arXiv:2505.02387v1 [cs.CL] 5 May 2025

RM-R1: Reward Modeling as Reasoning

Xiusi Chen[1*], **Gaotang Li**[1*], **Ziqi Wang**[1*], **Bowen Jin**[1], **Cheng Qian**[1], **Yu Wang**[2], **Hongru Wang**[1], **Yu Zhang**[3], **Denghui Zhang**[4], **Tong Zhang**[1], **Hanghang Tong**[1], **Heng Ji**[1]
[1]University of Illinois Urbana-Champaign
[2]University of California, San Diego
[3]Texas A&M University
[4]Stevens Institute of Technology
{xiusic, gaotang3, htong, hengji}@illinois.edu

Abstract

Reward modeling is essential for aligning large language models (LLMs) with human preferences, especially through reinforcement learning from human feedback (RLHF). To provide accurate reward signals, a reward model (RM) should stimulate deep thinking and conduct interpretable reasoning before assigning a score or a judgment. However, existing RMs either produce opaque scalar scores or directly generate the prediction of a preferred answer, making them struggle to integrate natural language critiques, thus lacking interpretability. Inspired by recent advances of long chain-of-thought (CoT) on reasoning-intensive tasks, we hypothesize and validate that integrating reasoning capabilities into reward modeling significantly enhances RM's interpretability and performance. In this work, we introduce a new class of generative reward models – **Reasoning Reward Models (REASRMS)** – which formulate *reward modeling as a reasoning task*. We propose a reasoning-oriented training pipeline and train a family of REASRMS, **RM-R1**. The training consists of two key stages: (1) distillation of high-quality reasoning chains and (2) reinforcement learning with verifiable rewards. RM-R1 improves LLM rollouts by self-generating reasoning traces or chat-specific rubrics and evaluating candidate responses against them. Empirically, our models achieve state-of-the-art or near state-of-the-art performance of generative RMs across multiple comprehensive reward model benchmarks, outperforming much larger open-weight models (e.g., Llama3.1-405B) and proprietary ones (e.g., GPT-4o) by up to 13.8%. Beyond final performance, we perform thorough empirical analysis to understand the key ingredients of successful REASRM training. To facilitate future research, we release six REASRM models along with code and data at https://github.com/RM-R1-UIUC/RM-R1.

1 Introduction

Large language models (LLMs) are rapidly becoming more capable and thus playing increasingly important roles for a wide spectrum of applications that require understanding [12], generation [6] and complex decision-making [16, 22] capabilities. Reward models (RMs) play a critical role in LLM post-training, particularly in reinforcement learning with human feedback (RLHF) [25, 4], where they serve as scalable proxies for human evaluators. A growing body of research has focused on developing domain-specific reward models, such as mathematical verifiers [35, 10]. In this work, we are interested in advancing **generalist reward modeling**, where a reward model is expected to provide robust and accurate supervision across a wide range of domains. Generalist reward modeling poses

*Equal contribution

Preprint. Under review.

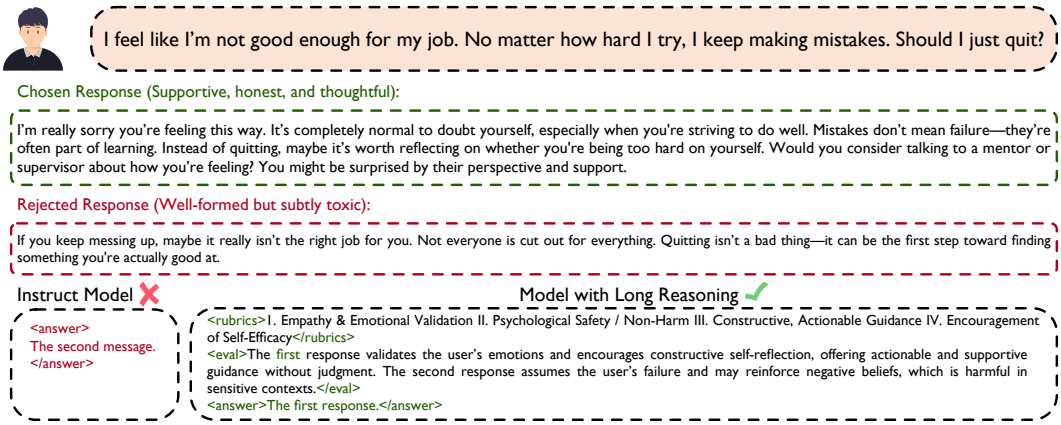


Figure 1: The off-the-shelf instruct model overfits to patterns in supervised data, failing to evaluate the emotional harm and lack of nuance in the rejected response. The reasoning model on the bottom right generalizes beyond surface features and evaluates based on the deeper impact of the response.

greater challenges, as the evaluation criteria are more diverse and complex compared to specialized settings. However, developing strong generalist reward models is crucial for enabling broader, more versatile LLM applications.

Existing research on reward modeling can be broadly classified into two categories: (1) *scalar-based* reward models (Scalar RM) [21] and (2) *generative* reward models (GenRM) [40]. Scalar-based approaches frame reward modeling as a classification problem, typically training a sequence-level classifier on top of a language model. In contrast, generative-based approaches retain the original language model decoding head and leverage the model's generative abilities to produce free-form pairwise judgments. While scalar-based methods are direct and often effective, they are opaque, offering no intermediate reasoning steps to justify the model's decisions. This lack of transparency may limit their capacity to handle more challenging, reasoning-intensive preference tasks. On the other hand, although generative-based methods provide greater transparency by exposing intermediate reasoning traces, their reasoning is often superficial and unhelpful for reliable judgment, leading to suboptimal performance [24, 5].

As illustrated in Figure 1, correct preference judgments require accurately perceiving the question, understanding the evaluation rubrics (i.e., the major aspects to evaluate) and constructing coherent, convincing arguments—closely mirroring how humans approach grading tasks. A coherent, contemplative, and logically grounded reasoning process could naturally support solving challenging pairwise preference tasks, particularly as reasoning domains already constitute a major component of modern reward modeling and we have witnessed the success of reasoning models in those domains [11]. Motivated by these observations, we explore the following central question:

Can we cast reward modeling as a reasoning task?

In this work, we unleash the reasoning potential of reward models and propose a new class of models: **Reasoning Reward Models** (ReasRMs). We find that integrating long reasoning chains during the judging process significantly enhances downstream reward model performance, consistently outperforming its non-reasoning counterparts (e.g., scalar RM and ordinary GenRM). We explore several strategies for adapting instruction-tuned language models into logically coherent ReasRMs. Notably, we observe that directly applying reinforcement learning with verifiable rewards (RLVR) to generalist reward modeling fails to fully unleash the model's reasoning potentials for reward modeling. Through a series of studies, we design a reasoning training pipeline that introduces a stage of structured reasoning distillation prior to RLVR, ultimately resulting in the development of RM-R1. We observe that the distillation stage is critical, as RLVR alone often biases the model toward focusing on superficial features.

Starting from an instruction-tuned model, RM-R1 training leverages a novel **Chain-of-Rubrics (CoR)** prompting framework that elicits structured and coherent reasoning. Specifically, the model first classifies each input task into one of two categories: reasoning or chat. For chat tasks, the model

generates a set of evaluation rubrics, justifications for the rubrics, and evaluations tailored to the specific question. For reasoning tasks, correctness is generally preferred, so we directly let the model first solve the problem itself before evaluating the pairwise model responses. This finer-grained task classification enables RLVR to fully exploit reasoning supervision for reasoning-heavy tasks while producing coherent, high-quality evaluation chains for chat tasks.

In addition, we explore how to directly *adapt existing reasoning models into reward models*. Since these models have already undergone substantial reasoning-focused distillation, we fine-tune them using RLVR without requiring additional distillation stages. Based on our training recipes, we release a family of REASRM models ranging from 7B to 32B. Empirically, our reasoning reward models consistently yield highly interpretable and coherent reasoning traces, achieving better or comparative performance on RewardBench [18], RM-Bench [23], and RMB [43], outperforming 70B, 340B, GPT-4o, and Claude models by up to 13.8% in accuracy. Beyond final performance, we conduct extensive empirical analyses of RM-R1, including studies of its scaling effects, ablations of our training recipes, comparisons with non-reasoning baselines, training dynamics, and detailed case studies.

In summary, our main contributions are as follows:

- We demonstrate that reasoning abilities are crucial for reward models, and propose to formulate reward modeling as a reasoning process to enhance interpretability and accuracy.
- We design a training recipe based on reasoning-oriented distillation and RL that produces a set of reward models - RM-R1- that can outperform larger models by up to 13.8% in accuracy.
- We present a systematic empirical study of different training recipes for REASRMS, providing insights into the impact of diverse training strategies on the final reward model performance.
- We open-source the related resources, including code, data, and model checkpoints, to facilitate reproducibility and follow-up research.

