RLLM: Unified Post-Training via On-Policy-Trained Language Model as a Reward Model

Chenxi Whitehouse [†], **Swarnadeep Saha**[†], **Ping Yu**, **Ilia Kulikov**, **Jason Weston**

[1]FAIR at Meta
[†]Equal Contribution

Post-training for LLMs typically follows one of two paradigms: Reinforcement Learning from Human Feedback (RLHF), which relies on scalar reward models trained from human preference data, or Reinforcement Learning with Verifiable Rewards (RLVR), which depends on rule-based verifiers. Scalar reward models do not generate chain-of-thought reasoning, making them prone to reward hacking and limiting their effectiveness on complex reasoning tasks. Rule-based verifiers, meanwhile, assume access to gold answers that can be both hard-to-obtain and hard-to-verify, limiting their utility to easily-verifiable math and code problems. We show that **RLLM**, **R**einforcement **L**earning with an **LM** itself as a Reward Model (RM), can serve as a single, unified post-training recipe for *easy-to-verify*, *hard-to-verify*, as well as *non-verifiable* domains. RLLM applies RL in two stages: (1) training an LM with verifiable rewards to act as a thinking Reward Model, and (2) post-training a policy-LM using the LM-as-RM's k-wise comparative judgments as rewards. We first demonstrate that RLLM outperforms RLHF (with scalar RMs) and RLVR (with rule-based verifiers) across easy-to-verify and hard-to-verify math and physics benchmarks as well as non-verifiable instruction-following tasks. We then show that on-policy training of the LM-as-RM outperforms both prompted LMs-as-RMs (including a larger GPT-OSS-120B) and off-policy trained ones. Finally, through extensive analyses across a wide range of policy–reward LM pairings – varying in model size, capability, and training data (easy- vs. hard-to-verify, reference-free vs. reference-based tasks) – we identify the key ingredients for effective post-training with Language Models as Reward Models.

Correspondence: chenxwh@meta.com, swarnadeep@meta.com

Meta

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performance across a wide variety of tasks, from general instruction following to complex reasoning (Guo et al., 2025a; Yang et al., 2025; Hurst et al., 2024; Grattafiori et al., 2024). A key factor behind this progress is the post-training stage that relies on Reinforcement Learning (RL) to align models to human preferences. Traditionally, this alignment has been achieved through Reinforcement Learning from Human Feedback (RLHF), where human preference data is used to train a scalar reward model that assigns single numerical scores to guide optimization (Ouyang et al., 2022). However, scalar reward models are limiting: they produce a final score without generating intermediate reasoning steps, making them mostly ineffective for improving reasoning capabilities and susceptible to issues such as reward hacking and poor generalization to out-of-distribution (OOD) scenarios (Gao et al., 2023).

More recently, with the advancement of o1 (Jaech et al., 2024) and R1-style thinking LLMs (Guo et al., 2025a), a new paradigm has emerged: Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2024). As illustrated in Figure 1, RLVR replaces the scalar reward model with rule-based verifiers (e.g., code compilers or math checkers) to provide grounded feedback. While RLVR offers verifiable correctness, it strictly requires access to "gold" answers that are often hard-to-obtain or hard-to-verify. This limits its utility primarily to domains with objective correctness criteria—such as simple math and coding—leaving a gap for complex reasoning tasks where answers are not easily checked by rules, or subjective tasks where no gold answer exists (Tao et al., 2025).

These limitations motivate a third paradigm: RLLM (Reinforcement Learning with an LM itself as a Reward Model). Unlike scalar models that lack deeper reasoning, or rule-based verifiers that lack flexibility, RLLM

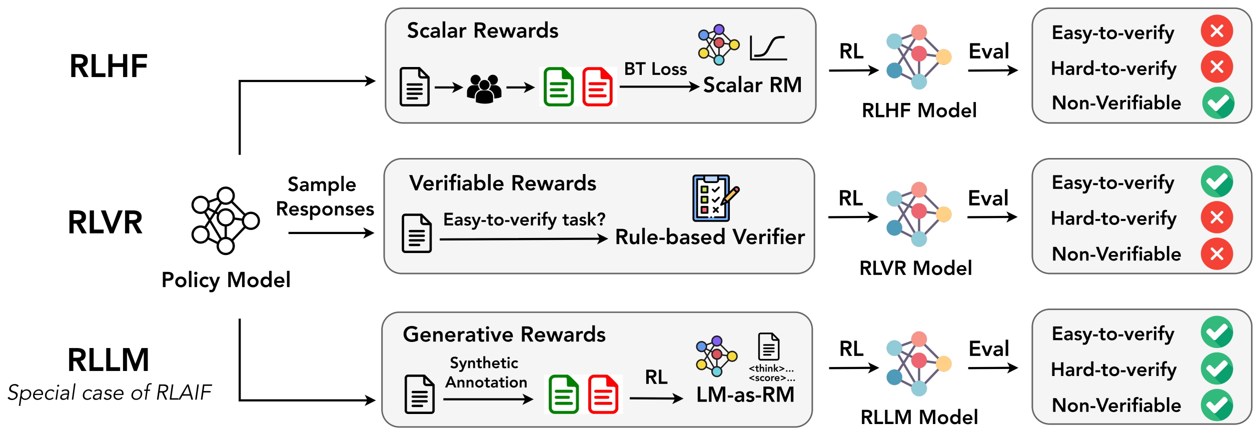


Figure 1 Comparison of RLHF, RLVR, and RLLM for post-training LLMs. RLHF optimizes a policy against a scalar reward model trained on human-annotated preference data using the Bradley-Terry objective. RLVR trains a policy using a rule-based verifier and hence is restricted to easy-to-verify tasks with ground-truth labels. RLLM (ours) is a special case of RL-from-AI-Feedback (RLAIF) that first trains an LM-as-RM on synthetic judgments using RL and then uses its generative rewards to optimize the policy. An LM-as-RM exploits an LLM's (1) reasoning capabilities to produce higher-quality reward signals and (2) instruction-following capabilities to allow flexible reward design. Thus, RLLM unifies the post-training paradigm, enabling the policy model to excel across easy-to-verify, hard-to-verify, and non-verifiable tasks.

leverages the inherent capabilities of the LLM itself to serve as a "thinking" Reward Model. The effectiveness of this approach stems from an LLM's two core strengths: the model's *reasoning capabilities* enable it to generate explicit thinking traces for higher-quality judgments, while its *instruction-following capabilities* allow for flexible reward design through natural language prompts. Consequently, RLLM serves as a single, unified framework effective across easy-to-verify, hard-to-verify, and non-verifiable domains.

Our proposed method operates in two distinct stages. First, in the **Thinking LM-as-RM Training** stage, we train an LLM to act as a thinking Reward Model. Crucially, we employ an on-policy training recipe: the RM is trained to evaluate responses sampled specifically from the policy model, using synthetic labels derived from a stronger teacher or verifier. Unlike scalar RMs, this LM-as-RM generates an explicit reasoning trace to justify its judgment before assigning a score. Second, in the **Policy Post-Training** stage, we use the generative rewards produced by our trained LM-as-RM to optimize the policy model via reinforcement learning.

We validate RLLM through extensive experiments on diverse benchmarks, including math, physics (Principia), and open-ended instruction following (AlpacaEval, ArenaHard). We demonstrate that RLLM significantly outperforms both RLHF (with SOTA scalar RMs) and RLVR (with rule-based verifiers). Notably, RLLM achieves large gains on hard-to-verify tasks, such as math or physics problems where standard verifiers fail, while simultaneously maintaining strong performance on non-verifiable creative tasks. Furthermore, we identify two critical ingredients for success: (i) the necessity of a sufficient *generator-verifier gap* (using a larger RM to train a smaller policy), and (ii) the importance of *on-policy* RM training, as we show that off-policy RMs struggle to provide accurate signals for downstream improvements.

Our contributions are summarized as follows:

- We introduce RLLM, a unified post-training framework that employs a "thinking" Language Model as a Reward Model, improving upon the limited rule-based verifiers (required by RLVR) or uninterpretable scalar scoring (used in RLHF).
- We demonstrate that RLLM outperforms leading RLHF and RLVR baselines for 1.7B scale models, achieving significant gains on both verifiable tasks (Math, Physics) and non-verifiable tasks (AlpacaEval, ArenaHard).
- We provide a comprehensive analysis of the recipe required for effective LM-as-RM training, empirically proving that on-policy training and a strong generator-verifier capability gap are essential for successful

policy improvement.

2 Preliminaries

RLHF: Reinforcement Learning with Human Feedback. The standard RLHF (Ouyang et al., 2022; Bai et al., 2022a) pipeline trains a *scalar* reward model on pairwise human preference data. Each data point $(x, y_c, y_r) \in \mathcal{D}$ includes an instruction x, a chosen response y_c, and a rejected response y_r. The reward model $r_\phi(x, y)$ is optimized using a Bradley-Terry objective:

$$\mathcal{L}_R = -\mathbb{E}_{(x,y_c,y_r)\in\mathcal{D}}[\log\sigma(r_\phi(x, y_c) - r_\phi(x, y_r))], \tag{1}$$

where $\sigma(\cdot)$ is the logistic sigmoid. This trains the model to assign higher rewards to the chosen responses. The trained scalar reward model is then used to optimize a policy model via Reinforcement Learning (e.g., PPO (Schulman et al., 2017)). In this paper, we use the term *RLHF* to specifically refer to the classical setup of InstructGPT (Ouyang et al., 2022), where a *scalar* reward model is trained from human preference data and thereafter, used to optimize a policy model.

