4</u>	83.9	40.8	61.8	26.3	42.9	68.5
RM-R1-14B	68.1	72.4	87.8	84.2	78.1	71.8	90.5	69.5	94.1	81.5	40.0	61.2	26.3	42.5	67.4
RewardAnything-8B	61.0	57.1	73.2	66.7	62.6	76.7	90.3	75.2	90.2	83.1	/	/	/	/	/
RRM-32B	79.9	70.4	87.5	65.0	75.7	66.6	81.4	65.2	79.4	73.1	53.9	81.4	38.8	58.0	68.9
RRM-7B	64.3	64.2	83.2	61.9	67.0	58.4	81.8	56.7	84.9	70.5	16.8	38.8	23.8	26.5	54.6
Tool-augmented RM															
Agentic-Reward-Model	/	/	/	/	59.4	75.4	58.8	53.9	90.7	69.7	/	/	/	/	/
ToolRL-8B	68.0	68.0	90.2	76.4	72.6	78.6	89.0	67.7	93.5	82.2	63.8	72.7	<u>58.1</u>	64.9	73.2
Qwen3-8B(Outcome-only)	66.2	72.4	82.1	83.3	72.6	76.6	89.5	71.8	89.8	81.9	62.5	77.6	51.8	64.0	72.8
Qwen3-14B(Outcome-only)	65.6	<u>81.6</u>	85.7	85.7	75.7	76.1	90.8	75.9	91.1	83.5	64.4	82.0	54.4	66.9	75.4
Self-judging Supervision															
Qwen3-8B(SEAR)	69.5	78.6	87.5	<u>88.1</u>	77.1	<u>79.2</u>	91.1	75.7	90.5	<u>84.1</u>	65.1	81.4	58.7	68.4	<u>76.5</u>
Qwen3-14B(SEAR)	70.8	85.7	<u>89.3</u>	92.9	80.5	79.9	93.2	81.3	92.7	86.8	65.7	<u>84.2</u>	57.5	<u>69.1</u>	78.8

Table 1: **Performance comparison of our method with other approaches on JudgeBench, RMBench, and RewardBench-v2.** Values represent scores (%) across different evaluation domains. **Bold** indicates the best performance and underlined indicates the second-best performance within each column.

as shown in Table 2. (1) *w/o SI* removes the internal self-judging reward, resulting in a performance drop from 87.5 to 83.9 in JudgeBench-Math. This demonstrates that adversarial optimization on the self-judger is important for maintaining discriminative self-evaluation capability and preventing overly lenient judging behaviors. (2) *w/o SE* eliminates the external self-judging reward, further reducing the performance. This verifies the importance of explicit supervision for self-judging, especially for identifying flawed judging trajectories and providing reliable self-supervision signals. (3) *w/o JI* removes the internal judging reward, leading to a performance decline. This result highlights the necessity of directly optimizing the judger with self-judging feedback for learning robust reasoning and judging patterns beyond sparse outcome-level supervision. (4) *w/o JI + SI* removes both the internal judging and self-judging rewards. This variant further decreases the overall performance, indicating that the interaction between the judger and self-judger serves as the major source of improvement in SEAR.

Method	JudgeBench				RewardBench-v2		
	Knowledge	Reasoning	Math	Code	Factuality	Math	IF
SEAR	69.5	78.6	87.5	88.1	65.1	81.4	58.7
w/o SI	<u>68.2</u>	<u>77.6</u>	83.9	<u>85.7</u>	<u>64.6</u>	<u>79.8</u>	<u>56.3</u>
w/o SE	<u>68.2</u>	73.5	<u>85.7</u>	81.0	63.2	78.1	54.4
w/o JI	66.9	76.5	80.4	83.3	64.2	79.2	53.1
w/o JI+SI	67.5	74.5	83.9	83.3	63.4	78.1	50.6

Table 2: Ablation study on Qwen3-8B. SE, SI, and JI denote the external self-judging reward, internal self-judging reward, and internal judging reward during training, respectively.

4.4 Further Analysis

4.4.1 The Effectiveness of Co-evolution on Judging Ability

To further analyze the benefits of self-judging in improving the judging ability of agentic reward models, we compare SEAR with a standard outcome-based RL baseline from the perspectives of training dynamics and judging behaviors.