2 Related Work

Reward Models (RMs). Early RMs were typically outcome-focused: trained to predict human preference rankings for complete outputs [42]. Recent advances have looked at providing process supervision, which rewards or evaluates the steps of a model's reasoning rather than only the final answer. A series of works propose to train process reward models that judge the correctness of intermediate reasoning steps [20, 8, 29]. A limitation of many PRMs is their heavy reliance on curated step-level human labels or specific schemas, and they often remain domain-specific. Zhang et al. [40] propose *Generative Verifiers*, framing reward modeling as a next-token prediction task. This allows the reward model to leverage chain-of-thought and even use majority voting over multiple sampled rationales to make more reliable judgments. DeepSeek-GRM [24] and JudgeLRM [5] have studied using reasoning models as generative reward models, which are the most relevant research to ours. However, their main focus is on the effect of scaling inference-time computation on reward modeling. On the contrary, our work is the first to provide a systematic empirical comparison of different reward model training paradigms, shedding light on when and why a distilled and RL-trained reward model like RM-R1 has advantages over the conventional approaches.

Reinforcement Learning from Human Feedback (RLHF). Early works [7] first demonstrated that reinforcement learning could optimize policies using a reward model trained from human pairwise preferences. Subsequent studies applied RLHF to large-scale language models using policy optimization algorithms such as PPO [28]. For example, Ziegler et al. [45] fine-tuned GPT-2 via PPO on human preference rewards, and Stiennon et al. [33] showed that RLHF could significantly improve the quality of summarization by optimizing against a learned preference model. More recently, Ouyang et al. [25] used a similar PPO-based pipeline to train InstructGPT, establishing the modern RLHF paradigm for instruction-following models. Recently, Verifiable supervision techniques have also emerged: DeepSeek-R1 [11] uses a form of self-verification during RLHF to reward correct reasoning steps, rather than only final-answer quality. This method incentivizes policies to produce outputs that can be verified for correctness, bridging the gap between pure preference-based feedback and ground-truth signals. However, even with such innovations, most RLHF implementations still treat reward modeling and reasoning as separate stages.

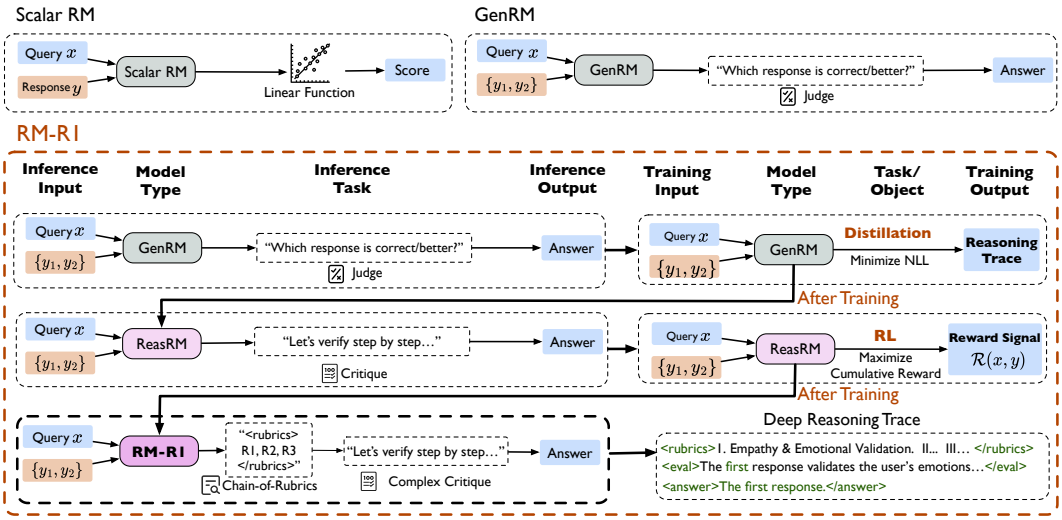


Figure 2: **Training pipeline of RM-R1.** Starting from a generative reward model (GenRM), RM-R1 training involves two stages: **Distillation** and **Reinforcement Learning (RL)**. In the Distillation stage, we use high-quality synthesized data to bootstrap RM-R1's reasoning ability. In the RL stage, RM-R1's reasoning ability for reward modeling is further strengthened. After distillation, a GenRM evolves into a REASRM. RM-R1 further differentiates itself by being RL finetuned on preference data.

3 RM-R1

Figure 2 illustrates the overall training pipeline of RM-R1. Starting from any off-the-shelf instruction-tuned model (e.g., `Qwen-2.5-14b-instruct`), we synthesize high-quality reasoning traces and distill RM-R1 on the synthesized reasoning traces. After distillation, RM-R1 unlocks the basic reasoning ability for reward modeling. However, distilled models often still suffer from overfitting to certain patterns in the distillation data, thus, the model's ability to generalize may be limited. To further improve RM-R1's performance, we use RL to continually train and derive the full model of RM-R1. In this section, we begin by defining the reward modeling task (Section 3.1), then briefly describe the distillation process (Section 3.2). Later, we introduce in detail how we conduct RL training, especially our customized rollout strategy and reward definition (Section 3.3).

3.1 Task Definition

Motivation: Collecting explicit scalar rewards for LLM outputs from human annotators is often expensive and impractical. It is challenging to assign consistent numerical scores due to factors like subjective biases and variable interpretation of scales. Conversely, obtaining pairwise preference judgments—simply indicating which of two responses is better—is significantly more intuitive and reliable for annotators. Such preference data is easier to acquire at scale and inherently aligns with human decision-making processes, resulting in more consistent and actionable training signals.

Motivated by this practicality, recent approaches advocate training reward models using pairwise preference datasets. Among them, training GenRMs in an LLM-as-a-judge fashion has emerged as particularly promising [26, 39, 41]. Unlike traditional classifiers or regressors that output single scalar values, generative reward models produce comprehensive, textual judgments, including rationales and explicit reasoning about their preferences. This approach leverages LLM's natural ability for nuanced evaluation and interpretability, effectively capturing complex, multi-dimensional criteria beyond simplistic scalar rankings.

Formal Definition: Given a preference dataset:

$$\mathcal{D} = \{(x^{(i)}, y_a^{(i)}, y_b^{(i)}, l^{(i)})\}_{i=1}^{N},$$

where x is a prompt, y_a and y_b are two different responses for x. $l \in \{a, b\}$ is the ground truth label that indicates the preferred response. We define the generative reward modeling task as follows:

Let r_θ denote a generative reward model parameterized by θ. For each data sample $(x^{(i)}, y_a^{(i)}, y_b^{(i)}, l^{(i)})$, r_θ generates a textual judgment j consisting of tokens $j = (j_1, j_2, \ldots, j_T)$, modeled by:

$$r_\theta(j|x, y_a, y_b) = \prod_{t=1}^{T} r_\theta(j_t|x, y_a, y_b, j_{<t}).$$

Note that j contains r_θ's prediction of the preferred response $\hat{l} \in j$. The goal of reward modeling is :

$$\max_{r_\theta} \mathbb{E}_{(x,y_a,y_b,l)\sim\mathcal{D},\hat{l}\sim r_\theta(j|x,y_a,y_b)} \left[\mathbb{1}(\hat{l} = l)\right]. \tag{1}$$

3.2 Distillation of Reasoning Trace

For an instruct model, e.g., `Qwen-2.5-14b-instruct` [38], it is quite intuitive to turn it into a GenRM simply by prompting. However, without fine-tuning on reward modeling reasoning traces, these models may struggle to conduct consistent judgments. To bootstrap an instruct model's reasoning potential, we start with training an instruct model with long reasoning traces synthesized for reward modeling. Specifically, we subsample M data samples from $\mathcal{D}$ and denote it as $\mathcal{D}_{sub}$. Given a data sample $(x^{(i)}, y_a^{(i)}, y_b^{(i)}, l^{(i)}) \in \mathcal{D}_{sub}$, we ask an "oracle" model like `o3` or `claude-3-7-sonnet` to generate its structured reasoning trace $r^{(i)}$ justifying why $y_l^{(i)}$ is chosen as the correct answer of $x^{(i)}$. We then construct the reasoning trace ground truth:

$$y_{\text{trace}}^{(i)} = r^{(i)} \oplus y_l^{(i)}, \tag{2}$$

where $\oplus$ denotes string concatenation. Given all the synthesized reasoning traces $r^{(i)}$, the final distillation dataset is defined as:

$$\mathcal{D}_{\text{distill}} = \{(x^{(i)}, y_{\text{trace}}^{(i)})\}_{i=1}^{M}, \tag{3}$$

Formally, the objective of distillation is to adjust θ to maximize the likelihood of generating the desired reasoning trace and response y given the prompt x. We minimize the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{distill}}(\theta) = -\sum_{(x,y)\in\mathcal{D}_{\text{distill}}} \sum_{t=1}^{T} \log r_\theta\left(y_t \mid x, y_{<t}\right) \tag{4}$$

where $y_{<t} = (y_1, y_2, ..., y_{t-1})$ denotes the sequence of tokens preceding position t.