RLVR: Reinforcement Learning with Verifiable Rewards. Scalar reward models remain vulnerable to reward hacking, also known as reward over-optimization (Amodei et al., 2016; Gao et al., 2023; Eisenstein et al., 2024), especially under distributional shift, and have shown limited ability to drive improvements on challenging reasoning problems. Thus, in an attempt to prevent such hacking, RLVR replaces the reward model in RLHF with a verification function (Lambert et al., 2024) such that the policy only receives a reward when its generated responses are verifiably correct. Given an instruction x, a candidate response y, and a reference answer y_{ref}, the verification function $\psi(\cdot)$ is defined as:

$$\psi(x, y, y_{\text{ref}}) = \begin{cases} \gamma, & \text{if correct, i.e., } y \text{ is equivalent to } y_{\text{ref}}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

These verifiers are typically rule-based (e.g., `math-verify` for math or unit tests for coding) but can also be model-based (e.g., a model that checks equivalence between two mathematical expressions). RLVR has shown great success on tasks with verifiable outcomes such as mathematical problem-solving (Shao et al., 2024) and verifiable instruction-following tasks (Lambert et al., 2024). However, its reliance on high-quality reference answers that are also easy-to-verify, makes its application limited to easy-to-verify math and coding tasks.

3 RLLM: Reinforcement Learning with Language Models as Reward Models

To reduce dependence on costly human preference collection, a third paradigm – Reinforcement Learning from AI Feedback (RLAIF) – was introduced (Bai et al., 2022b; Lee et al., 2024) where an AI model is itself used to provide rewards. Our **RLLM** paradigm is a special case of RLAIF (see Figure 1), which employs a Language Model as a *thinking* Reward Model (LM-as-RM) to unify post-training across diverse task types.

In contrast to scalar reward models $r_\phi(\cdot)$ used in RLHF or deterministic verifiers $\psi(x, y, y_{\text{ref}})$ used in RLVR, RLLM utilizes *generative rewards* $r_{\text{LM}}(\cdot)$ obtained directly from a thinking LLM. This unifies the post-training recipe and allows the flexibility of computing rewards through pointwise, pairwise, or listwise judgments, in both reference-free $r_{\text{LM}}(x, y)$ and reference-based $r_{\text{LM}}(x, y_{\text{ref}}, y)$ settings, depending on the task requirements described below.

We consider the standard RL objective for maximizing expected reward:

$$\max_{\pi_{\theta_{\text{policy}}}} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi(\cdot|x)}\left[r_{\text{LM}}(x, y)\right] - \beta\mathbb{D}_{\text{KL}}(\pi_{\theta_{\text{policy}}}||\pi_{\text{ref}}) \tag{3}$$

where $\pi_{\theta_{\text{policy}}}$ is the policy, π_{ref} is the reference model, and β controls the KL-divergence penalty. Importantly, RLLM uses RL both to train an LLM as an RM and to optimize the policy using rewards produced by this LM-as-RM.

3.1 Task Settings and LM-as-RM Variants

We consider a general post-training setting where LLMs are expected to improve across *Verifiable* (e.g., Math, Code) and *Non-Verifiable* tasks (e.g., Open-ended Chat). Verifiable tasks are those with objective correctness criteria while non-verifiable tasks are inherently subjective and lack definitive, externally verifiable references (Lu, 2025). Within verifiable tasks, we further differentiate between *easy-to-verify* problems (gradeable via simple rules like `math-verify`) and *hard-to-verify* problems (requiring intermediate reasoning to check equivalence). Given this set of tasks, we define the following LM-as-RM reward formulations:

Reference-Free Pointwise LM-as-RM: $r_{LM}(x, y) \to (t, s)$. In a reference-free pointwise setting, the LM-as-RM assigns rewards to individual responses independently. Given an instruction x and a candidate response y, the LLM generates a formatted judgment consisting of (i) a reasoning trace t enclosed within <think> tags, where the model analyzes the quality of y, and (ii) a final scalar score s within <score> tags.

Reference-Based Pointwise LM-as-RM: $r_{LM}(x, y_{\text{ref}}, y) \to (t, s)$. In the reference-based setting, the LM-as-RM is additionally provided with a ground-truth reference y_{ref} and instructed to ground its judgment in this solution. This enables robust evaluation for both easy-to-verify and hard-to-verify problems that require substantial reasoning, effectively allowing the model to act as a *generative verifier*.

Pairwise and Listwise LM-as-RM: $r_{LM}(x, \mathbf{y}) \to (t, \mathbf{s})$ or $r_{LM}(x, y_{\text{ref}}, \mathbf{y}) \to (t, \mathbf{s})$. Pointwise evaluation can be extended to *pairwise* or more generally, to *listwise* evaluation, where the LM-as-RM jointly evaluates multiple candidate responses. In this setting, the input consists of the instruction x and a set of k candidate responses $\mathbf{y} = \{y_1, \ldots, y_k\}$. The model jointly reasons about the candidates in t and outputs a list of scores $\mathbf{s} = \{s_1, \ldots, s_k\}$. Such comparative evaluation has been shown to substantially improve zero-shot LLM-judge performance, particularly for non-verifiable tasks (Whitehouse et al., 2025a).

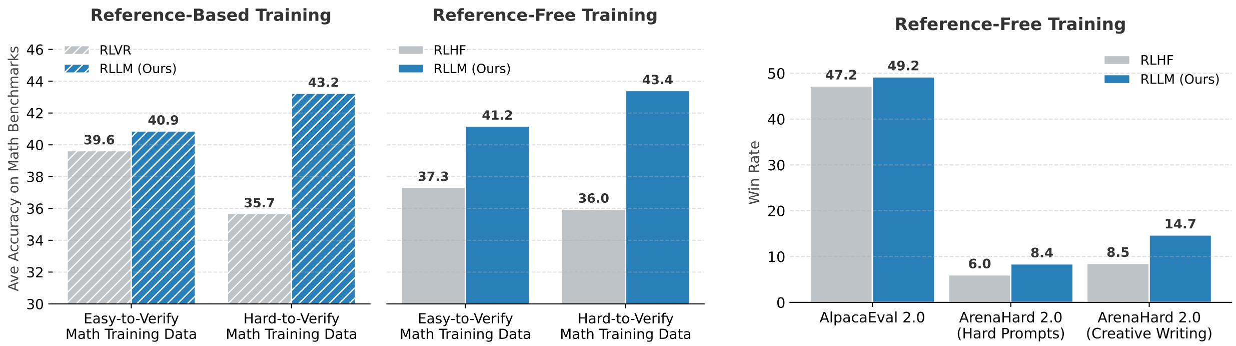
In Appendix A, we provide the prompt template for reference-free pointwise LMs-as-RMs in Figure 7, and pairwise in Figure 8. In Figure 9 we show the prompt template for reference-based pointwise LMs-as-RMs.

3.2 LM-as-RM Training via RLVR

Here we first describe how we use RL to train an LM-as-RM. While off-the-shelf LLMs can also serve as zero-shot RMs, their judgments often become unreliable when policy outputs drift out-of-distribution or when the *generator–verifier gap* is insufficient to provide trustworthy signals (Shao et al., 2025). To obtain more reliable and task-aligned reward signals, we thus train an LM-as-RM via Reinforcement Learning with Verifiable Rewards using the `J1` framework (Whitehouse et al., 2025b). `J1` constructs synthetic judgment tasks with labels, converting diverse tasks into a unified verifiable format compatible with RLVR-style training.

Synthetic Training Data Generation. Let $\pi_{\theta_{\text{policy}}}$ denote the initial policy LLM that we want to optimize using an LM-as-RM. To train the LM-as-RM, we first sample *on-policy* responses from $\pi_{\theta_{\text{policy}}}$ and synthetically annotate the responses for the reward modeling task. Specifically, given a dataset $\mathcal{D}$ with instructions x and optionally available reference answers y_{ref}, we generate training data in three steps: (i) sample a set of responses $\mathbf{y}$ from the policy $\pi_{\theta_{\text{policy}}}$; (ii) employ a stronger teacher LLM to rate the correctness or quality of these responses, obtaining scores $\mathbf{s}$. For mathematical reasoning tasks, these ratings are typically binary (correct/incorrect); for non-verifiable tasks, the scores span a continuous scale $[s_{\min}, s_{\max}]$ reflecting response quality. Finally, we create a balanced dataset to ensure a uniform distribution over the assigned scores. The resulting synthetic dataset for LM-as-RM training is denoted as: $\mathcal{D}_{\text{LM}} = \big\{(x, y_{\text{ref}}, \mathbf{y}, \mathbf{s}) \mid (x, y_{\text{ref}}) \in \mathcal{D}\big\}$, where $y_{\text{ref}} = \emptyset$ in reference-free settings. This would enable us to train an LM-as-RM in both reference-free or reference-based scenarios.

RLVR Training. Given the scores, we now train the LM-as-RM using RLVR. Following Whitehouse et al. (2025b), we format examples from the synthetic dataset $\mathcal{D}_{\text{LM}}$ into seed LM-as-RM prompts (depending on the training configuration), and instruct the model to generate a judgment (t', s'), consisting of a reasoning trace t' and a predicted score s'. We optimize the model using GRPO (Shao et al., 2024), assigning a reward of 1 if the predicted score matches the teacher score, and 0 otherwise.



(a) Verifiable Tasks (Math) **(b)** Non-Verifiable Tasks

Figure 2 Performance comparison of post-trained Qwen-1.7B models on (a) verifiable tasks (average of five math benchmarks) and (b) non-verifiable instruction-following tasks. Models are trained via RLHF (with `Skywork-Reward-V2-Llama-3.1-8B` as scalar-RM), RLVR (with `Math-Verify` as rule-based verifier) and, our RLLM (with `J1-Qwen3-32B` as LM-as-RM). Post-training data for verifiable tasks is either (1) easy-to-verify, (2) hard-to-verify, (3) reference-free, or (4) reference-based. Across all these settings, RLLM achieves consistently higher accuracy and win rates than RLVR and RLHF, with particularly large gains when trained on hard-to-verify problems.