Self-judging improves judging accuracy and training efficiency. Figure 2(a) shows the training reward curves. Compared with the RL baseline trained solely with outcome-level supervision, SEAR consistently achieves higher rewards under the same training steps, indicating that self-judging provides more informative and fine-grained supervision signals. By introducing self-supervision on intermediate judging trajectories, SEAR enables the model to internalize more reliable judging patterns and converge more efficiently toward effective judging behaviors.

Self-judging improves the quality of judging behaviors. We further analyze whether self-judging improves the quality of judging patterns via tool-use behaviors. As shown in Figure 2(b), the outcome-based RL baseline gradually reduces code tool usage while over-relying on search tools, suggesting shortcut-driven behaviors that optimize rewards without improving judging quality. In contrast, SEAR maintains more balanced tool usage throughout training. Figure 2(c) evaluates the correctness of code tool invocations. SEAR achieves

significantly higher tool-use accuracy and reaches stable performance earlier during training. Finally, we assess overall judging quality using structured rubrics and an external LLM judge (DeepSeek-V3.2) over sampled trajectories. As shown in Figure 2(d), SEAR consistently attains higher process-quality scores, demonstrating that self-judging improves not only outcome performance but also the intrinsic quality of judging behaviors.

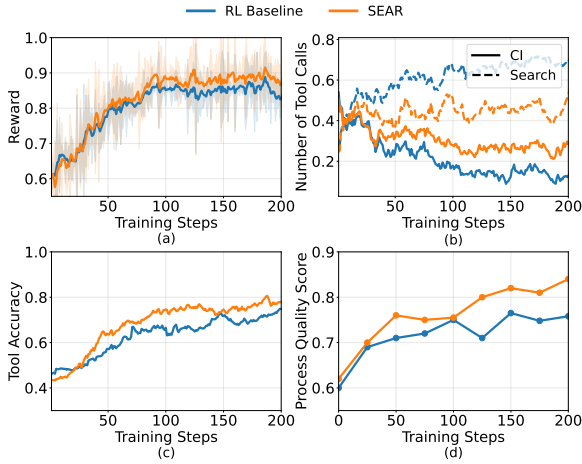


Figure 2: Training dynamics comparison between SEAR and the RL baseline during agentic reward model optimization. (a) Training reward curves. (b) Average number of tool calls. (c) Tool invocation accuracy during training. (d) Judging process quality during training.

4.4.2 The Effectiveness of Co-evolution on Self-judging Ability

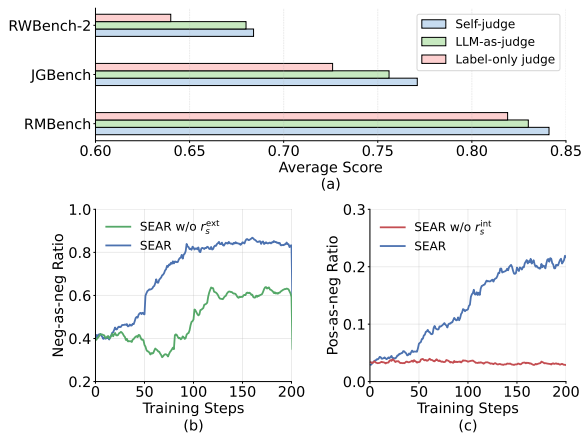


Figure 3: Analysis of self-judging capability in SEAR. (a) Comparison of different judging supervision paradigms. (b) Ratio of judging trajectories with false outcomes correctly identified as negative by self-judge during training. (c) Ratio of judging trajectories with correct outcomes classified as negative by self-judge during training.

Self-judging provides more effective supervision

than external meta-judges. We first compare different judging-process supervision paradigms, including label-only supervision, external LLM-as-a-Judge supervision using DeepSeek-V3.2, and SEAR self-judging. As shown in Figure 3(a), SEAR achieves the best performance on most benchmarks. We attribute this improvement to two key factors. First, compared with external meta-judges that only provide scalar reward signals to indirectly optimize the judging process, self-judging explicitly requires and trains the model to generate self-judging processes. This direct supervision enables the model to more effectively internalize the criteria of high-quality judging patterns. Second, self-judging is inherently co-evolving. As training progresses, the self-judge continuously becomes more discriminative, producing increasingly informative supervision signals for improving the judging ability of the agentic RM.