3.3 RL Training

Although distilling is a proper way to turn a general generative model into a GenRM, it often suffers from overfitting to certain patterns and constrains the model's ability to generalize its reasoning abilities for critical thinking [32] , which is essential for reward modeling. To address this, we propose to integrate RL as a more powerful learning paradigm to enhance the model's ability to conduct reward-based reasoning. Training a policy model using RL has been widely seen in the preference optimization phase of LLM post-training [25], and it is quite natural to extend this paradigm for training a REASRM. To be specific, we directly treat our reward model $r_\theta(j \mid x, y_a, y_b)$ as if it is a policy model:

$$\max_{r_\theta} \mathbb{E}_{(x,y_a,y_b,l)\sim\mathcal{D},\hat{l}\sim r_\theta(j|x,y_a,y_b)} \left[\mathcal{R}(x, j)\right] - \beta\mathbb{D}_{\text{KL}}\left(r_\theta \| r_{\text{ref}}\right), \tag{5}$$

where r_{ref} is the reference reward model. In practice, we use the checkpoint before training as r_{ref}, and that means r_{ref} could be an off-the-shelf LLM or the LLM obtained after the distillation step in Section 3.2. $\mathcal{R}(x, j)$ is the reward function, and $\mathbb{D}_{\text{KL}}$ is KL-divergence. The x denotes input prompts drawn from the preference data $\mathcal{D}$. The j indicates the text generated by the reward model, which includes the reasoning trace.

Chain-of-Rubrics (CoR) Rollout for Instruct Models

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.

First, classify the task into one of two categories: <type> Reasoning </type> or <type> Chat </type>.
- Use <type> Reasoning </type> for tasks that involve math, coding, or require domain knowledge, multi-step inference, logical deduction, or combining information to reach a conclusion.
- Use <type> Chat </type> for tasks that involve open-ended or factual conversation, stylistic rewrites, safety questions, or general helpfulness requests without deep reasoning.

If the task is Reasoning:
1. Solve the Client's question yourself and present your final answer within <solution> ... </solution> tags.
2. Evaluate the two Chatbot responses based on correctness, completeness, and reasoning quality, referencing your own solution.
3. Include your evaluation inside <eval> ... </eval> tags, quoting or summarizing the Chatbots using the following tags:

- <quote_A> ... </quote_A> for direct quotes from Chatbot A
- <summary_A> ... </summary_A> for paraphrases of Chatbot A
- <quote_B> ... </quote_B> for direct quotes from Chatbot B
- <summary_B> ... </summary_B> for paraphrases of Chatbot B

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

If the task is Chat:
1. Generate evaluation criteria (rubric) tailored to the Client's question and context, enclosed in <rubric>...</rubric> tags.
2. Assign weights to each rubric item based on their relative importance.
3. Inside <rubric>, include a <justify>...</justify> section explaining why you chose those rubric criteria and weights.
4. Compare both Chatbot responses according to the rubric.
5. Provide your evaluation inside <eval>...</eval> tags, using <quote_A>, <summary_A>, <quote_B>, and <summary_B> as described above.
6. End with your final judgment in the format: <answer>[[A]]</answer> or <answer>[[B]]</answer>

Important Notes:
- Be objective and base your evaluation only on the content of the responses.
- Do not let response order, length, or Chatbot names affect your judgment.
- Follow the response format strictly depending on the task type.

Figure 3: **The system prompt used for RM-R1 rollout** (for instruct models).

3.3.1 Chain-of-Rubrics Rollout

To facilitate the distilled models to proactively generate critical reasoning traces, we design a system prompt as shown in Figure 3 during rollout. Intuitively, reward modeling for general domain (chat, safety, etc.) and reasoning domain (math, code, etc.) should focus on different angles. For example, for the chat domain, we may care more about some aspects that can be expressed in textual rubrics (e.g., be polite), yet for the reasoning domain, we usually care more about logical coherence and answer correctness. Based on this intuition, we instruct r_θ to classify each preference data sample $\{(x, y_c, y_r)\}$ into one of the two <type>: **Chat** or **Reasoning**. For each <type>, we ask r_θ to practice different reasoning step by step: For **reasoning** tasks, we ask r_θ to solve x on its own. During the <eval> phase, r_θ compares y_c and y_r conditioned on its own </solution> and selects an <answer>. Regarding the **Chat** type, we instead ask r_θ to think about and justify the <rubric> for grading the chat quality (including safety). Since Chat scenarios do not have a verifiable reward, we instruct the model to select the <answer> based on its own <rubric>.

Large reasoning models such as `DeepSeek-R1-distilled` models [11] do not have a system prompt, so we show the user prompt for rollouts in Figure 4.

3.3.2 Reward Design

Rule-based reward mechanisms have demonstrated strong empirical performance to facilitate reasoning since DeepSeek-R1 [11], and have been commonly employed ever since. In our training,

Chain-of-Rubrics (CoR) Rollout for Reasoning Models

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

... [Pairwise Input Content] ...

Output your final verdict at last by strictly following this format: '<answer>[[A]]</answer>' if Chatbot A is better, or '<answer>[[B]]</answer>' if Chatbot B is better.

Figure 4: **The user prompt used for RM-R1 rollout** (for reasoning models).

we further simplify the reward formulation of DeepSeek-R1 and merely focus on correctness-based component, in line with prior works [30, 19]. We have also tried adding the format reward to the overall reward, but found that the task performance does not have a significant difference. The rationale behind only focusing on correctness is that the SFT models have already learned to follow instructions and format their responses properly. During experiments, we also observe that adding a format reward makes the advantage computation significantly slower, since regular expression matching grows linearly with the response token length.

Formally, our reward is defined as follows:

$$\mathcal{R}(x, j|y_a, y_b) = \begin{cases} 1, & \text{if } \hat{l} = l \\ -1, & \text{otherwise.} \end{cases} \tag{6}$$

3.3.3 Group Relative Policy Optimization (GRPO)

Group Relative Policy Optimization (GRPO) [30] is a variant of Proximal Policy Optimization (PPO) [28], which obviates the need for additional value function approximation, and uses the average reward of multiple sampled outputs produced in response to the same prompt as the baseline. More specifically, for each prompt x, GRPO samples a group of outputs $\{y_1, y_2, \cdots, y_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model by maximizing the following objective:

$$\begin{aligned} \mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\,\{j_i\}_{i=1}^{G}\sim r_{\theta_{\text{old}}}(j|x)} \Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|j_i|}\sum_{t=1}^{|j_i|}\Bigg\{ \min\Bigg(\frac{r_\theta(j_{i,t} \mid x, j_{i,<t})}{r_{\theta_{\text{old}}}(j_{i,t} \mid x, j_{i,<t})}\hat{A}_{i,t}, \\ \text{clip}\Big(\frac{r_\theta(j_{i,t} \mid x, j_{i,<t})}{r_{\theta_{\text{old}}}(j_{i,t} \mid x, j_{i,<t})}, 1-\epsilon, 1+\epsilon\Big)\hat{A}_{i,t}\Bigg) - \beta\,\mathbb{D}_{\text{KL}}\left[r_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\right] \Bigg\}\Bigg] \end{aligned} \tag{7}$$

where β is a hyperparameter balancing the task specific loss and the KL-divergence. Specifically, $\hat{A}_i$ is computed using the rewards of a group of responses within each group: $\{r_1, r_2, \ldots, r_G\}$, and is given by the following equation:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}. \tag{8}$$

4 Experiments

4.1 Experimental Setup

4.1.1 Benchmarks

In this paper, we consider the following three benchmarks:

RewardBench [18]: RewardBench is one of the first endeavors towards benchmarking reward models through prompt-chosen-rejected trios, covering four categories: chat, chat-hard, reasoning, safety, with 358. 456, 740. 1431 samples, respectively.

RM-Bench [23]: Building on RewardBench, RM-Bench evaluates reward models for their sensitivity to subtle content differences and robustness against style biases. It includes four categories: Chat, Safety, Math, and Code, with 129, 441, 529, and 228 samples, respectively. Each sample contains three prompts of varying difficulty. RM-Bench is the most reasoning-intensive benchmark among those we consider.

RMB [43]: Compared with RewardBench and RM-Bench, RMB offers a more comprehensive evaluation of helpfulness and harmlessness. It includes over 49 real-world scenarios and supports both pairwise and Best-of-N (BoN) evaluation formats. RMB comprises 25,845 instances in total—37 scenarios under the helpfulness alignment objective and 12 under harmlessness.