3.3 RLLM Policy Training with LM-as-RM

Given an LM-as-RM $\pi_{\theta_{\text{LM}}}$, either an off-the-shelf LLM or a trained one (as described above), we optimize the policy model $\pi_{\theta_{\text{policy}}}$ using the LM-as-RM's generative rewards. All LM-as-RM variants generate thinking traces and scores; however, only the scores are used for policy optimization. In particular, we sample rollouts $y_i \sim \pi_{\theta_{\text{policy}}}(\cdot \mid x)$ and depending on whether the LM-as-RM is a pointwise, pairwise, or listwise model and if reference answers are available or not, we compute rewards from the rollouts as follows:

Pointwise RLLM. In this setting, the reward for each rollout y_i is computed independently as $r_i = r_{\text{LM}}(x, y_i)$ (or with reference y_{ref}). This approach is most computationally efficient, providing an absolute quality score without requiring comparisons with other rollouts.

Pairwise RLLM. Pairwise comparison requires constructing pairs of rollouts to be evaluated jointly by the LM-as-RM. Let $\mathbf{y} = \{y_1, \ldots, y_n\}$ denote the n rollouts for a given prompt. We consider two pairwise strategies and both evaluate the ordered responses (y_a, y_b) and (y_b, y_a) to reduce positional bias (Zheng et al., 2023).

- **Exhaustive Pairwise.** We compare each rollout with every other rollout, resulting in all $\binom{n}{2}$ ordered comparisons. Each rollout appears in $2(n-1)$ comparisons, and its final reward is obtained by averaging the scores across these comparisons. This provides the strongest comparative signal but has $O(n^2)$ judgment complexity.
- **Pivot Pairwise.** To reduce the quadratic complexity of all-pairs, we randomly choose one rollout y_{pivot} as a reference and pair it with every other rollout in both orders, resulting in $2(n-1)$ comparisons: $(y_a,\, y_{\text{pivot}})$, $(y_{\text{pivot}},\, y_a)$. All non-pivot rollouts receive scores relative to the same anchor, thus producing more consistent rewards compared to randomly constructing $O(n)$ pairs. The pivot rollout's reward is obtained by averaging the scores across all the comparisons. This setting remains lightweight, with $O(n)$ LM-as-RM evaluations.

Listwise RLLM. Similar constructions can be extended to a list of $K > 2$ comparisons, where the LM-as-RM jointly evaluates sets of K rollouts rather than pairs, while increasing the reward modeling complexity.

The policy and the LM-as-RM may be initialized from the same or different base LLMs; when they coincide, this corresponds to a self-rewarding RLLM setting. Our experiments consider both cases.

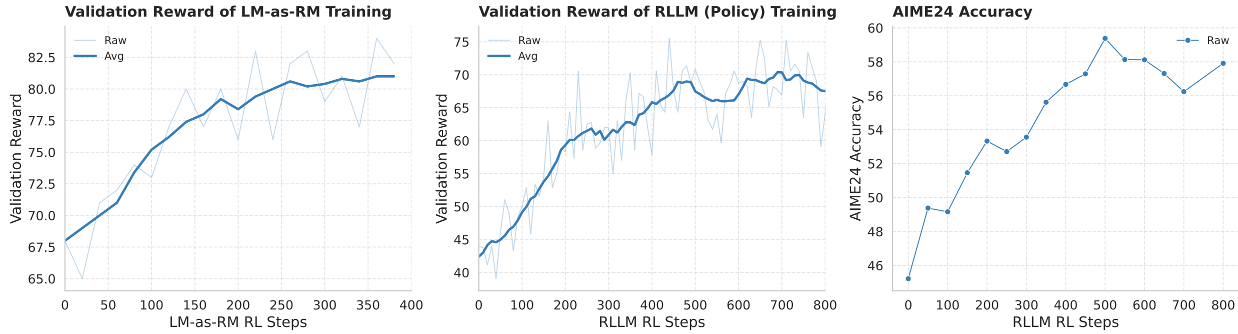


Figure 3 RL validation rewards for LM-as-RM training (left) and RLLM policy training (center), alongside the downstream AIME24 accuracy of the policy model (right). Centered moving average is shown to highlight the trend.

4 Experimental Setup

4.1 LM-as-RM Models

Training Recipe. For the purpose of our main experiments, we use the `J1` recipe to train *on-policy* LM-as-RM models starting from a *large* `Qwen3-32B` (thinking) model. As we will show later in our analysis (subsection 5.2), (1) training a large RM such that it ensures a substantial generator-verifier gap and (2) training the RM on-policy are both critical for best downstream policy improvements. Recall that LM-as-RM training can be done in both reference-free and reference-based settings, in which case we will call the resultant RMs `J1-Qwen3-32B-RM` and `J1-Qwen3-32B-Verifier`, respectively. We train these `J1` models by (1) sourcing hard-to-verify prompts from OpenMathReasoning and Principia Collection, (2) generating 16 responses per prompt from `Qwen3-1.7B` model (i.e., the policy model to be trained), and (3) labeling each response as correct or incorrect using `GPT-OSS-120B` (since the final answers are hard-to-verify). See Figure 10 and Figure 11 for two such training examples of correct and incorrect responses. This results in a synthetically annotated and balanced dataset of 18,774 examples for pointwise LM-as-RM training with verifiable correctness rewards. For the purposes of ablations, whenever we train a weaker LM-as-RM model with off-policy samples, we will follow a similar `J1` recipe with the exception that either the base model will change or the responses will be sampled from a different model than the policy to be aligned.

Training Hyperparameters. Following Whitehouse et al. (2025b), we implement all `J1` models on top of `verl` (Sheng et al., 2024). We use a train batch size of 512 with a maximum sequence length of 4096 for both input and output. We also set the KL coefficient to 0.001, the number of rollouts to 8, and the sampling temperature to 1.0 during RL training. All 32B `J1` models are trained using 128 H200 GPUs. The best `J1` checkpoint is chosen based on a held-out validation set of 100 samples.

Inference Hyperparameters. In addition to using `J1` models during policy training, we also perform offline evaluation of these models on in-distribution validation data. Inference is done using vLLM (Kwon et al., 2023) with a temperature of 0.6 and top_p value of 0.95.

4.2 RLLM Policy Models

Training Recipe. In our primary RLLM experiments, we optimize a `Qwen3-1.7B` (Instruct) policy using the aforementioned `J1-Qwen3-32B` reward models.[1] This choice of model sizes for the policy and the RM creates a substantial generator–verifier gap, which we find to be essential for achieving strong downstream policy improvements. Starting with the same `Qwen3-1.7B` policy, we experiment with four different RLLM training paradigms that vary in their training data or the LM-as-RM:

- **Policy Trained on Hard-to-Verify Samples without Reference Answer.** This forms the main experimental

[1] Qwen instruct models are already post-trained using RL, making them harder to improve on, compared to their base counterparts. Despite that, we conduct all RLLM experiments on top of such strong instruct models and show further improvements.

setting of RLLM where we assume access to a post-training dataset of only hard-to-verify math problems without any reference answers. As LMs continue to grow in capability, we will increasingly confront tasks for which no human-curated reference answers exist. To test such a post-training setting, we sample 3000 *hard-to-verify* problems from the OpenMathReasoning dataset (Moshkov et al., 2025). Following Tao et al. (2025), we select a sample as hard-to-verify if the final answer cannot be deterministically validated using a rule-based symbolic verifier like `math_verify` (see example in Appendix Figure 12) and easy-to-verify, otherwise (see example in Appendix Figure 13). Given the reference-free setting, our RLLM recipe uses `J1-Qwen3-32B-RM` as the reward model for optimization.

- **Policy Trained on Hard-to-Verify Samples with Reference Answer.** This resembles an RLLM setting where we train on the same dataset as above but this time, assuming access to reference answers. Since these reference answers are hard-to-verify, our RLLM recipe uses `J1-Qwen3-32B-Verifier` as the (reference-based) verifier for policy training.
- **Policy Trained on Easy-to-Verify Samples with/without Reference Answer.** Our motivation to perform RLLM on *easy-to-verify* samples is to show that even when reference answers exist and they are easy-to-verify, RLLM with a strong LM-as-RM can substitute rule-based verifiers like `math-verify`. Note that an LM-as-RM has the advantage of additionally evaluating the entire CoT and identifying *process* errors, unlike `math-verify` that only performs equivalence checks between the final answers. For the purpose of this study, we conduct RLLM experiments by sampling 3000 *easy-to-verify* training examples from the same OpenMathReasoning dataset (Moshkov et al., 2025).

In subsequent analyses and ablations of RLLM, we also consider different combinations of policy and reward models e.g., other sizes of Qwen3 models, Octothinker (Wang et al., 2025), and Llama models. For non-verifiable instruction-following tasks, we train RLLM models on 1K Wildchat prompts (Zhao et al., 2024).

Training Hyperparameters. We implement all policy models on top of `fairseq2` (Balioglu et al., 2023). Models are trained using 64 H200 GPUs, allocating 48 GPUs for the trainer and 16 GPUs for inference. Fairseq2's trainer is implemented as `Single Program Multiple Data (SMPD)` and all models (policy model, reference model, and `J1` reward model) run as Ray actors. RLLM training uses a batch size of 32 with 8 rollouts per prompt, sampled with a temperature of 1.0. Similar to `J1` training, we also set the KL coefficient to 0.001 for policy training. RLLM models are trained for a maximum of 1000 steps, checkpointing every 50 steps. To mitigate the effect of mismatch in log probabilities between the trainer and the inference engine in GRPO training, we apply truncated importance sampling correction with the maximum clip ratio set to 2.0. We also set max input and generation length to 4096 tokens and train `Qwen3-1.7B` in non-thinking mode to prevent long thinking sequences from exceeding the context window, ensuring that the reward models can evaluate the complete answers.