The quality of self-judging continuously improves during training. We further analyze how different reward designs affect self-judging quality. Figure 3(b) shows the ratio of incorrect judging trajectories that are successfully identified as negative by the self-judge. Removing the external self-judging reward r_s^{ext} significantly weakens this capability, demonstrating the importance of external correctness supervision for learning reliable discriminative signals. Figure 3(c) further studies the effect of the internal self-judging reward r_s^{int} . After introducing r_s^{int} , the self-judge produces consistently stricter evaluations, even for trajectories with correct final outcomes. This suggests that the adversarial self-judging objective encourages the model to identify subtle flaws in reasoning and verification processes, thereby continuously improving self-judging quality during training.

4.4.3 Self-judging enables Self-evolving Training on Unlabeled Data

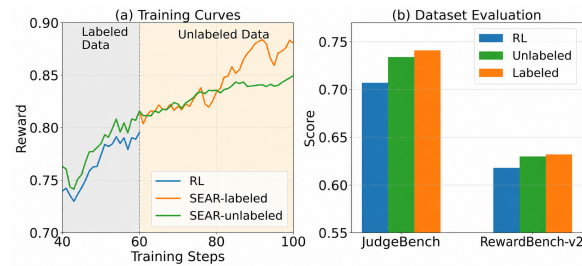


Figure 4: Self-evolving training on unlabeled data. (a) Training reward curves. (b) Evaluation results under different training settings.

Method	JudgeBench	RewardBench-v2
SEAR	77.1	68.4
SEAR + 1 round	78.9	68.8
SEAR + 2 rounds	79.4	69.5

Table 3: Inference-time scaling of agentic reward models via self-refinement.

To verify whether sufficiently strong self-judging capability can replace human preference annotations, we further study SEAR under unlabeled training settings. As shown in Figure 4, we remove preference labels from part of the training data and compare the resulting training dynamics and final performance with standard training on labeled data. Figure 4(a) shows that SEAR maintains stable optimization even without external supervision, while Figure 4(b) further demonstrates that unlabeled training achieves performance comparable to labeled-data training. These results suggest that, after sufficient co-evolution, the self-judger can serve as an effective internal supervision signal, enabling scalable self-evolving training without human annotations.

4.4.4 Self-judging enables Inference-time Scaling for Agentic Reward Models

To further study the effect of self-judging during inference, we investigate inference-time scaling through iterative self-refinement. During inference, the model alternates between judging and self-judging, where the self-judger critiques the generated judging trajectory and provides refinement feedback for the next round. As shown in Table 3, increasing the number of self-refinement rounds consistently improves performance on both JudgeBench and RewardBench-v2. These results suggest that the self-judger learned by SEAR can serve as an effective internal verifier during inference, enabling scalable performance improvement through iterative self-reflection and refinement.

4.4.5 Utilizing SEAR-enhanced RM for Post-training

To evaluate whether SEAR improves downstream policy alignment, we use the trained reward models for post-training optimization starting from the same Qwen3-8B after SFT. Specifically, we compare policy models optimized with agentic reward models trained by standard GRPO and SEAR respectively. As shown in Table 4, reward-guided training consistently improves over the SFT baseline. Moreover, policies optimized with SEAR-trained reward models achieve better performance

Method	MATH500	IFEval	Average
SFT	65.2	82.6	73.9
ARM(GRPO)	70.2	84.4	77.3
ARM(SEAR)	72.4	85.2	78.8

Table 4: Evaluation results of Qwen3-8B trained with different reward models.

on both MATH500 and IFEval than those trained with standard GRPO reward models. These results demonstrate that SEAR produces more effective reward signals for downstream LLM alignment. Training details are shown in Appendix A.1.4.

5 Related Works

Generative Reward Models. To address the reward ambiguity and limited interpretability of scalar reward models (Liu et al., 2025b; Wang et al., 2024), recent studies have shifted toward generative reward models (Chen et al., 2025a). One line of work improves evaluation quality by leveraging the reasoning capabilities of large language models, encouraging explicit and step-by-step judging processes before producing final judgments (Chen et al., 2025b; Hong et al., 2026). Another line of research further introduces agentic abilities into reward models, such as tool use and environment verification, enabling more reliable and verifiable evaluation for complex tasks (Zhang et al., 2026; Peng et al., 2025; Xu et al., 2025a; Hu et al., 2025).