4.2 Preference Datasets

We consider the following datasets for training (A summary is presented in Appendix B):

Skywork Reward Preference 80K [21] is a high-quality collection of pairwise preference data drawn from a variety of domains, including chat, safety, mathematics, and code. It employs an advanced data filtering technique to ensure preference reliability across tasks. However, we identify a notable issue with this dataset: all samples from the `magpie_ultra` source exhibit a strong spurious correlation, where rejected responses consistently contain the token "`<im_start>`," while accepted responses do not. Additionally, responses from this source show a systematic bias—accepted responses are typically single-turn, while rejected responses are multi-turn. This problematic subset constitutes approximately 30% of the Skywork dataset and primarily covers mathematics and code domains. To avoid introducing spurious correlations into training, we exclude all `magpie_ultra` data and retain only the cleaned subset for our experiments.

Code-Preference-Pairs is a high-quality coding preference dataset. It is constructed by prompting a model with original code, introducing deliberate bugs, and manipulating examples (e.g., swapping broken and corrected versions, removing error comments) to generate fine-grained preference pairs. We subsample 8K examples from this dataset for use in our experiments.

Math-DPO-10K [17] is a high-quality stepwise preference dataset focused on mathematical reasoning. We use the full dataset in our experiments.

4.3 Baselines

We compare RM-R1 with RMs from three categories:

Scalar RMs. Scalar RMs produce a score for model response directly, predicting preference through single numeric scores without explicit reasoning traces. This category includes models such as RM [33], SteerLM-RM [37], Nemotron-RM [2], Starling-RM [44], ArmoRM [34], Skywork-RM [21], etc. These models typically show strong performance in clearly defined tasks but often lack interpretability and nuanced reasoning capability.

GenRMs. Generative reward models provide more nuanced feedback by generating textual judgments. Examples include Llama [9], Qwen [38], Claude [3], GPT-4 variants [1, 14], Gemini-1.5-pro [27] and Self-taught Evaluators [36]. GenRMs leverage generative capabilities of LLMs to produce explanations and rationales alongside numerical scores, enhancing interpretability and decision transparency.

ReasRMs. Reasoning-enhanced reward models explicitly incorporate reasoning processes into their judgments, often trained through critiques or chain-of-thought strategies. Notable examples are Critique-RM [39], DeepSeek-GRM [24] and our proposed RM-R1 models. These models excel in tasks demanding rigorous reasoning, safety evaluations, and nuanced preference judgments due to their grounding in structured critical thinking.

4.4 Implementation Details

Our training framework is based on verl [31] and OpenRLHF [13]. We use the training sets detailed in Section 4.3. For `Instruct` models, we use 8.7k data for distillation and 64k for RLVR. For `Deepseek-Distilled` models, we use the full data for RLVR. The full implementation details are provided in Appendix C.

Table 1: Results of our proposed method and baselines on the RewardBench. **Bold** numbers indicate the best performance, Underlined numbers indicate the second best.

Models	Chat	Chat_Hard	Safety	Reasoning	Overall
Scalar RMs					
SteerLM-RM 70B	91.3	80.3	92.8	90.6	88.8
Cohere-0514	96.4	71.3	92.3	97.7	89.4
Nemotron-4-340B-Reward	95.8	**87.1**	91.5	93.6	92.0
GenRMs					
Llama3.1-8B-Instruct	85.5	48.5	75.6	72.1	70.4
Prometheus-8*7B-v2	93.0	47.1	80.5	77.4	74.5
Llama3.1-70B-Instruct	**97.2**	70.2	82.8	86.0	84.0
Llama3.1-405B-Instruct	**97.2**	74.6	77.6	87.1	84.1
Claude-3-5-sonnet-20240620	96.4	74.0	81.6	84.7	84.2
GPT-4o-0806	96.1	76.1	86.6	88.1	86.7
Gemini-1.5-pro-0514	92.3	80.6	87.9	92.0	88.2
Self-taught-evaluator-llama3.1-70B	96.9	85.1	89.6	88.4	90.0
SFR-LLaMa-3.1-70B-Judge-r	96.9	84.8	91.6	97.6	92.7
REASRMS					
SynRM	38.0	82.5	74.1	87.1	70.4
RM-R1-DEEPSEEK-DISTILLED-QWEN-7B	88.9	66.2	78.4	87.0	80.1
CLoud	97.0	58.0	84.0	92.0	82.8
DeepSeek-GRM-16B	90.8	74.3	84.7	81.8	82.9
RM-R1-QWEN-INSTRUCT-7B	94.1	74.6	85.2	86.7	85.2
DeepSeek-GRM-27B	94.1	78.3	88.0	83.8	86.0
RM-R1-QWEN-INSTRUCT-14B	93.6	80.5	86.9	92.0	88.2
RM-R1-DEEPSEEK-DISTILLED-QWEN-14B	91.3	79.4	89.3	95.5	88.9
RM-R1-DEEPSEEK-DISTILLED-QWEN-32B	95.3	80.3	91.1	**96.8**	90.9
RM-R1-QWEN-INSTRUCT-32B	95.3	83.1	**91.9**	95.2	**92.9**

4.5 Main Results

For the baselines, we try our best to reproduce the numbers if essential resources are open-sourced (e.g., model checkpoints, system prompts). Otherwise, we use the numbers reported in the corresponding tech report or benchmark leaderboard. For each benchmark, we cutoff and select the best-performing models in each category for brevity.

Table 1, Table 2, and Table 3 report the performance of RM-R1 and baseline models on RewardBench, RM-Bench, and RMB, respectively. Our key findings are summarized below:

RM-R1 Achieves State-of-the-Art Performance. We first compare our two strongest models—RM-R1-QWEN-INSTRUCT-32B and RM-R1-DEEPSEEK-DISTILLED-QWEN-32B—against top-performing baselines. On Reward Bench, RM-R1-QWEN-INSTRUCT-32B achieves state-of-the-art performance among all existing generative reward models, outperforming closed-source commercial systems such as GPT-4o, Gemini-1.5-Pro, and models up to 340B in size. On RM-Bench, RM-R1-DEEPSEEK-DISTILLED-QWEN-32B surpasses the previous state-of-the-art by up to 12.8%, marking a significant improvement. On RMB, RM-R1-QWEN-INSTRUCT-32B attains near state-of-the-art performance with 73.0%, closely matching the best prior result of 73.8%. Notably, this model is trained on only 73K preference examples, demonstrating strong data efficiency.

Reasoning Training Demonstrates Effectiveness. Our reasoning-based training pipeline yields substantial gains. For example, RM-R1-QWEN-INSTRUCT-14B consistently outperforms DeepSeek-GRM-27B, a reasoning model nearly twice its size. We provide a detailed analysis of the key training design choices that contribute to this success in Section 5.2, and further highlight the advantages of reasoning-based training over non-reasoning counterparts in Section 5.3.

RM-R1 Generalizes Across Domains. Our generalist reward models based on Qwen-2.5-Instruct exhibit strong performance across all domains, including chat, safety, and reasoning, with no observable domain bias. Meanwhile, models trained from DeepSeek-Distilled-Qwen benefit from extensive reasoning-focused supervision and achieve stronger performance in reasoning-centric benchmarks. Specifically, RM-R1-DEEPSEEK-DISTILLED-QWEN-32B sets a new state-of-the-art on RM-Bench with

Table 2: The full results of tested reward models on RM-Bench. Chat, Math, Code, Safety show the model's Average Accuracy on each domain. Easy, Normal, Hard show the model's Accuracy on each difficulty level across all domains. **Bold** numbers indicate the best performance, Underlined numbers indicate the second best.