Inference Hyperparameters. We evaluate RLLM-trained policy models using vLLM with a temperature of 0.6 and a top_p value of 0.95. The maximum decoding length is set to 40K tokens. Given the hybrid nature of Qwen3 models, we evaluate our post-trained models in both thinking and non-thinking modes.

4.3 Evaluation Benchmarks and Metrics

To evaluate the effectiveness of RLLM as a general post-training recipe, we conduct experiments on both verifiable reasoning tasks as well as non-verifiable tasks.

Verifiable Reasoning Benchmarks. Within verifiable reasoning, we experiment with both easy-to-verify benchmarks (e.g., questions with numerical answers) as well as hard-to-verify benchmarks (e.g., questions with mathematical objects as answers like equations).

- **Easy-to-verify Math Benchmarks.** We evaluate on five popular competition math benchmarks from MathArena (Balunović et al., 2025) – AIME24, AIME25, BRUMO25, HMMT24, and HMMT25. For each benchmark, we report `mean@16` results.
- **Hard-to-verify Physics Benchmark.** We also evaluate on the Physics subset of PrincipiaBench (). This serves as a test bed for both hard-to-verify reasoning problems as well as our models' generalization to a different domain (given that RLLM's training data only consists of math prompts). Following (), we use o3 (OpenAI, 2025) to evaluate the correctness of the final answers and report `mean@8` scores.

Table 1 Comparison of different post-trained Qwen3-1.7B (Instruct) models using RLLM or RLHF on easy-to-verify and hard-to-verify reasoning benchmarks. All models are trained on hard-to-verify samples in a reference-free setting. RLHF'ed models are optimized using SOTA scalar RMs. RLLM models are optimized using either prompted LM-as-RM or our trained J1 LM-as-RM. We observe improved RLLM results by scaling up the LM-as-RM, with J1-Qwen3-32B-RM improving AIME24 by 12% on top of a Qwen3-1.7B (Instruct) model.

Method	RM	RM Type	RM Size	Math (Easy-to-Verify)						Hard-to-Verify
				AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**	**Physics**
`Qwen3-1.7B`	–	–	–	45.22	36.20	49.78	23.30	21.44	35.20	15.57
Policy Trained on hard-to-verify samples w/o Reference Answer										
RLHF	`Nexusflow/Athene-RM-8B`	Scalar	8B	40.22	32.92	38.95	17.08	19.58	29.75	13.18
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	48.33	36.26	47.91	22.92	24.36	35.96	16.93
RLHF	`nvidia/AceMath-7B-RM`	Scalar	7B	48.76	38.34	47.69	23.12	22.71	36.12	16.24
RLLM	`Llama-3.1-8B-Instruct`	Generative	8B	48.54	36.68	46.47	22.51	24.18	35.67	15.68
RLLM	`Qwen3-1.7B`	Generative	1.7B	51.25	37.71	51.67	26.70	26.88	38.84	17.84
RLLM	`Qwen3-32B`	Generative	32B	54.38	43.53	53.76	26.70	**31.02**	41.88	16.48
RLLM	`GPT-OSS-120B`	Generative	120B	52.09	39.79	52.91	33.30	28.34	41.29	16.36
RLLM	`J1-Qwen3-32B-RM`	Generative	32B	**57.91**	**44.17**	**54.16**	**33.30**	27.50	**43.41**	**18.75**
Δ w/ Qwen3-1.7B	–	–	–	+12.59	+7.97	+4.38	+10.0	+6.06	+8.21	+3.18

Non-verifiable Tasks. Finally, we also evaluate RLLM on two non-verifiable instruction following benchmarks – AlpacaEval 2.0 (Li et al., 2023) and ArenaHard 2.0 (Li et al., 2025). Following past work (Lanchantin et al., 2025), we report win rates (with and without length-control), using GPT-4o and GPT-4.1 as the evaluators for AlpacaEval and ArenaHard respectively.

4.4 Baselines

We compare RLLM to two groups of baselines, RLHF and RLVR.

- **RLHF.** This represents RL post-training with *scalar* reward models. In particular, we consider two strong general-purpose RMs – `Skywork-Reward-V2-Llama-3.1-8B` (Liu et al., 2025a)[2] and `Athene-RM-8B` (Frick et al., 2024). We also compare against a math-specific reward model `AceMath-7B-RM` (Liu et al., 2024b).
- **RLVR.** This refers to RL post-training with a verifier. The verifier operates in the presence of a reference answer and can either be rule-based (e.g., `math-verify`) or model-based (e.g., `general-verifier` (Ma et al., 2025). Even though `general-verifier` is a model-based verifier, it only checks the equivalence between the final answers (without any CoT reasoning). We compare RLLM to both these RLVR methods.

Within the RLLM framework, we compare our `J1`-trained LMs-as-RMs to various prompted LLMs-as-RMs. These belong to different families and are of different sizes and capabilities, thus enabling us to analyze how scaling up the RM/Verifier affects policy training and downstream task improvements. In particular, we report RLLM results with `Llama-3.1-8B-Instruct`, `Qwen-1.7B`, `Qwen-32B`, and `GPT-OSS-120B` as prompted LMs-as-RMs.

5 Results

5.1 Main Results

RLLM (with prompted and trained LMs-as-RMs) outperforms RLHF (with scalar RMs) on both easy-to-verify and hard-to-verify reasoning tasks. First, in Table 1, we compare different post-trained Qwen3-1.7B models, optimized via either scalar RMs (RLHF) or LM-as-RM (RLLM). Within RLLM, we compare our trained J1-Qwen3-32B-RM to different prompted LMs-as-RMs of varying sizes. All models are trained on the same hard-to-verify math prompts in a reference-free setting. Our main conclusions are listed below:

- All RLLM models (rows annotated in green), except for the one trained with the weaker Llama-3.1-8B-Instruct model, outperform all RLHF models, showcasing the effectiveness of LMs-as-RMs over scalar RMs (36.12 → 43.41). Importantly, while scalar RMs like Skywork-Reward-V2 may be the

[2]This is the best-performing publicly available reward model, according to the RewardBench2 leaderboard: `https://huggingface.co/spaces/allenai/reward-bench`.

Table 2 Comparison of different post-trained Qwen3-1.7B (Instruct) models using RLLM or RLVR on easy-to-verify and hard-to-verify reasoning benchmarks. All models are trained on hard-to-verify examples in a reference-based setting. RLVR models are optimized using either rule-based or model-based verifiers. RLLM models are optimized using either prompted or trained LM-as-RM (functioning as reference-based verifiers). All RLLM variants outperform all RLVR variants.

Method	Verifier	Verifier Type	Verifier Size	Math (Easy-to-Verify)						Hard-to-Verify
				AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**	**Physics**
`Qwen3-1.7B`	–	–	–	45.22	36.20	49.78	23.30	21.44	35.20	15.57
Policy Trained on hard-to-verify samples w/ Reference Answer										
RLVR	Math-Verify	Rule-based	–	48.96	34.79	47.51	21.24	25.83	35.67	15.95
RLVR	`TIGER-Lab/general-verifier`	Generative	1.5B	50.42	41.66	49.38	23.55	23.33	37.67	17.39
RLLM	`Qwen3-1.7B`	Generative	1.7B	52.50	41.46	49.59	**33.30**	26.66	40.70	16.93
RLLM	`Qwen3-32B`	Generative	32B	**57.29**	43.33	52.93	23.30	28.97	41.16	
RLLM	`J1-Qwen3-32B-Verifier`	Generative	32B	55.83	**46.05**	**53.32**	30.00	**31.03**	**43.24**	19.36
Δ w/ Qwen3-1.7B	–	–	–	+10.61	+9.85	+3.54	+6.70	+9.59	+8.04	

Table 3 Comparison of RLLM, RLHF, and RLVR across different training datasets – easy-to-verify, hard-to-verify, reference-free, and reference-based. RLLM on hard-to-verify data with a strong LM-as-RM outperforms all models trained on easy-to-verify data.

Method	RM/Verifier	RM Type	RM Size	Math					
				AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**
`Qwen3-1.7B`	–	–	–	45.22	36.20	49.78	23.30	21.44	35.20
Policy Trained on easy-to-verify samples w/o Reference Answer									
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	48.96	35.21	43.96	33.30	25.20	37.33
RLLM	`J1-Qwen-32B-RM`	Generative	32B	53.34	45.43	52.08	26.70	27.33	41.18
Policy Trained on easy-to-verify samples w/ Reference Answer									
RLVR	Math-Verify	Rule-based	–	53.75	40.84	50.42	26.70	26.44	39.63
RLLM	`J1-Qwen-32B-Verifier`	Generative	32B	54.99	44.36	52.29	25.83	26.88	40.87
Policy Trained on hard-to-verify samples w/o Reference Answer									
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	48.33	36.26	47.91	22.92	24.36	35.96
RLLM	`J1-Qwen3-32B-RM`	Generative	32B	**57.91**	44.17	**54.16**	**33.30**	27.50	**43.41**
Policy Trained on hard-to-verify samples w/ Reference Answer									
RLVR	Math-Verify	Rule-based	–	48.96	34.79	47.51	21.24	25.83	35.67
RLLM	`J1-Qwen-32B-Verifier`	Generative	32B	55.83	**46.05**	53.32	30.00	**31.03**	43.24

best-performing RM on offline benchmarks (e.g., RewardBench2), such performance does not translate to best downstream performance when doing online RL. This echoes the findings of past work that also shows that highest scoring RMs on static benchmarks does not ensure a good post-trained model (Malik et al., 2025).