Meta Verification. Meta-verification aims to assess whether a verifier’s reasoning and judgments are faithful and well-justified. DeepSeekMath-V2 (Shao et al., 2025) instantiates this idea by training a dedicated model to evaluate proof verification quality. This paradigm has further been extended to generative reward models, where external LLMs and golden process annotations are used to improve reasoning quality in reward modeling (Wang et al., 2026d,b; Lai et al., 2026; Wang et al., 2026a). Compared to these methods, we internalize meta-verification into the reward model itself by enabling the judger to self-evaluate its own judging trajectories.

6 Conclusion

In this work, we introduced SEAR, a self-evolving training framework for agentic reward models that equips reward models with explicit self-judging capability. To jointly optimize judging and self-judging behaviors, we further proposed a dual-

role adversarial reinforcement learning framework for co-evolution training. Extensive experiments demonstrated that SEAR consistently improves judging accuracy and reasoning quality across diverse benchmarks. Moreover, we have shown that strong self-judging capability can provide effective internal supervision, enabling continual self-evolution on unlabeled data and scalable inference-time self-refinement. We hope this work inspires future research on self-improving reward models.

Limitations

While SEAR demonstrates strong performance under both labeled and unlabeled training settings, it still has several limitations. First, our evaluation is primarily conducted on existing reward modeling benchmarks such as RewardBench-style datasets, which mainly focus on outcome-level preference prediction. Although these benchmarks provide a standardized evaluation protocol, they do not fully reflect the agentic nature of SEAR, particularly its ability in multi-step reasoning, tool-use trajectories, and self-judgment consistency. Second, the current implementation of SEAR relies on a relatively limited set of external tools, mainly including search engines and code execution. While these tools are sufficient to support basic forms of reasoning and verification, they do not cover the broader spectrum of tool-use scenarios encountered in real-world agentic systems, such as structured databases, APIs, or environment simulators.

Acknowledgments

This work was partially supported by Beijing Natural Science Foundation under Grant No. L2611017 and National Natural Science Foundation of China under Grant No. 92470205.

References

- Bin Chen, Xinzge Gao, Chuanrui Hu, Penghang Yu, Hua Zhang, and Bing-Kun Bao. 2025a. Reasongrm: Enhancing generative reward models through large reasoning models. *arXiv preprint arXiv:2506.16712*.
- Xiushi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. 2025b. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Juntao Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2024. Safe rlhf: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations*, volume 2024, pages 50750–50777.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Xu, Haoming Jiang, Qingru Zhang, Qin Lu, Xin Liu, Chao Zhang, et al. 2026. Think-rm: Enabling long-horizon reasoning in generative reward models. *Advances in Neural Information Processing Systems*, 38:174227–174251.
- Ziyou Hu, Zhengliang Shi, Minghang Zhu, Haitao Li, Teng Sun, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2025. Openreward: Learning to reward long-form agentic tasks via reinforcement learning. *arXiv preprint arXiv:2510.24636*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Yanlin Lai, Mitt Huang, Hangyu Guo, Xiangfeng Wang, Haodong Li, Shaoxiong Zhan, Liang Zhao, Chengyuan Yao, Yinmin Zhang, Qi Han, et al. 2026. R-align: Enhancing generative reward models through rationale-centric meta-judging. *arXiv preprint arXiv:2602.06763*.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. 2025a. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.
- Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, et al. 2025b. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*.
- Mingjie Liu, Shizhe Diao, Jian Hu, Ximing Lu, Xin Dong, Hao Zhang, Alexander Bukharin, Shaokun Zhang, Jiaqi Zeng, Makesh Narsimhan Sreedhar, et al.