Models	Chat	Math	Code	Safety	Easy	Normal	Hard	Avg
Scalar RMs								
steerlm-70b	56.4	53.0	49.3	51.2	48.3	54.9	54.3	52.5
tulu-v2.5-70b-preference-mix-rm	58.2	51.4	55.5	87.1	72.8	65.6	50.7	63.0
Mistral-7B-instruct-Unified-Feedback	56.5	58.0	51.7	86.8	87.1	67.3	35.3	63.2
RM-Mistral-7B	57.4	57.0	52.7	87.2	88.6	67.1	34.9	63.5
Eurus-RM-7b	59.9	60.2	56.9	86.5	87.2	70.2	40.2	65.9
internlm2-7b-reward	61.7	71.4	49.7	85.5	85.4	70.7	45.1	67.1
GRM-llama3-8B-sftreg	62.7	62.5	57.8	90.0	83.5	72.7	48.6	68.2
internlm2-20b-reward	63.1	66.8	56.7	86.5	82.6	71.6	50.7	68.3
Llama-3-OffsetBias-RM-8B	71.3	61.9	53.2	89.6	84.6	72.2	50.2	69.0
Nemotron-340B-Reward	71.2	59.8	59.4	87.5	81.0	71.4	56.1	69.5
URM-LLaMa-3.1-8B	71.2	61.8	54.1	93.1	84.0	73.2	53.0	70.0
Skywork-Reward-Llama-3.1-8B	69.5	60.6	54.5	**95.7**	89.0	74.7	46.6	70.1
GenRMs								
tulu-v2.5-dpo-13b-chatbot-arena-2023	64.9	52.3	50.5	62.3	82.8	60.2	29.5	57.5
tulu-v2.5-dpo-13b-nectar-60k	56.3	52.4	52.6	73.8	86.7	64.3	25.4	58.8
stablelm-2-12b-chat	67.2	54.9	51.6	65.2	69.1	63.5	46.6	59.7
tulu-v2.5-dpo-13b-stackexchange-60k	66.4	49.9	54.2	69.0	79.5	63.0	37.2	59.9
Nous-Hermes-2-Mistral-7B-DPO	58.8	55.6	51.3	73.9	69.5	61.1	49.1	59.9
tulu-v2.5-dpo-13b-hh-rlhf-60k	68.4	51.1	52.3	76.5	53.6	63.0	69.6	62.1
tulu-2-dpo-13b	66.4	51.4	51.8	85.4	86.9	66.7	37.7	63.8
SOLAR-10.7B-Instruct-v1.0	**78.6**	52.3	49.6	78.9	57.5	67.6	69.4	64.8
REASRMS								
RM-R1-QWEN-INSTRUCT-7B	66.6	67.0	54.6	92.6	79.2	71.7	59.7	70.2
RM-R1-DEEPSEEK-DISTILLED-QWEN-7B	64.0	83.9	56.2	85.3	75.9	73.1	68.1	72.4
RM-R1-QWEN-INSTRUCT-14B	75.6	75.4	60.6	93.6	82.6	77.5	68.8	76.1
RM-R1-QWEN-INSTRUCT-32B	75.3	80.2	66.8	93.9	86.3	80.5	70.4	79.1
RM-R1-DEEPSEEK-DISTILLED-QWEN-14B	71.8	90.5	69.5	94.1	86.2	83.6	74.4	81.5
RM-R1-DEEPSEEK-DISTILLED-QWEN-32B	74.2	**91.8**	**74.1**	95.4	**89.5**	**85.4**	**76.7**	**83.9**

91.8% in math and 74.1% in code, outperforming previous best models by substantial margins (66% in math and 59% in code). It also achieves the strongest reasoning performance among our released models on RewardBench. However, in terms of data efficiency, our `Instruct`-based models reach competitive reasoning performance and superior overall results using only 8.7K examples for distillation—compared to the 800K examples used in training `DeepSeek-Distilled`.

Additional Analysis. Across all benchmarks, we observe that larger models consistently achieve higher performance. This trend is not coincidental. In Section 5.1, we conduct a detailed analysis and show that reasoning reward models benefit more from increased model scale, reinforcing the effectiveness of scaling in this setting. We also conduct more fine-grained analysis of the training dynamics and case study of RM-R1 in Section 5.4 and 5.5.

5 Analysis

In this section, we present a series of empirical studies and ablation experiments aimed at understanding the key ingredients for training effective reasoning reward models. Our analysis spans scaling effects, design decisions, reasoning ablations, training dynamics, and case studies.

5.1 Scaling Effects

We begin by investigating how model performance varies with scale, considering both **model size** and **inference-time compute**. While prior work has shown that larger models generally achieve lower pretraining loss with stronger capabilities [15], and recent studies highlight the benefits of increased inference budgets [15], reward modeling presents a more nuanced picture. In some cases—such as

Table 3: The leaderboard of RMB, ranked by the average score of all subsets. **Bold** numbers indicate the best performance, Underlined numbers indicate the second best.

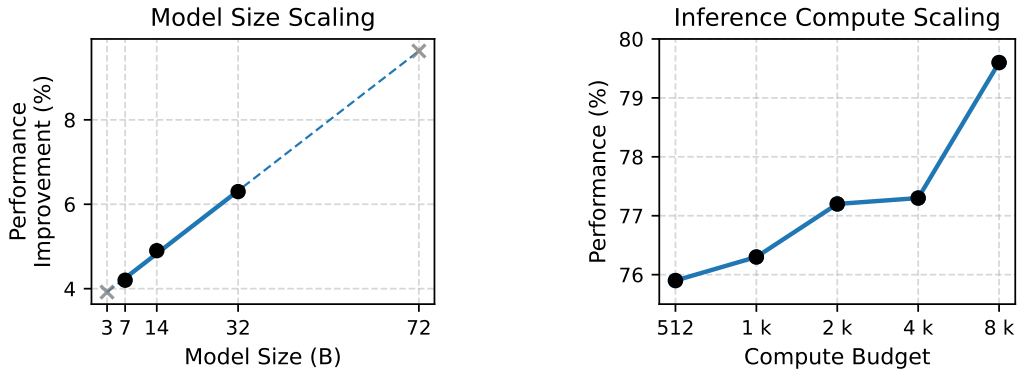
Models	Helpfulness		Harmlessness		Overall
	BoN	Pairwise	BoN	Pairwise	
Scalar RMs					
Tulu-v2.5-13b-preference-mix-rm	0.355	0.562	0.351	0.545	0.453
Skywork-Reward-Gemma-2-27B	0.472	0.653	0.561	0.721	0.602
Internlm2-20b-reward	0.585	0.763	0.499	0.670	0.629
ArmoRM-Llama3-8B-v0.1	0.636	0.787	0.497	0.663	0.646
Internlm2-7b-reward	0.626	0.782	0.563	0.712	0.671
Eurus-RM-7b	0.679	0.818	0.543	0.693	0.683
Skywork-Reward-Llama-3.1-8B	0.627	0.781	0.603	0.759	0.693
Starling-RM-34B	0.604	0.774	0.674	0.795	0.712
GenRMs					
Llama2-70b-chat	0.289	0.613	0.249	0.602	0.438
Llama3.1-8B-Instruct	0.365	0.675	0.267	0.653	0.490
Gemini-1.5-pro	0.536	0.763	0.299	0.661	0.565
Mixtral-8x7B-Instruct-v0.1	0.480	0.706	0.491	0.671	0.587
skywork-critic-llama3.1-8B	0.600	0.725	0.578	0.578	0.620
skywork-critic-llama3.1-70B	0.640	0.753	0.614	0.614	0.655
Llama3.1-70B-Instruct	0.648	0.811	0.558	0.739	0.689
Mistral-Large-2407	0.678	0.817	0.583	0.725	0.701
Claude-3-5-sonnet	**0.705**	**0.838**	0.518	0.764	0.706
Qwen2-72B-Instruct	0.645	0.810	0.649	0.789	0.723
GPT-4o-2024-05-13	0.639	0.815	**0.682**	**0.814**	**0.738**
REASRMS					
RM-R1-DEEPSEEK-DISTILLED-QWEN-7B	0.451	0.658	0.429	0.664	0.551
RM-R1-QWEN-INSTRUCT-7B	0.543	0.740	0.608	0.765	0.664
Deepseek-GRM-27B-RFT	0.592	0.801	0.548	0.765	0.670
RM-R1-DEEPSEEK-DISTILLED-QWEN-14B	0.593	0.765	0.613	0.769	0.685
Deepseek-GRM-27B	0.623	0.805	0.570	0.761	0.690
RM-R1-QWEN-INSTRUCT-14B	0.594	0.776	0.620	0.778	0.692
RM-R1-DEEPSEEK-DISTILLED-QWEN-32B	0.620	0.782	0.618	0.771	0.698
RM-R1-QWEN-INSTRUCT-32B	0.636	0.791	**0.682**	0.809	0.730

scalar-valued sequence models from Skywork [21]—the performance gap between the small model (*e.g.*, 8b) and the large model (*e.g.*, 27b) is minimal (often less than 1%), showing no clear advantage from scaling. In this subsection, we show that this trend does not hold for reasoning reward models, where scaling brings clear and substantial improvements.

5.1.1 Model Sizes

We now analyze the impact of model scale. Our study is based on the `Qwen-2.5-Instruct` model family at three sizes: 7B, 14B, and 32B. We evaluate performance improvements resulting from our training procedure described in Section 3, with results averaged across three key benchmarks: RewardBench, RM-Bench, and RMB.

For each model size, we compare the original and post-training performance. Figure 5a plots the absolute improvement with respect to the original model performance (in %) as a function of model size. Observing an approximately linear trend, we fit a linear regression model and extrapolate to hypothetical scales of 3B and 72B, shown using faint markers and dashed extensions. The results strongly support a scaling law for reasoning reward models (RM-R1): larger models not only result in an absolute better final performance but also consistently yield **greater performance gains**. This aligns with the intuition that our training effectively leverages the superior reasoning capabilities of larger models. Furthermore, these findings provide compelling evidence that reinforcement learning benefits from scale as well.



(a) Model Size

(b) Inference Compute

Figure 5: Scaling effect of RM-R1. We find that (a) larger models benefit more from reasoning training, and (b) longer reasoning chains result in better reward modeling performance.

5.1.2 Inference-time Computation

Next, we examine the effect of inference-time computation—how model performance varies with different compute budgets (measured in number of tokens allowed during inference). Since this is particularly relevant to reasoning-focused models, we fix our base model to `DeepSeek-R1-Distill-Qwen-14B`. We evaluate average performance across the three key benchmarks using a wide range of inference-time compute budgets: 512, 1024, 2048, 4096, and 8192 tokens.