- In a self-rewarding RLLM setting where both the policy and the RM are Qwen3-1.7B models (5th row), we observe improvements over the base policy (35.20 $\rightarrow$ 38.84). This result demonstrates that a thinking LM-as-RM can provide sufficiently high-quality rewards to drive self-improvement, at least to some extent.

- Furthermore, RLLM scales with the size and the capability of the LM-as-RM, with our on-policy-trained J1-Qwen3-32B-RM obtaining the best results and even outperforming a larger GPT-OSS-120B as the RM. In particular, our best RLLM model obtains an average of 8% absolute improvement on competition math over the Qwen3-1.7B Instruct model. This is noteworthy given that Qwen3 models are already heavily post-trained for these benchmarks.

- RLLM-trained models also generalize to the out-of-domain and hard-to-verify Physics benchmark, while only training on math prompts.

In summary, our results show that RLLM with strong LLMs-as-RMs can outperform RLHF. Moreover, on-policy training of an LM-as-RM can lead to further improvements. Using an LM for both the policy and as an RM also provides a natural framework for building self-rewarding and self-improving LLMs (Yuan et al., 2024).

Table 4 Comparison of Win Rate (WR) and Length Controlled Win Rate (LCWR) of RLLM and RLHF on non-verifiable instruction-following tasks when training a Qwen3-1.7B policy (either in thinking or non-thinking mode). For AlpacaEval 2.0, we use GPT-4o as the evaluator and for ArenaHard 2.0, we use GPT-4.1 as the evaluator. RLLM matches or outperforms RLHF, obtaining best win rates on hard prompts of ArenaHard 2.0.

Method	RM	RM Type	RM Size	Datasets		
				AlpacaEval 2.0 (LCWR / WR)	**ArenaHard 2.0** (Hard Prompts)	**ArenaHard 2.0** (Creative Writing)
Policy Trained on non-verifiable WildChat samples in thinking mode						
`Qwen3-1.7B`	–	–	–	37.5 / 39.2	6.4 (-0.6 / +0.7)	6.9 (-1.0 / +1.1)
RLHF	`Nexusflow/Athene-RM-8B`	Scalar	8B	38.0 / 42.6	4.3 (-0.6 / +0.7)	**15.8 (-2.0 / +1.9)**
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	43.0 / 47.2	6.0 (-0.6 / +0.6)	8.4 (-0.9 / +1.0)
RLLM	`Qwen3-1.7B`	Generative	1.7B	**43.9 / 49.2**	**8.5 (-0.8 / +0.8)**	14.7 (-1.4 / +1.8)
Policy Trained on non-verifiable WildChat samples in non-thinking mode						
`Qwen3-1.7B`	–	–	–	27.8 / 30.0	3.1 (-0.6 / +0.4)	3.1 (-0.9 / +0.7)
RLHF	`Nexusflow/Athene-RM-8B`	Scalar	8B	37.7 / 38.1	3.9 (-0.6 / +0.7)	8.0 (-1.4 / +1.3)
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	34.9 / 40.0	3.6 (-0.8 / +0.6)	7.8 (-1.2 / +1.2)
RLLM	`Qwen3-1.7B`	Generative	1.7B	**41.4 / 43.7**	**6.0 (-0.8 / +0.7)**	**9.1 (-1.2 / +1.2)**

RLLM also outperforms RLVR (with rule-based and model-based verifiers). Table 2 compares our RLLM-trained models with RLVR-trained models. In this setting, we train on the same dataset as above but assume access to reference answers. We list our main takeaways below:

- When reference answers do exist, under the same RLLM framework, we can also employ strong thinking-LLMs as *reference-based verifiers*. RLLM with such thinking-verifiers can thus outperform RLVR with rule-based verifiers like `math-verify` (35.67 → 43.24), which is known to fail in cases where answers are complex mathematical objects (Tao et al., 2025).
- Similar to the reference-free setting described above, training an on-policy J1-Qwen3-32B-Verifier model improves math results by 2% over prompted Qwen3-32B (41.16 → 43.24), thus highlighting the effectiveness of RL-trained LMs-as-RMs over prompted ones. Our RLLM model also generalizes better to the hard-to-verify physics benchmark.

RLLM on hard-to-verify training data outperforms RLVR on easy-to-verify training data. So far, we have shown that when the training data is hard-to-verify, RLLM can outperform RLHF and RLVR. A natural question then arises that if we already have access to easy-to-verify data (in which case, a rule-based verifier like `math-verify` might also suffice), do we even need to train on hard-to-verify data? We answer this in Table 3 by training RLLM on all different subsets of training data – easy-to-verify, hard-to-verify, reference-free, and reference-based. Our conclusions are as follows:

- Reference-free RLLM training on hard-to-verify data outperforms reference-based RLVR training on easy-to-verify data (39.63 → 43.41). This suggests that one way to scale RL for post-training (beyond easily verifiable domains) is to train strong LMs-as-RMs.
- RLVR training on easy-to-verify data works better than on hard-to-verify data because of more accurate rewards (35.67 → 39.63). However, even when training on easy-to-verify data, RLLM can match or outperform RLHF and RLVR.
- RLLM training on hard-to-verify data leads to better downstream performance than training on easy-to-verify data (41.18 → 43.41).

Overall, this helps establish RLLM as a unified and superior recipe across all post-training regimes. Our results also indicate that the following RLLM setting leads to the best downstream performance: (1) hard training prompts with the (2) strongest on-policy trained LM-as-RM, (3) capable of rewarding rollouts either in reference-free or reference-based setting.

Given the hybrid nature of Qwen3 models, we also evaluate our post-trained models in *non-thinking* mode. Results in Appendix Table 9 demonstrate that RLLM outperforms RLHF and RLVR by large margins. We draw similar conclusions in Appendix Table 10 when experimenting with a Llama-based OctoThinker-8B-Hybrid-Base model and in Appendix Table 8 with a larger Qwen3-8B seed model.

Beyond verifiable domains, RLLM is also performant on non-verifiable instruction-following tasks. Next, in Table 4, we evaluate the effectiveness of RLLM for non-verifiable tasks. Unlike RLLM training for verifiable tasks which used a *pointwise* LM-as-RM, here we use a *pairwise* LM-as-RM. This is because non-verifiable tasks do not have a strict notion of correctness and past works have shown that such responses are also easier to evaluate in a comparative setting. In particular, we consider a self-rewarding RLLM setting with Qwen3-1.7B both as the policy and the LM-as-RM, construct pairs of rollouts, and prompt the Qwen3-1.7B LM-as-RM to assign scores between 0-10 to each of the two rollouts. These scores are then averaged across all pairs to get pointwise reward estimates. Later in the ablations, we also compare pointwise, pairwise, and k-wise reward assignment for these tasks. We observe that RLLM can match or outperform RLHF, while obtaining better win rates for harder prompts. Appendix Table 11 reports similar conclusions when training a Qwen3-8B policy model. This result again reinforces the effectiveness of RLLM as a general framework for post-training on all kinds of tasks. Like verifiable tasks, we expect further improvements on non-verifiable tasks upon on-policy training of the LM-as-RM, which we leave for future work.

5.2 Analyses, and Ablations

Generator-Verifier Gap. In this section, we investigate the impact of the *generator–verifier gap* on RLLM training, specifically examining how the capability gap between the policy LM and the LM-as-RM influences downstream policy improvements. Recall that for our main experiments, we trained a Qwen3-1.7B policy with a J1-Qwen3-32B-RM where the RM was trained on-policy (by sampling responses from the Qwen3-1.7B policy). Now we ask if we train a weaker 1.7B LM-as-RM on its own responses i.e., J1-Qwen-1.7B-RM, can that also lead to downstream improvements? As shown in Table 5, we do not observe further improvements on top of the prompted Qwen3-1.7B-as-RM with J1 training. This result is further evidenced by Figure 4, where we compare the raw accuracy of different LMs-as-RMs on an in-distribution validation set. We observe that J1 training of a Qwen3-32B model leads to 10% improvement in judgment accuracy (averaged across 8 seeds) while providing almost no improvement on top of Qwen3-1.7B. In summary, training a Qwen3-1.7B model to evaluate its own responses leads to limited success and consequently, the resultant RM also does not lead to any downstream policy improvements. This underscores the importance of the capability gap between the generator and the verifier for obtaining downstream improvements. In Appendix Figure 14, we show examples of correct and incorrect thinking traces generated by J1-Qwen3-1.7B-RM and J1-Qwen3-32B-RM respectively.

Off-policy vs On-policy trained LM-as-RM. In Table 6, we compare an on-policy trained LM-as-RM with two off-policy trained RMs. All three RMs are trained on top of the same Qwen3-32B model using the same J1 recipe, differing only in their training data: the off-policy RMs are trained on responses generated either by a weaker Llama model or by a stronger Qwen3-8B model. Athough Figure 5 shows that J1 training improves judgment accuracy for all these models on their respective in-distribution validation sets, the off-policy trained LMs-as-RMs do not transfer to downstream policy improvements.[3] This again shows that RM capability improvements measured on static, offline benchmarks (with different data distributions) may not always be indicative of downstream task improvements because of lack of OOD generalization.

Reference-free vs Reference-based LM-as-RM. LMs-as-RMs have the flexibility of being trained and used in both a reference-free and a reference-based manner. In Figure 6, we compare LMs-as-RMs with and without reference answers, indicated by the suffixes '-Verifier' and '-RM' respectively. Unsurprisingly, presence of reference answers allows for more accurate judgments with Qwen3-32B-Verifier outperforming Qwen3-32B-RM by 16%. The performance improves further to 92% after J1 training, showing that our recipe for training LMs-as-RMs is performant in both reference-free and reference-based settings.