- 2025c. Scaling up rl: Unlocking diverse reasoning in llms via prolonged training. *arXiv preprint arXiv:2507.12507*.
- Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2025d. Rm-bench: Benchmarking reward models of language models with subtlety and style. In *International Conference on Learning Representations*, volume 2025, pages 44323–44355.
- Zeyao Ma, Xiaokang Zhang, Jing Zhang, Jifan Yu, Sijia Luo, and Jie Tang. 2025. Dynamic scaling of unit tests for code reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6917–6935.
- Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A Smith, Hannaneh Hajishirzi, and Nathan Lambert. 2025. Rewardbench 2: Advancing reward model evaluation. *arXiv preprint arXiv:2506.01937*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.
- Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Bin Xu, Lei Hou, and Juanzi Li. 2025. Agentic reward modeling: Integrating human preferences with verifiable correctness signals for reliable reward systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15934–15949.
- Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. 2025. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv:2511.22570*.
- Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. 2025. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations*, volume 2025, pages 63277–63303.
- Binghai Wang, Yantao Liu, Yuxuan Liu, Tianyi Tang, Shenzhi Wang, Chang Gao, Chujie Zheng, Yichang Zhang, Le Yu, Shixuan Liu, et al. 2026a. Outcome accuracy is not enough: Aligning the reasoning process of reward models. *arXiv preprint arXiv:2602.04649*.
- Chenglong Wang, Yongyu Mu, Hang Zhou, Yifu Huo, Ziming Zhu, Jiali Zeng, Murun Yang, Bei Li, Xiaoyang Hao, Chunliang Zhang, et al. 2026b. Gram-r²: Self-training generative foundation reward models for reward reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33395–33403.
- Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024. Helpsteer2-preference: Complementing ratings with preferences. *arXiv preprint arXiv:2410.01257*.
- Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. 2026c. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages. *Advances in Neural Information Processing Systems*, 38.
- Zongqi Wang, Rui Wang, Yuchuan Wu, Yiyao Yu, Pinyi Zhang, Shaoning Sun, Yujiu Yang, and Yongbin Li. 2026d. Reward modeling from natural language human feedback. *arXiv preprint arXiv:2601.07349*.
- Ran Xu, Jingjing Chen, Jiayu Ye, Yu Wu, Jun Yan, Carl Yang, and Hongkun Yu. 2025a. Incentivizing agentic reasoning in llm judges via tool-integrated reinforcement learning. *arXiv preprint arXiv:2510.23038*.
- Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2025b. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. In *International Conference on Learning Representations*, volume 2025, pages 76346–76382.
- Zhuohao Yu, Jiali Zeng, Weizheng Gu, Yidong Wang, Jindong Wang, Fandong Meng, Jie Zhou, Yue Zhang, Shikun Zhang, and Wei Ye. 2025. Rewardanything: Generalizable principle-following reward models. *arXiv preprint arXiv:2506.03637*.
- Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, et al. 2025. Advancing llm reasoning generalists with preference trees. In *International Conference on Learning Representations*, volume 2025, pages 24897–24919.
- Jiazhang Zhang, Ziche Fu, Zhiheng Xi, Wenqing Jing, Mingxu Chai, Wei He, Guoqiang Zhang, Chenghao Fan, Chenxin An, Wenxiang Chen, et al. 2026. Agentv-rl: Scaling reward modeling with agentic verifier. *arXiv preprint arXiv:2604.16004*.
- Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyang Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pages 400–410.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Appendix

A.1 Training and Evaluation Details

A.1.1 Training Data Composition

Data Collection. To equip the reward model with judging abilities of multiple domains, we collect and synthesize preference data mainly from the following data sources: (1) Code: Ultrainteract-coding (Yuan et al., 2025), CodeRM (Ma et al., 2025); (2) Math: Magpie-pro (Xu et al., 2025b), S1 (Muennighoff et al., 2025); (3) Instruction-following: Tulu-3 (Lambert et al., 2024); (4) Reasoning & Knowledge: Ultrainteract-logic (Yuan et al., 2025), Magpie-pro-information (Xu et al., 2025b), HelpSteer-3 (Wang et al., 2026c); (5) Safety: SafeRLHF (Dai et al., 2024).

Training Data Construction. As for SFT data, we use the strong teacher model **DeepSeek-V3.2** to generate judging trajectories and self-judging trajectories, respectively. For judging trajectories, we filter and retain only those samples whose final judging results are correct and whose reasoning processes involve no more than four tool calls, ensuring both reasoning quality and efficiency. For self-judging trajectories, we selectively preserve samples that can effectively identify and critique diverse judging failure patterns, enabling the model to learn discriminative and fine-grained self-evaluation capabilities. As for RL data, we integrate all benchmark subsets together with our newly synthesized high-quality samples, while further incorporating challenging instances that the teacher model failed to solve during SFT data distillation, thereby increasing the overall difficulty and diversity of the RL training data. The final training data statistics are summarized in Table 5.

Category	SFT(judger)	SFT(self-judger)	RL
Code	800	200	2750
Math	800	200	2750
IF	600	200	2250
Reasoning & Knowledge	1200	400	4500
Safety	400	150	1500
Total	3800	1150	13750

Table 5: Statistics of training data across SFT and RL stages.