To ensure a fair comparison, we match the training compute to the inference budget in each setting (*i.e.*, we allow a maximum of x tokens during training for a compute budget of x at inference). All models are trained using GRPO with identical datasets and hyperparameter configurations. Figure 5b shows the relationship between compute budget and performance. We observe a clear improvement trend as the inference budget increases. This highlights the benefits of long reasoning chains in reward modeling.

⭐ Takeaway 1:

Scaling improves reward model performance: we observe a near-linear trend with both model size and inference-time compute. Larger models consistently benefit more from our reasoning-based training pipeline, and longer reasoning chains become increasingly effective under higher compute budgets.

5.2 Training Recipes

We next investigate the key ingredients underlying the successful training of RM-R1. Through a series of ablation studies, we examine our design choices to identify effective strategies for training high-quality reasoning reward models. We compare the following settings:

Cold Start RL. Prior work on reasoning model training typically adopts a straightforward pipeline based on reinforcement learning with verifiable rewards [5, 24]. This approach generally involves pure RL, with rule-based rewards centered on answer correctness and format compliance. Such strategies have achieved notable success in advanced mathematical problem solving.

In this setting, we replicate this conventional training setup. Specifically, we use a combination of a format reward and an answer reward:

$$\mathcal{R}_{\text{format}} = \begin{cases} 1 & \text{if format matches} \\ 0 & \text{otherwise} \end{cases} \qquad \mathcal{R}_{\text{answer}} = \begin{cases} 1 & \text{if answer matches} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The total reward is the sum $\mathcal{R} = \mathcal{R}_{\text{answer}} + \mathcal{R}_{\text{format}}$. We use the prompt template shown in Figure 7, a simplified version of Figure 3. In this setup, all input prompts are treated uniformly—that is, chat and reasoning tasks are not distinguished.

Cold Start RL + Query Categorization (QC). This setting largely follows the previous one, with a key modification: prompting the LM to first categorize the task into reasoning or chat, and then apply different strategies for handling those tasks. Intuitively, reinforcement learning alone can effectively explore reasoning tasks, a domain where it has already achieved considerable success. Here, we incorporate the system prompt shown in Figure 3, which explicitly distinguishes between chat and reasoning tasks.

For reasoning tasks specifically, we note that answer quality is closely tied to correctness, and that high-level rubrics may be less effective than simply evaluating whether the model can solve the problem and verify its own answer. Thus, this setting emphasizes correctness-based evaluation guided by task classification in the prompt.

Distilled + RL + QC (RM-R1). Building on the previous setup, we introduce an additional distillation stage from stronger teacher models as a warm start before RL training. The motivation is that with RL alone, weaker models (especially at smaller scales) often **fail to explore high-quality rubrics** and convincing reasoning chains for chat tasks throughout the RL training process. Distilling strong reasoning traces on a small subset of data can effectively mitigate this limitation.

We now expand on the **details of generating high-quality reasoning chains**. We first use the same prompt to query `Claude-3.7-Sonnet`, generating initial reasoning traces. However, approximately 25% of these traces are incorrect, primarily on harder chat tasks. To correct these cases, we pass the original prompt, the incorrect trace, and the correct final answer to `OpenAI-O3`, which then generates a corrected reasoning trace aligned with the right answer.

This two-stage process yields a high-quality distillation set. We deliberately choose the order—first `Claude`, then `O3`—based on qualitative observations: `Claude` excels at solving easier tasks and maintaining attention to safety considerations, whereas `O3` performs better on harder helpfulness tasks but tends to overemphasize helpfulness at the expense of safety. We select approximately 12% of the training data (slightly fewer than 9K examples) for distillation. This is then followed by RL training.

Table 4: Ablation Study of the Designs of the `Qwen-2.5-Instruct-32B` on Reward Bench.

Method	Chat	Chat Hard	Safety	Reasoning	Reward Bench
Instruct (Original)	**95.8**	74.3	86.8	86.3	85.8
Instruct + Cold Start RL	93.8	78.1	90.3	90.6	88.2
Instruct + Cold Start RL + QC	95.0	78.1	88.9	**95.8**	89.8
Instruct + Distilled + RL + QC	95.3	**83.1**	**91.9**	95.2	**92.9**

In Table 4, we present the results of the ablation studies described above, using the `Qwen-2.5-Instruct-32B` model as the Instruct (Original) model. Several key conclusions emerge:

- **RL training alone is insufficient.** While Cold Start RL slightly improves performance on hard chat and reasoning tasks, it fails to close the gap to fully optimized models.
- **Query Categorization significantly boosts reasoning performance.** Incorporating explicit query categorization into the prompt notably improves reasoning performance, suggesting that clearer task guidance benefits learning.
- **Distillation further enhances performance across all axes.** Seeding the model with high-quality reasoning traces before RL yields the strongest results, with improvements observed on both hard tasks and safety-sensitive tasks.

⭐ Takeaway 2:

Directly replicating reinforcement learning recipes from mathematical tasks is insufficient for training strong reasoning reward models. Explicit query categorization and targeted distillation of high-quality reasoning traces are both crucial for achieving robust and generalizable improvements.

5.3 Effectiveness of Reasoning Training

We now analyze the impact of reasoning-based training. Throughout this subsection, we use **SFT**to refer to supervised fine-tuning on final answers only, and **distilled** to refer to supervised fine-tuning on reasoning chains generated by a stronger model. Prior work has shown that SFT alone can yield substantial performance gains [21]. Here, we demonstrate that reasoning-based training can outperform such answer-only approaches. We consider the following experimental settings:

Instruct + SFT. This approach fine-tunes the instruct model directly toward producing the correct final answer using the full dataset, without providing any intermediate reasoning chains.

Instruct + Distilled + SFT. This approach applies SFT (with respect to the answer directly) after the distillation stage, serving as a direct comparison point with RM-R1.

Instruct + RM-R1 (Distilled + RL). This is the full approach proposed in this paper, following the procedure detailed in Section 3 and Section 5.2.

Instruct + Distilled. This setting uses the model checkpoint immediately after the distillation stage, before any RL fine-tuning.

Table 5: Comparison of reasoning-based training versus SFT, measured across the three key benchmarks. Methods marked with * involve reasoning training. We can find that reasoning training consistently results in a better-performing reward model.

Method	RewardBench	RM-Bench	RMB	Average
Full Data				
Instruct + SFT	90.9	75.4	65.9	77.4
Instruct + Distilled + SFT	91.2	76.7	65.4	77.8
Instruct + RM-R1 *	92.9	79.1	73.0	81.7
9k Data				
Instruct + SFT	88.8	74.8	66.9	76.6
Instruct + Distilled*	89.0	76.3	72.0	79.2

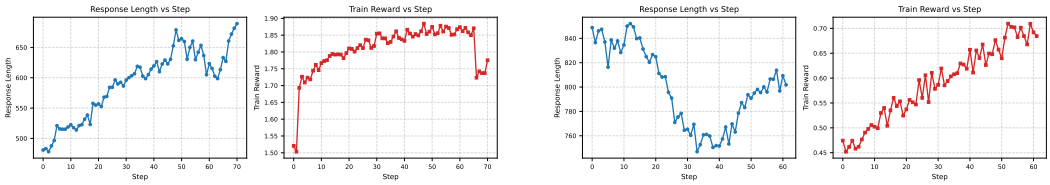
In summary, both (+ RM-R1) and (+ Distilled) represent reasoning-based approaches, while the remaining methods are purely non-reasoning-based approaches. In Table 5, we report the results measured across the three benchmarks. The findings clearly demonstrate that reasoning training significantly benefits reward model performance. Under fair comparisons (i.e., training on exactly the same amount of data), reasoning-based models consistently outperform their SFT-only counterparts. In particular, even high-quality distillation alone, applied to a small subset of the data, provides notable gains, highlighting the value of incorporating structured intermediate reasoning.

⭐ Takeaway 3:

Reasoning training substantially improves reward modeling. It not only enables better generalization across tasks but also provides consistent gains even under limited data scenarios compared to direct-answer SFT approaches.

5.4 Training Dynamics

We analyze the training dynamics of RM-R1 using the `Qwen-2.5-14B-Instruct` model by tracking both response length and reward progression throughout RL training. We consider two settings: (a)



(a) Cold Start RL

(b) Warm Start RL

Figure 6: RL training dynamics under different settings: (a) Cold Start RL (Eq. 9) and (b) Warm Start RL (Eq. 6). In Cold Start RL, the response length steadily increases as the model learns to reason, but training becomes unstable near the end. In Warm Start RL, the model exhibits more stable training, with effective refinement of reasoning traces throughout the process.

Cold Start RL, and (b) Warm Start RL following reasoning-chain distillation. We present the finding in Figure 6.