Pointwise, Pairwise, vs Listwise LM-as-RM Rewards. Recall that for non-verifiable tasks, we employed a pairwise LM-as-RM, primarily because non-verifiable tasks benefit from relative judgments. In Table 7, we compare the effect of scaling up reward modeling compute by conducting either pointwise, pairwise, or listwise scoring from the LM-as-RM. Since the complexity of pairwise scoring is quadratic in the number of rollouts, we also explore a second pairwise setting where one of the rollouts is chosen at random as a pivot (or reference) rollout to compare against. We observe that on the hard prompts, win rates improve with more judgments while for

[3]The reward curves also reflect the respective hardness of the RM data wherein responses from the weaker Llama model are easiest to judge, followed by Qwen3-1.7B, and Qwen3-8B.

Table 5 Analysis of Generator-Verifier Gap. RLLM post-training of a Qwen3-1.7B policy with a J1-Qwen3-1.7B LM-as-RM does not improve performance over the prompted LM-as-RM baseline while post-training with a stronger J1-Qwen3-32B LM-as-RM improves over the corresponding prompted baseline.

Method	RM/Verifier	RM Type	RM Size	Math (Easy-to-Verify)						Hard-to-Verify
				AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**	**Physics**
`Qwen3-1.7B`	–	–	–	45.22	36.20	49.78	23.30	21.44	35.20	15.57
Policy Trained on hard-to-verify samples w/o Reference Answer										
RLLM	`Qwen3-1.7B`	Generative	1.7B	51.25	37.71	51.67	26.70	26.88	**38.84**	**17.84**
RLLM	`J1-Qwen3-1.7B`	Generative	1.7B	52.29	39.59	45.63	30.00	21.46	37.79	15.11
Policy Trained on hard-to-verify samples w/o Reference Answer										
RLLM	`Qwen3-32B`	Generative	32B	54.38	43.53	53.76	26.70	31.02	41.88	16.48
RLLM	`J1-Qwen3-32B-RM`	Generative	32B	57.91	44.17	54.16	33.30	27.50	**43.41**	**18.75**

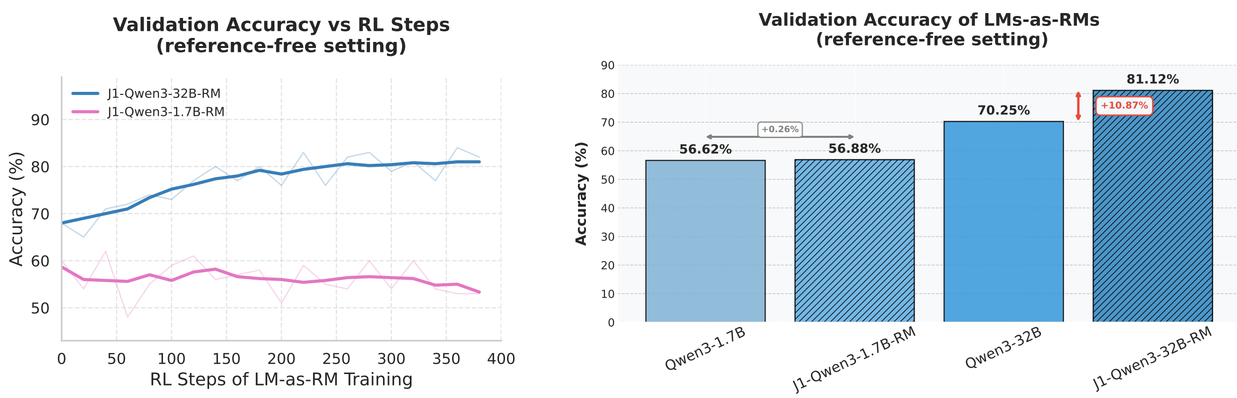


Figure 4 Analysis of Generator-Verifier Gap. (a) Comparison of different LMs-as-RMs in a reference-free setting on a held-out validation set (of correct/incorrect responses). `J1` training on top of a weaker Qwen3-1.7B does not lead to further improvements, while the same on top of a stronger Qwen3-32B leads to 10% absolute improvement. Results are averaged across 8 seeds. (b) Corresponding validation reward curves for `J1` training across RL steps.

the other categories, results mostly saturate at pairwise comparisons. Overall, this highlights the flexibility of an LM-as-RM's rewarding mechanism, allowing increased compute to be spent on evaluation.

6 Related Work

Reward Models. In recent years, several reward modeling benchmarks have been introduced to track progress in RM development (Lambert et al., 2025; Malik et al., 2025; Frick et al., 2025; Tan et al., 2025; Liu et al., 2024a). The primary goal of these benchmarks is to provide proxy evaluations for downstream tasks to enable faster iterations of model development. Alongside these benchmarks, a growing body of work has explored diverse training paradigms for both scalar and generative reward models. Different from scalar RMs that

Table 6 Comparison of RLLM post-training of Qwen3-1.7B with on-policy versus off-policy `J1`-trained LMs-as-RMs. On-policy J1-Qwen3-32B-RM is trained on Qwen3-1.7B responses while off-policy models are trained on either weaker Llama responses or stronger Qwen3-8B responses. On-policy trained LM-as-RM outperforms off-policy trained ones.

Method	RM/Verifier	RM Training Data	Math					
			AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**
`Qwen3-1.7B`	–	–	45.22	36.20	49.78	23.30	21.44	35.20
Policy Trained on hard-to-verify samples w/o Reference Answer								
RLLM	`Qwen3-32B (prompted)`	–	54.38	43.53	53.76	26.70	**31.02**	41.88
RLLM	`J1-Qwen3-32B-RM (off-policy-trained)`	Llama-3.1-8B-Instruct	57.09	44.37	48.96	26.87	28.12	41.08
RLLM	`J1-Qwen3-32B-RM (off-policy-trained)`	Qwen3-8B	54.99	**45.82**	50.00	23.55	27.71	40.41
RLLM	`J1-Qwen3-32B-RM (on-policy-trained)`	Qwen3-1.7B	**57.91**	44.17	**54.16**	**33.30**	27.50	**43.41**

Table 7 Effect of scaling up reward modeling compute in RLLM via pointwise, pairwise, pairwise with a pivot rollout, and triplet-based scoring between rollouts.

Method	RM	RM Type	#RM Judgments	Datasets: AlpacaEval 2.0 (LCWR / WR)	ArenaHard 2.0 (Hard Prompts)	ArenaHard 2.0 (Creative Writing)
Qwen3-1.7B	–	–	–	37.5 / 39.2	6.4 (-0.6 / +0.7)	6.9 (-1.0 / +1.1)
Policy Trained on WildChat non-verifiable samples in thinking mode						
RLLM	Qwen3-1.7B (prompted)	Pointwise	n	40.1 / 46.2	5.3 (-0.7 / +0.8)	12.4 (-1.6 / +1.4)
RLLM	Qwen3-1.7B (prompted)	Pairwise with pivot	$2n$	41.0 / 42.8	6.3 (-0.8 / +0.8)	13.9 (-1.3 / +1.2)
RLLM	Qwen3-1.7B (prompted)	Pairwise	$\binom{n}{2}$	**43.9 / 49.2**	8.5 (-0.8 / +0.8)	**14.7 (-1.4 / +1.8)**
RLLM	Qwen3-1.7B (prompted)	Triplet	$\binom{n}{3}$	42.0 / 48.1	**10.1 (-0.9 / +1.0)**	13.1 (-1.4 / +1.8)

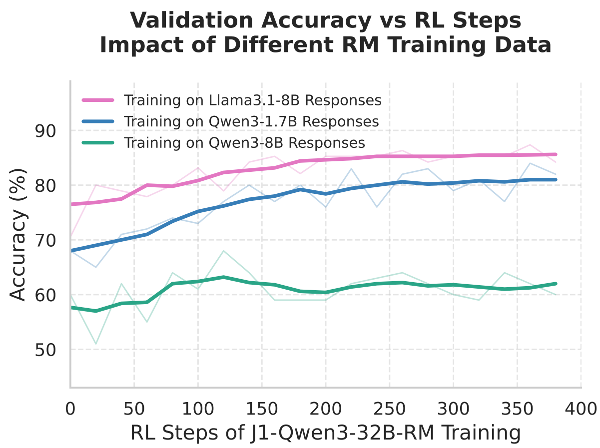


Figure 5 Comparison of validation reward curves for J1 training on different distributions of data. Data is either sampled from a Llama3 model (off-policy weakest), Qwen3-8B model (off-policy strongest), or Qwen3-1.7B model (on-policy). While all LMs-as-RMs improve on their respective in-distribution validation data, off-policy trained RMs do not lead to downstream task improvements potentially because of lack of OOD generalization (Table 6).

only output scores (Ouyang et al., 2022), generative RMs also output Chain-of-Thoughts alongside the scores (where the scores may be generated as tokens or from a separate reward head) (Mahan et al., 2024; Ankner et al., 2024; Zhang et al., 2025; Whitehouse et al., 2025b). Different methods have been proposed to train these generative RMs, including SFT (Wang et al., 2024), DPO (Saha et al., 2025; Mahan et al., 2024), and more recently, online RL (Whitehouse et al., 2025b; Liu et al., 2025b; Guo et al., 2025b; Chen et al., 2025).

Policy Learning with Reward Models. Evaluations in most prior studies remain confined to RM benchmarks, without using the RM to post-train a model with RL. In cases when such post-training is indeed performed, we also observe a lack of clear baselines (e.g., comparing a post-trained model to just the base model and not other RMs), which makes it harder to understand the effectiveness of an RM. This evaluation strategy is suboptimal: RMs are susceptible to reward hacking because of lack of generalization to OOD rollouts. Moreover, improvements in offline judgment accuracy do not necessarily translate into proportional downstream policy gains. For instance, we find that a 10% absolute increase in judgment accuracy under J1 training (Figure 4) yields 2% improvement in policy performance (Table 1). One contributing factor is policy diversity: even a highly accurate reward model may provide no learning signal if the policy predominantly samples rollouts that are uniformly correct or uniformly incorrect. Moreover, when assessing reward model progress, absolute performance matters in addition to relative improvement – for example, an RM that improves but remains near random accuracy may still be insufficient to effectively train the policy. We therefore recommend that offline RM evaluations, while informative, be complemented with downstream policy training results.