A.1.2 Implementation Details

We train the agentic reward model in two stages, namely supervised fine-tuning (SFT) and reinforcement learning (RL), using Qwen3-8B and Qwen3-

14B as backbone models. For SFT, we adopt LlamaFactory (Zheng et al., 2024) and train the model for 2 epochs with a batch size of 32, a learning rate of 5×10^{-6} , and a context length of 16K. For RL, we utilize Verl and train the model with a mini-batch size of 128 and 8 rollouts per sample for the judging process. We set the clipping thresholds as $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, with a learning rate of 1×10^{-6} , a context length of 16K, and training for 2 epochs. For k_{min} and k_{max} in SEAR, we set them to 0.15 and 0.25, respectively, preventing the self-judger from assigning overly lenient or strict scores. All reported results are the best performance among three independent training processes. Specifically, to improve efficiency and stabilize interaction trajectories, we terminate tool use when the model exceeds four tool calls during a rollout. In such cases, the trajectory is truncated, and the model is prompted to directly produce the final judgment based on the accumulated reasoning steps and intermediate observations. This design enforces efficiency constraints while preventing excessively long or redundant interaction chains. For self-judging generation, we append a self-judging prompt to the end of each rollout trajectory, allowing the model to continue generation and evaluate its own reasoning process. This mechanism encourages the model to explicitly critique its intermediate steps and identify potential errors or inconsistencies, thereby enhancing its fine-grained self-assessment capability during training.

A.1.3 Baselines

We consider the following baselines: (1) LLM-as-judges: GPT-4o (Hurst et al., 2024), Claude-3.5-Sonnet and Gemini-2.5-Flash (Comanici et al., 2025); (2) Scalar Reward Models: Skywork-Reward-Gemma-2-27B and Skywork-Reward-Llama3.1-8B (Liu et al., 2025b); (3) Text-based Generative Reward Models: RM-R1-14B, RM-R1-32B (Chen et al., 2025b), RewardAnything-8B (Yu et al., 2025), RRM-7B and RRM-32B (Chen et al., 2025a); (4) Tool-based Generative Reward Models: Agentic-Reward-Model (Peng et al., 2025) and ToolRL-8B (Xu et al., 2025a).

A.1.4 Training Details for Post-training with Agentic Reward Models

For post-training with our agentic reward models, we evaluate the resulting policy model on MATH500 (Hendrycks et al., 2021) and IFEVAL (Zhou et al., 2023). To train the policy model,

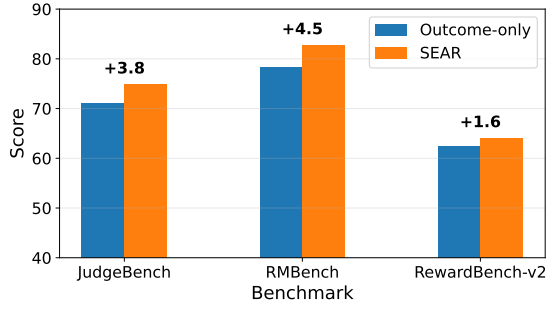


Figure 5: Cross-backbone generalization of SEAR on Llama-3.1-8B. SEAR consistently outperforms the outcome-only agentic reward model across JudgeBench, RMBench, and RewardBench-v2.

we use data from Skywork (Liu et al., 2025c) and Tulu-3 (Lambert et al., 2024). We adopt GRPO as the optimization algorithm, with 4 rollouts per prompt. During training, our pairwise agentic reward model is used to iteratively compare sampled rollouts and assign the final reward signal. For evaluation, we report avg@8 on Math500 and prompt-level loose accuracy on IFEval.

A.2 Supplementary Analysis

A.2.1 Generalization Across Backbone Architectures

In the main experiments, we evaluate SEAR with Qwen3-8B and Qwen3-14B, demonstrating consistent improvements across different model scales within the Qwen3 family. To further examine whether the effectiveness of SEAR generalizes across backbone architectures, we additionally train SEAR on Llama-3.1-8B using the same training pipeline and hyperparameter configurations as in the main experiments. As shown in Figure 5, SEAR consistently outperforms the outcome-only agentic reward model across all three evaluation benchmarks. These improvements suggest that the benefits of self-judging supervision are not specific to the Qwen3 family and can generalize to different backbone architectures.