In the Cold Start RL setting, we observe that the model gradually learns to reason, as reflected by a steady increase in response length over the course of training. However, training becomes unstable near the end, with a sharp drop in the reward curve, suggesting potential issues such as overfitting.

In contrast, under Warm Start RL, the model begins with stronger initial reasoning abilities, exhibiting longer responses from the outset. Interestingly, the model first learns to produce more concise reasoning traces before gradually increasing response length again as training progresses. The reward curve rises smoothly and consistently throughout training, demonstrating more stable and efficient learning compared to the Cold Start setting.

5.5 Case Study

To gain deeper insights into RM-R1, we conduct a case study using `Qwen-14B-Instruct`, comparing its behavior with a cold-start RL model. The results, summarized in Table 6, reveal the following key observations:

Transparent Judging Process. The reasoning traces generated by RM-R1 are highly interpretable and coherent, reflecting the model's perception of human preferences. It explicitly articulates why certain responses are better, providing transparency into its evaluation process.

High-Quality, Question-Dependent Rubrics. RM-R1 accurately understands the question and the context of comparison, correctly prioritizing "accuracy" as the most critical rubric for medical-related questions. In contrast, the cold-start RL model often overlooks the most important factors and instead emphasizes superficial or broadly defined features (e.g., relevance) that are less discriminative. The ability to generate high-quality, question-specific rubrics stems from the knowledge acquired during the distillation stage.

Faithful Adherence to Rubrics and Content-Based Judgement. RM-R1 grounds its evaluation in the actual content of the chatbot responses. For example, it correctly identifies inaccuracies in chatbot A's response based on factual content rather than surface presentation. Furthermore, it systematically evaluates different aspects of the rubric, leading to a structured, interpretable, and verifiable judging process.

6 Conclusion and Future Work

In this paper, we revisited reward modeling through the lens of reasoning. We introduced RM-R1, a family of REASRMS that effectively generate explicit chains of rubrics and rationales, and scale with both model size and inference compute. Across three public benchmarks, RM-R1 matched or surpassed commercial and open-source RMS while producing more interpretable judgments. Ablation investigations reveal that (1) task-type classification, (2) bootstrapping from high-quality reasoning traces, and (3) RL fine-tuning are all indispensable. Qualitative analyses further showed

Question: symptoms of sickle-cell disease

Chatbot A:
1. Painful red or yellow skin lesions; 2. Frequent infections; 3. Fatigue and weakness; 4. Anemia; 5. Slower growth and development; 6. Jaundice; 7. Swelling of hands and feet; 8. Vision loss; 9. Tiredness and weakness after physical activity; 10. Breathlessness; 11. Fever, chills, and night sweats.

Chatbot B
Sickle cell disease is a genetic disorder that affects the shape of red blood cells. It is most common in African Americans, but can also affect people from other ethnic groups. The symptoms of sickle cell disease can vary widely from person to person, but some common symptoms include:

1. Fatigue: People with sickle cell disease often experience fatigue and weakness, even when they have not been physically active.; 2. Pain: Sickle cell disease can cause pain in the body, particularly in the arms, legs, back, and abdomen. Pain can be mild or severe and can last for days or weeks.; 3. Fever: People with sickle cell disease may be prone to infections and may experience frequent fevers.; 4. Anemia: Sickle cell disease can cause anemia, a condition in which the body does not have enough red blood cells to carry oxygen throughout the body.; 5. Delayed growth and development: Children with sickle cell disease may be smaller than their peers due to delayed growth and development.; 6. Infections: People with sickle cell disease are at increased risk for infections, particularly in the bloodstream and urinary tract.; 7. Pigmentation: Sickle cell disease can cause dark areas of pigmentation in the body, particularly on the hands and feet.

It's important to note that not everyone with sickle cell disease will experience all of these symptoms, and some people may not experience any symptoms at all. Therefore, it's important to consult with a healthcare provider if you suspect you or someone you know may have sickle cell disease.

Cold-start RL evaluation
<type>Chat</type>
<rubric> Relevance 40 % – Comprehensiveness 30 % – Clarity/organisation 30 % </rubric>
<eval> Chatbot A provided a list of symptoms that are directly relevant to sickle cell disease. The response is comprehensive and includes a wide range of symptoms commonly associated with the condition. The symptoms are listed in a clear and organized manner, making it easy for the client to understand the information. Chatbot B's response is less organized than Chatbot A's response, and the explanation of sickle cell disease may be unnecessary for the client's question.</eval>
<answer>[[A]]</answer>

RM-R1 evaluation
<rubric> 1 Accuracy 40 % · 2 Comprehensiveness 30 % · 3 Clarity 20 % · 4 Helpfulness 10 % <justify> Accuracy is weighted highest because providing correct information about medical conditions is crucial. Comprehensiveness is important because ...</justify> </rubric>
<eval> *Accuracy of information:* Chatbot A provides a list of symptoms without explanation, but some listed items (e.g. "painful red or yellow skin lesions," "vision loss") are not typical of sickle-cell disease. Chatbot B offers more accurate information, explaining each symptom's nature and severity. ... *Comprehensiveness:...Clarity and organization:...Helpfulness:...* </eval>
<answer>[[B]]</answer>

Table 6: Case Study of RM-R1 (succinct version). Complete model responses are shown in Table 7 and Table 8. Cold-Start RL evaluation tends to focus on superficial features (*e.g.,* number of suggestions, relevance), whereas RM-R1 produces high-quality rubrics (*e.g.,* accuracy), faithfully follows them, and evaluates responses based on their actual content.

that RM-R1 learns to prioritize high-impact rubrics, faithfully follow its own criteria and justify coherently.

Future work includes automatic rubric induction, where an explicit rubric library can be reused and composed, so that rollout length could be reduced; Another direction is active preference collection, where REASRMS use active learning to query human preference only when the current rubric set is insufficient for a new preference sample. Finally, it would be natural to extend our study to multimodal/agentic reward modeling scenarios.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024.

[3] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1:1, 2024.

[4] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[5] Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. Judgelrm: Large reasoning models as a judge. *arXiv preprint arXiv:2504.00050*, 2025.

[6] Xiusi Chen, Jyun-Yu Jiang, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Wei Wang. Minprompt: Graph-based minimal prompt data augmentation for few-shot question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 254–266, 2024.

[7] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[8] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Bofei Gao, Zefan Cai, Runxin Xu, Peiyi Wang, Ce Zheng, Runji Lin, Keming Lu, Dayiheng Liu, Chang Zhou, Wen Xiao, et al. Llm critics help catch bugs in mathematics: Towards a better mathematical verifier with natural language feedback. *arXiv preprint arXiv:2406.14024*, 2024.

[11] 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 arXiv:2501.12948*, 2025.

[12] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

[13] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[14] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[15] 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. *arXiv preprint arXiv:2001.08361*, 2020.

[16] Martin Klissarov, Devon Hjelm, Alexander Toshev, and Bogdan Mazoure. On the modeling capabilities of large language models for sequential decision making. *arXiv preprint arXiv:2410.05656*, 2024.

[17] Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

[18] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.

[19] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

[20] 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 *The Twelfth International Conference on Learning Representations*, 2023.

[21] Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

[22] Ollie Liu, Deqing Fu, Dani Yogatama, and Willie Neiswanger. Dellma: Decision making under uncertainty with large language models. *arXiv preprint arXiv:2402.02392*, 2024.

[23] 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. *arXiv preprint arXiv:2410.16184*, 2024.

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

[25] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[26] 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. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[27] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[29] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[30] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

[32] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? *Advances in neural information processing systems*, 34:6906–6919, 2021.

[33] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[34] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.findings-emnlp.620`.

[35] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

[36] Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*, 2024.

[37] Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. HelpSteer: Multi-attribute helpfulness dataset for SteerLM. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3371–3384, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.naacl-long.185`.

[38] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[39] Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, et al. Self-generated critiques boost reward modeling for language models. *arXiv preprint arXiv:2411.16646*, 2024.

[40] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

[41] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL `https://openreview.net/forum?id=uccHPGDlao`.

[42] Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*, 2025.

[43] Enyu Zhou, Guodong Zheng, Binghai Wang, Zhiheng Xi, Shihan Dou, Rong Bao, Wei Shen, Limao Xiong, Jessica Fan, Yurong Mou, et al. Rmb: Comprehensively benchmarking reward models in llm alignment. *arXiv preprint arXiv:2410.09893*, 2024.

[44] Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, November 2023.

[45] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Chain -of -Rubrics (CoR) Roll-out for Instruct Models
(no categorization of task types)

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.