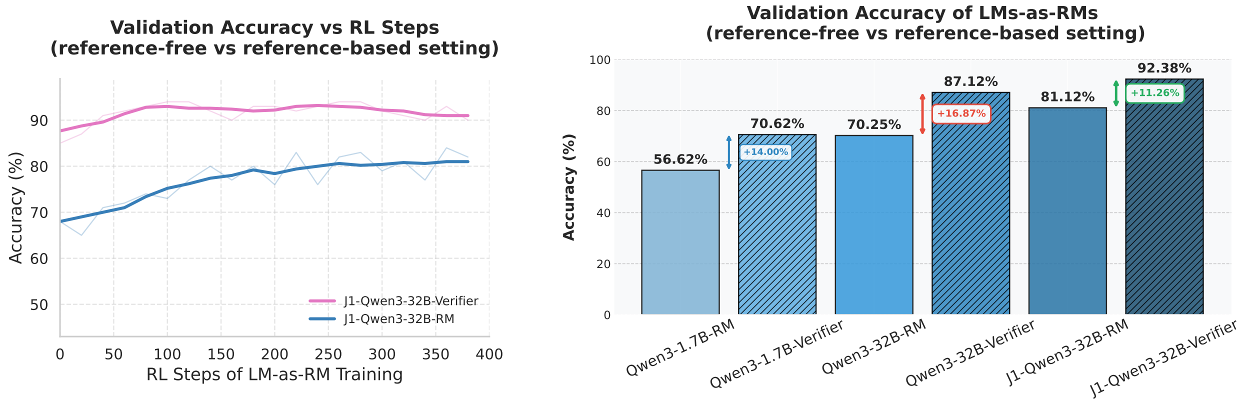


Figure 6 (a) Comparison of different LMs-as-RMs evaluated in either a reference-free setting (suffixed with '-RM') or in a reference-based setting (suffixed with '-Verifier'. Reference-based LMs-as-RMs are more accurate than reference-free ones (e.g., J1-Qwen3-32B-Verifier is 10% better than J1-Qwen3-32B-RM). (b) Validation reward curves for a reference-free J1-Qwen3-32B-RM and a reference-based J1-Qwen3-32B-Verifier.

7 Conclusion

We showed that RLLM – RL with (RL-trained) language models as reward models – can serve as a single, unified post-training recipe across easy-to-verify, hard-to-verify, and non-verifiable tasks. Through extensive experiments, we demonstrated that RLLM outperforms both RLHF (with scalar RMs) and RLVR (with rule-based rewards), showcasing particularly large gains when training on hard-to-verify tasks. We also studied the importance of on-policy training of LM-as-RM models alongside the impact of generator-verifier gap and showed that these are important components for successful RLLM training.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. URL https://arxiv.org/abs/1606.06565.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan Daniel Chang, and Prithviraj Ammanabrolu. Critique-out-Loud Reward Models. In *Pluralistic Alignment Workshop at NeurIPS 2024*, 2024. URL https://openreview.net/forum?id=CljYUvIlRW.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a Helpful and Harmless Assistant with Reinforcement Learning From Human Feedback. *arXiv preprint arXiv:2204.05862*, 2022a. URL https://arxiv.org/abs/2204.05862.

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*, 2022b. URL https://arxiv.org/abs/2212.08073.

Can Balioglu, Alexander Erben, Martin Gleize, Artyom Kozhevnikov, Ilia Kulikov, and Julien Yao. fairseq2, 2023. URL http://github.com/facebookresearch/fairseq2.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating LLMs on Uncontaminated Math Competitions. *arXiv preprint arXiv:2505.23281*, 2025. URL https://arxiv.org/abs/2505.23281.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. RM-R1: Reward Modeling as Reasoning. *arXiv preprint arXiv:2505.02387*, 2025. URL https://arxiv.org/abs/2505.02387.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alexander Nicholas D'Amour, Krishnamurthy Dj Dvijotham, Adam Fisch, Katherine A Heller, Stephen Robert Pfohl, Deepak Ramachandran, et al. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2312.09244.

Evan Frick, Peter Jin, Tianle Li, Karthik Ganesan, Jian Zhang, Jiantao Jiao, and Banghua Zhu. Athene-70B: Redefining the Boundaries of Post-Training for Open Models, July 2024. URL https://nexusflow.ai/blogs/athene.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to Evaluate Reward Models for RLHF. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cbttLtO94Q.

Leo Gao, John Schulman, and Jacob Hilton. Scaling Laws for Reward Model Overoptimization. In *International Conference on Machine Learning*, pages 10835–10866, 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

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*, 2025a. URL https://arxiv.org/abs/2501.12948.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Reward Reasoning Model. *arXiv preprint arXiv:2505.14674*, 2025b. URL https://arxiv.org/abs/2505.14674.

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. URL https://arxiv.org/abs/2410.21276.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 System Card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. URL https://dl.acm.org/doi/abs/10.1145/3600006.3613165.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing Frontiers in Open Language Model Post-training. *arXiv preprint arXiv:2411.15124*, 2024. URL https://arxiv.org/abs/2411.15124.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797, 2025. URL https://aclanthology.org/2025.findings-naacl.96/.

Jack Lanchantin, Angelica Chen, Janice Lan, Xian Li, Swarnadeep Saha, Tianlu Wang, Jing Xu, Ping Yu, Weizhe Yuan, Jason E Weston, et al. Bridging Offline and Online Reinforcement Learning for LLMs. *arXiv preprint arXiv:2506.21495*, 2025. URL https://arxiv.org/abs/2506.21495.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *International Conference on Machine Learning*, pages 26874–26901. PMLR, 2024. URL https://arxiv.org/abs/2309.00267.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From Crowdsourced Data to High-quality Benchmarks: Arena-Hard and Benchbuilder Pipeline. In *Forty-second International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2406.11939.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An Automatic Evaluator of Instruction-following Models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, et al. Skywork-Reward-V2: Scaling Preference Data Curation via Human-AI Synergy. *arXiv preprint arXiv:2507.01352*, 2025a. URL https://arxiv.org/abs/2507.01352.

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, 2024a. URL https://arxiv.org/abs/2410.16184.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceMath: Advancing Frontier Math Reasoning with Post-Training and Reward Modeling. *arXiv preprint*, 2024b. URL https://arxiv.org/abs/2412.15084.

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*, 2025b. URL https://arxiv.org/abs/2504.02495.

Xun Lu. Writing-Zero: Bridge the Gap Between Non-verifiable Problems and Verifiable Rewards. *arXiv preprint arXiv:2506.00103*, 2025. URL https://arxiv.org/abs/2506.00103.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun MA, and Wenhu Chen. General-reasoner: Advancing LLM reasoning across all domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pBFVoll8Xa.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative Reward Models. *arXiv preprint arXiv:2410.12832*, 2024. URL https://arxiv.org/abs/2410.12832.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing Reward Model Evaluation. *arXiv preprint arXiv:2506.01937*, 2025. URL https://arxiv.org/abs/2506.01937.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 Winning Solution: Building State-of-the-Art Mathematical Reasoning Models with OpenMathReasoning Dataset. *arXiv preprint arXiv:2504.16891*, 2025. URL https://arxiv.org/abs/2504.16891.

OpenAI. OpenAI o3 and o4-mini System Card, 2025. URL https://openai.com/index/o3-o4-mini-system-card.

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. URL https://papers.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason E Weston, and Tianlu Wang. Learning to Plan & Reason for Evaluation with Thinking-LLM-as-a-Judge. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=PNRznmmWP7.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. DeepSeekMath-V2: Towards Self-Verifiable Mathematical Reasoning. *arXiv preprint arXiv:2511.22570*, 2025. URL https://arxiv.org/abs/2511.22570.

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. URL https://arxiv.org/pdf/2409.19256.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. JudgeBench: A Benchmark for Evaluating LLM-Based Judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Leitian Tao, Ilia Kulikov, Swarnadeep Saha, Tianlu Wang, Jing Xu, Sharon Li, Jason E Weston, and Ping Yu. Hybrid Reinforcement: When Reward Is Sparse, It's Better to Be Dense. *arXiv preprint arXiv:2510.07242*, 2025. URL https://arxiv.org/abs/2510.07242.

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. URL https://arxiv.org/abs/2408.02666.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training Incentivizes Reinforcement Learning Scaling. *arXiv preprint arXiv:2506.20512*, 2025. URL https://arxiv.org/abs/2506.20512.

Chenxi Whitehouse, Sebastian Ruder, Tony Lin, Oksana Kurylo, Haruka Takagi, Janice Lam, Nicolò Busetto, Denise Diaz, and Francisco Guzmán. MENLO: From Preferences to Proficiency–Evaluating and Modeling Native-like Quality Across 47 Languages. *arXiv preprint arXiv:2509.26601*, 2025a. URL https://arxiv.org/abs/2509.26601.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing Thinking in LLM-as-a-Judge via Reinforcement Learning. *arXiv preprint arXiv:2505.10320*, 2025b. URL https://arxiv.org/abs/2505.10320.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2401.10020.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative Verifiers: Reward Modeling as Next-Token Prediction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ccwp4tFEtE.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. WildChat: 1M ChatGPT Interaction Logs in the Wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Prompt Template for Pointwise Reference-free LMs-as-RMs

```
You are given a user question and a response from an AI assistant. Your task is to act as an impartial judge and
evaluate how well the response fulfills the user's instructions. You will be shown multiple responses to the
same prompt, but only one at a time. Evaluate each response independently.

Think carefully about how to assess the quality of the response and assign the assistant's response a score 1 if
the response is correct, and 0 if not. Enclose the score within <score> and </score> tags.

Format your output like this:
<think> your_thinking_process </think>
<score> 0 or 1 </score>

Below are the user's question and the assistant's response:

[User Question]
{instruction}

[The Start of the Assistant's Answer]
{response}
[The End of the Assistant's Answer]
```

Figure 7 Prompt template for pointwise reference-free LMs-as-RMs

A Prompt Templates

Figure 7, Figure 8, and Figure 9 show the prompt templates for training pointwise reference-free, pairwise, and pointwise reference-based LMs-as-RMs, respectively.