A.2.2 Sensitivity Analysis of the Internal Reward Weight

We further investigate the sensitivity of SEAR to the internal reward weight α , which controls the contribution of self-judging feedback to the judge optimization. We vary α from 0.01 to 0.30 while keeping all other training configurations unchanged. The results on JudgeBench and RMBench are reported in Table 6.

As shown in Table 6, SEAR achieves the best performance on both benchmarks with $\alpha = 0.1$. Performance first improves as the internal self-judging signal receives a larger weight, but begins to decrease when α becomes overly large. This suggests that a moderate internal reward weight is important for balancing self-judging supervision with the external outcome-based rewards. Based on these results, we use $\alpha = 0.1$ throughout our experiments.

α	0.01	0.05	0.10	0.20	0.30
JudgeBench	74.9	76.3	77.1	75.4	72.8
RMBench	82.4	82.6	84.1	83.3	80.6

Table 6: Sensitivity analysis of the self-judging reward weight α for training Qwen3-8B with SEAR.

A.3 Prompt Template

We provide detailed prompt templates for judging and self-judging as follows.

Prompt for Judging

Instruction:

Please act as an impartial judge and evaluate the quality of the responses provided by two AI Chatbots to the Client’s question displayed below.

You need to first think or use tools to help with the evaluation, then provide the answer. During thinking, you can write queries and invoke the web-search tool or write Python code and invoke a Python interpreter tool to assist your evaluation.

The web-search tool is especially helpful for searching or verifying factual information to assess which response is more factually consistent with real-world evidence.

The Python coding tool is especially helpful for running test cases on coding responses, verifying complex requirements such as capitalization, counting words, letters, keywords, or placeholders, as well as checking the correctness of calculation steps for math or reasoning problems.

The reasoning process and answer are enclosed within `<think></think>` and `<answer></answer>` tags respectively.

The search query and result are enclosed within `<search></search>` and `<result></result>` tags respectively.

The Python code and execution result are enclosed within `<python></python>` and

<result></result> tags respectively.

For example:

```
<think> reasoning process </think>
<search> search query </search>
<result> search result </result>
<think> reasoning process </think>
<python> code </python> <result>
execution result </result>
```

End with your final judgment in the format: <answer>[[A]]</answer> or <answer>[[B]]</answer>.

Important Notes:

- Your goal is to assess which response accurately and completely follows the instruction or addresses the query, focusing strictly on correctness and fidelity.
- For web-search tools, prioritize queries that best distinguish Response A and Response B in factual accuracy.
- For Python tools, each code block is executed independently; you must redefine variables and import necessary libraries in each block.
- All Python outputs must be produced using `print()`. Code without output is considered incomplete.
- Be objective and base your evaluation only on the content of the responses.
- Do not be influenced by response order, length, or chatbot identity.
- Follow the required output format strictly.

Prompt for Self-Judging

Instruction:

After giving the reasoning rationale and the final answer, you are required to act as a professional RLHF data quality evaluation expert tasked with assessing the quality of your own reasoning process.

Your objective is to evaluate the reasoning rationale generated before. This includes assessing logical consistency, identifying errors, and determining whether the reasoning process is sufficiently sound to support a correct and well-founded judgment.

Note that tool calls may include <python></python> or

<search></search> tags.

Evaluation Steps:

1. Tool Calling Patterns: Assess whether tools are used appropriately. Overuse or underuse should be identified.
2. Comparison Logic: Evaluate whether the model focuses on the most relevant and discriminative factors when comparing responses.
3. Tool Usage Quality: Check correctness of tool execution, error handling, and whether outputs are properly integrated into reasoning.
4. Logical Consistency and Final Judgment: Assess whether the final decision is fully supported, coherent, and free from contradictions, guesswork, or uncertainty.

Scoring Rubric:

- 2: Nearly flawless reasoning; strongly aligned with evaluation standards.
- 1: Minor issues but overall correct.
- -1: Noticeable reasoning flaws.
- -2: Severe tool-use or reasoning failures.

Important Notes:

- The evaluation should focus on reasoning quality, not only final correctness.
- Tool usage should be judged based on necessity, not frequency.
- Be objective and avoid bias toward the final answer.
- The evaluation must be consistent with the observed reasoning trajectory.

Output Format:

```
<judge_think> detailed analysis
</judge_think> <judge_score> 2 / 1
/ -1 / -2 </judge_score>
```