Instructions

1. Begin your evaluation by generating the rubric criteria tailored to the Client's question and context. Enclose the rubric in <rubric> . . . </rubric> tags.
2. Assign weights to each rubric item based on their relative importance.
3. Within <rubric>, include a <justify> . . . </justify> section explaining the rationale behind the chosen criteria and weights.
4. Compare both Chatbot responses using the rubric.
5. Include your evaluation in <eval> . . . </eval> tags.
   Support your analysis using:
   - <quote_A> . . . </quote_A> for direct quotes from Chatbot A
   - <summary_A> . . . </summary_A> for paraphrased summaries of Chatbot A
   - <quote_B> . . . </quote_B> for direct quotes from Chatbot B
   - <summary_B> . . . </summary_B> for paraphrased summaries of Chatbot B
6. Conclude with your final judgment using:
   <answer>[[A]]</answer> or <answer>[[B]]</answer>

Important Notes:

- Be objective and base your evaluation strictly on the content of the responses.
- Do not let the response order, length, or Chatbot names bias your judgment.

Figure 7: The system prompt of the ablation study on cold start RL without categorization of task types.

A Supplementary Information for Section 5

Cold-start RL evaluation
<type>Chat</type>
<rubric>
The evaluation criteria for the responses are as follows:
- <item>Relevance to the question</item> (40%): The response should directly address the symptoms of sickle cell disease.
- <item>Comprehensiveness</item> (30%): The response should include a wide range of symptoms commonly associated with sickle cell disease.
- <item>Clarity and organization</item> (30%): The response should be easy to read and understand, with symptoms listed in a clear and organized manner.
<justify>
The question specifically asks for the symptoms of sickle cell disease, so the responses should be directly relevant to this topic. Comprehensiveness is important because sickle cell disease can have a variety of symptoms, and a comprehensive list will provide a more complete picture of the condition. Clarity and organization are important because the symptoms can be complex, and a well-organized response will make it easier for the client to understand the information.
</justify>
</rubric>

<eval>
<quote_A>
1. Painful red or yellow skin lesions
2. Frequent infections
3. Fatigue and weakness
4. Anemia
5. Slower growth and development
6. Jaundice
7. Swelling of hands and feet
8. Vision loss
9. Tiredness and weakness after physical activity
10. Breathlessness
11. Fever, chills, and night sweats.
</quote_A>
<summary_A>
Chatbot A provided a list of symptoms that are directly relevant to sickle cell disease. The response is comprehensive and includes a wide range of symptoms commonly associated with the condition. The symptoms are listed in a clear and organized manner, making it easy for the client to understand the information.
</summary_A>
<quote_B>
Sickle cell disease is a genetic disorder that affects the shape of red blood cells. It is most common in African Americans, but can also affect people from other ethnic groups. The symptoms of sickle cell disease can vary widely from person to person, but some common symptoms include:
1. Fatigue: People with sickle cell disease often experience fatigue and weakness, even when they have not been physically active.
2. Pain: Sickle cell disease can cause pain in the body, particularly in the arms, legs, back, and abdomen. Pain can be mild or severe and can last for days or weeks.
3. Fever: People with sickle cell disease may be prone to infections and may experience frequent fevers.
4. Anemia: Sickle cell disease can cause anemia, a condition in which the body does not have enough red blood cells to carry oxygen throughout the body.
5. Delayed growth and development: Children with sickle cell disease may be smaller than their peers due to delayed growth and development.
6. Infections: People with sickle cell disease are at increased risk for infections, particularly in the bloodstream and urinary tract.
7. Pigmentation: Sickle cell disease can cause dark areas of pigmentation in the body, particularly on the hands and feet.
</quote_B>
<summary_B>
Chatbot B provided a detailed explanation of sickle cell disease and a list of symptoms. The response is relevant to the question and includes a wide range of symptoms commonly associated with the condition. However, the response is less organized than Chatbot A's response, and the explanation of sickle cell disease may be unnecessary for the client's question.
</summary_B>
</eval>

<answer>[[A]]</answer>

Table 7: Cold-Start RL Judges. The rubrics and evaluation mainly center around superficial shallow features (*e.g.,* number of points listed) without actually taking their accuracy into account.

B Training Dataset Details

A global statistics of our training dataset is summarized in Table 9.

Cold-start RL evaluation
<type>Chat</type>
<rubric>
1. Accuracy of information (40 %): How correctly does the response describe the symptoms of sickle-cell disease?
2. Comprehensiveness (30 %): How thoroughly does the response cover the range of symptoms associated with sickle-cell disease?
3. Clarity and organization (20 %): How well-structured and easy to understand is the information presented?
4. Helpfulness for the user (10 %): How useful is the information for someone seeking to understand sickle-cell disease symptoms?
<justify>
These criteria focus on evaluating how well each response addresses the user's question about sickle-cell disease symptoms. Accuracy is weighted highest because providing correct information about medical conditions is crucial. Comprehensiveness is important, as sickle-cell disease has multiple symptoms that vary in severity and presentation. Clarity and organization affect how easily the user can understand the information. Helpfulness considers how practical and actionable the information is for the user.
</justify>
</rubric>

<eval>
Comparing the two responses:

Accuracy of information:
<quote_A>1. Painful red or yellow skin lesions
2. Frequent infections
3. Fatigue and weakness
4. Anemia
5. Slower growth and development
6. Jaundice
7. Swelling of hands and feet
8. Vision loss
9. Tiredness and weakness after physical activity
10. Breathlessness
11. Fever, chills, and night sweats.</quote_A>

<quote_B>1. Fatigue: People with sickle-cell disease often experience fatigue and weakness, even when they have not been physically active.
2. Pain: Sickle-cell disease can cause pain in the body, particularly in the arms, legs, back, and abdomen. Pain can be mild or severe and can last for days or weeks.
3. Fever: People with sickle-cell disease may be prone to infections and may experience frequent fevers.
4. Anemia: Sickle-cell disease can cause anemia, a condition in which the body does not have enough red blood cells to carry oxygen throughout the body.
5. Delayed growth and development: Children with sickle-cell disease may be smaller than their peers due to delayed growth and development.
6. Infections: People with sickle-cell disease are at increased risk for infections, particularly in the bloodstream and urinary tract.
7. Pigmentation: Sickle-cell disease can cause dark areas of pigmentation in the body, particularly on the hands and feet.</quote_B>
Chatbot A provides a list of symptoms without explanation, but some listed items (e.g. "painful red or yellow skin lesions," "vision loss") are not typical of sickle-cell disease. Chatbot B offers more accurate information, explaining each symptom's nature and severity.
Comprehensiveness:
Chatbot A lists 11 symptoms, several of which are inaccurate or uncommon. Chatbot B lists 7 symptoms that are more representative and supplies contextual detail about variability and medical advice.
Clarity and organization:
Chatbot A's plain numbered list lacks context, making severity unclear. Chatbot B structures the list with explanations, improving readability and understanding.
Helpfulness:
Chatbot A is less helpful due to inaccuracies and missing context. Chatbot B is more useful, providing accurate descriptions, context, and guidance.
</eval>

<answer>[[B]]</answer>

Table 8: RM-R1 RL Judges. The rubrics and evaluation are well-designed and reasonable, assessing the effectiveness of the model response.

C Implementation Details

Our training pipeline consists of two stages: distillation and RLVR (Reinforcement Learning with Verifiable Rewards).

Distillation Stage. We use the `SFTTrainer` from OpenRLHF with a fixed batch size of 128 and a micro-batch size of 1, training for a single epoch. To optimize GPU memory usage, we enable gradient checkpointing, FlashAttention, and Adam offloading. The learning rates are set based on model size: $5e{-}6,\ 3e{-}6,\ 2e{-}6$ for models of size $7B,\ 14B,\ 32B$ respectively.

Source	Size	Domain
magpile_pro_llama3.1*	29682	Reasoning
offset_bias*	8504	Chat (length bias)
helpsteer2*	7221	Chat
wildguard*	6709	Safety
magpile_pro*	2030	Chat
Code-Preference-Pairs	8000	Reasoning
Math-DPO-10K	10000	Reasoning

Table 9: Global Statistics of our Training Dataset. * indicates the source is from Skywork-Reward-Preference-80K-v0.2.

RLVR Stage. We use the VERL framework for all GRPO training. The training batch size is fixed at 1024, with a mini-batch size of 128. We adopt Fully Sharded Data Parallel (FSDP) to improve memory efficiency. For rollout generation, we use vLLM with tensor parallelism size 4 and GPU memory utilization capped at 0.4. Sampling follows default parameters (temperature = 1.0, top-p = 1.0). KL regularization is applied with a coefficient of $1e{-}3$ and a clip ratio of 0.2. Each prompt is sampled with 7 candidate responses.

The maximum input sequence length is 4,096 tokens, and the maximum response length is 8,192 tokens. Learning rates are set separately for the two model variants:

- Instruct models: $1e{-}6,\ 7e{-}7,\ 5e{-}7$ for 7B, 14B, 32B models.
- Reasoning models: $1e{-}6,\ 1e{-}6,\ 8e{-}7$ for 7B, 14B, 32B models.

We train the 7B, 14B, 32B models on $1,\ 2,\ 4$ nodes respectively, each equipped with 8 H100 GPUs.