Prompt Template for Pairwise LMs-as-RMs for Non-verifiable Tasks.

```
You are given a user question and two responses from two AI assistants. Your task is to act as an impartial judge
and evaluate which response better follows the user's instructions and provides a higher-quality answer. Avoid
any position biases and ensure that the order in which the responses were presented does not influence your
decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of
the assistants. Be as objective as possible.

Think carefully about how to assess the quality of the responses and assign each response a score from 0 to 10,
using either an integer or a decimal with up to 0.1 precision, with a higher score indicating a higher-quality
response that better satisfies the criteria.  Enclose the scores within the tags <score_A> </score_A>, and
<score_B> </score_B>.

Format your output like this:
<think> your_thinking_process </think>
<score_A> your_score_a </score_A> <score_B> your_score_b </score_B>

Below are the user's question and the two responses:

[User Question]
{instruction}

[The Start of Assistant A's Answer]
{response A}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{response B}
[The End of Assistant B's Answer]
```

Figure 8 Prompt template for pairwise LMs-as-RMs for non-verifiable tasks. Note that unlike verifiable tasks, here we ask the model to assign a score between 0 to 10 to capture the fine-grained quality.

Table 8

Method	RM/Verifier	RM Type	RM Size	Math (Easy-to-Verify)						Hard-to-Verify
				AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**	**Physics**
Policy Trained on hard-to-verify samples w/o Reference Answer										
`Qwen3-8B`	–	–	–	76.00	67.30	68.75	35.41	41.25	57.74	51.70
RLHF	`Nexusflow/Athene-RM-8B`	Scalar	8B	73.34	60.42	65.00	37.91	38.33	55.00	
RLVR	`nvidia/AceMath-7B-RM`	Scalar	7B	72.91	65.00	65.41	33.75	42.49	55.91	
RLLM	`Qwen3-8B`	Generative	8B	75.84	70.00	69.58	50.00	43.33	61.75	54.32
RLLM	`J1-Qwen3-32B`	Generative	32B	73.32	70.43	73.74	43.30	50.82	62.32	

Table 9 Comparison of different post-trained Qwen3-1.7B (Instruct) models using RLLM, RLVR, or RLHF. All models are evaluated in *non-thinking* mode. Similar to our conclusions with results in thinking mode, RLLM outperforms RLHF and RLVR.

Method	RM/Verifier	RM Type	RM Size	Math (easy-to-verify)						Hard-to-Verify
				AIME24	**AIME25**	**BRUMO25**	**HMMT24**	**HMMT25**	**Average**	**Physics**
`Qwen3-1.7B`	–	–	–	13.33	10.83	16.04	6.25	5.21	10.33	8.75
Policy Trained on hard-to-verify samples w/o Reference Answer										
RLHF	`Nexusflow/Athene-RM-8B`	Scalar	8B	5.84	6.25	12.08	2.28	1.65	5.62	7.95
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	21.87	20.21	21.87	12.92	10.00	17.37	10.00
RLHF	`nvidia/AceMath-7B-RM`	Scalar	7B	20.21	16.87	18.96	7.72	5.84	13.92	10.45
RLLM	`J1-Qwen3-32B-RM`	Generative	32B	**39.58**	**35.21**	**43.97**	**20.21**	**17.51**	**31.30**	**16.02**
Δ w/ Qwen3-1.7B	–	–	–	+26.25	+24.38	+27.93	+13.96	+12.30	+20.97	+7.27
Policy Trained on hard-to-verify samples w/ Reference Answer										
RLVR	Math-Verify	Rule-based	–	12.51	12.71	20.00	8.12	7.93	12.25	8.75
RLVR	`TIGER-Lab/general-verifier`	Generative	1.5B	17.71	15.41	18.97	7.92	7.92	13.58	11.93
RLLM	`J1-Qwen3-32B-Verifier`	Generative	32B	**35.41**	**30.01**	**41.04**	**20.00**	**15.00**	**28.29**	

Table 10 Comparison of different post-trained OctoThinker-8B-Hybrid-Base models using RLLM, RLVR, or RLHF. This shows the generalization of RLLM to Llama-based models.

Method	RM/Verifier	RM Type	RM Size	**AIME24**	**AIME25**	**HMMT24**	**HMMT25**	**BRUMO25**	**MATH500**	**Average**
`OctoThinker-8B-Hybrid-Base`	–	–	–	2.49	2.70	4.30	0.00	0.00	52.36	10.32
Policy Trained on hard-to-verify samples w/o Reference Answer										
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	7.30	3.74	6.88	3.30	0.00	60.04	13.54
RLLM	`Qwen3-1.7B (prompted)`	Generative	1.7B	**7.93**	**4.37**	**9.79**	**6.70**	**0.83**	**60.86**	**15.08**
Δ w/ OctoThinker-8B-Hybrid-Base	–	–	–	+5.44	+1.67	+5.49	+6.70	+0.83	+8.50	+4.76
Policy Trained on hard-to-verify samples w/ Reference Answer										
RLVR	Math-Verify	Rule-based	–	3.31	1.45	6.67	3.30	**0.41**	55.64	11.80
RLVR	`TIGER-Lab/general-verifier`	Generative	1.5B	2.29	1.24	5.00	3.30	0.00	47.67	9.92
RLLM	`Qwen3-1.7B (prompted)`	Generative	1.7B	**10.84**	**7.08**	**7.52**	**3.30**	0.00	**61.75**	**15.08**
Δ w/ OctoThinker-8B-Hybrid-Base	–	–	–	+8.35	+4.38	+3.22	+3.30	+0.00	+9.39	+4.76

Table 11 Comparison of Win Rate (WR) and Length Controlled Win Rate (LCWR) of RLLM and RLHF on non-verifiable instruction-following tasks when training a Qwen3-8B policy. For AlpacaEval 2.0, we use GPT-4o as the evaluator and for ArenaHard 2.0, we use GPT-4.1 as the evaluator. RLLM matches or outperforms RLHF, obtaining best win rates on hard prompts of ArenaHard 2.0.

Method	RM	RM Type	RM Size	Datasets		
				AlpacaEval 2.0 (LCWR / WR)	**ArenaHard 2.0** (Hard Prompts)	**ArenaHard 2.0** (Creative Writing)
Policy Trained on non-verifiable WildChat samples in thinking mode						
`Qwen3-8B`	–	–	–	63.1 / 65.1	22.4 (-1.5 / +1.7)	33.3 (-2.5 / +2.7)
RLHF	`Nexusflow/Athene-RM-8B`	Scalar	8B	70.9 / 71.2	25.4 (-1.6 / +1.6)	57.9 (-2.8 / +2.0)
RLHF	`Skywork-Reward-V2-Llama-3.1-8B`	Scalar	8B	68.6 / 72.3	26.4 (-1.6 / +1.7)	50.9 (-2.9 / +2.9)
RLLM	`Qwen3-8B (prompted)`	Generative	8B	**71.4 / 77.1**	**32.7 (-1.6 / +1.5)**	**61.9 (-2.3 / +2.3)**

Prompt Template for Pointwise Reference-based LMs-as-RMs

```
You are given a user question, a reference answer, and a response from an AI assistant. Your task is to act as an
impartial judge and evaluate how well the response fulfills the user's instructions. You will be shown multiple
responses to the same prompt, but only one at a time. Evaluate each response independently.

Think carefully about how to assess the quality of the response and assign the assistant's response a score 1 if
the response is correct, and 0 if not. Enclose the score within <score> and </score> tags.

Format your output like this:
<think> your_thinking_process </think>
<score> 0 or 1 </score>

Below are the user's question and the assistant's response:

[User Question]
{instruction}

[Reference Answer]
{reference_answer}

[The Start of the Assistant's Answer]
{response}
[The End of the Assistant's Answer]
```

Figure 9 Prompt template for Pointwise Reference-based LMs-as-RMs, where an additional reference answer is provided compared to the reference-free version. Note that the reference answer is only provided as additional context and the LM-as-RM is still prompted to evaluate the entire response.

Pointwise LM-as-RM Training Example (Correct)

Question:
A conducting cylinder of radius a and finite length L ($0 \le z \le L$) is filled with the steady helical flow

$$\mathbf{U}(r,\theta,z) = U_0\,\hat{\mathbf{z}} + \Omega\, r\,\hat{\boldsymbol{\theta}}, \qquad 0 \le r \le a.$$

The magnetic field obeys the induction equation with magnetic diffusivity η. Assume that the end caps at $z = 0$ and $z = L$ are perfectly conducting, so that the axial dependence of any normal-mode perturbation must satisfy the boundary condition $\mathbf{B} = 0$ at the end caps. Consequently the axial wavenumber is quantised as

$$k_n = \frac{n\pi}{L}, \qquad n = 1, 2, \ldots.$$

Adopt the normal-mode ansatz

$$\mathbf{B}(r,\theta,z,t) = \mathbf{b}(r)\, e^{\mathrm{i}m\theta + \mathrm{i}k_n z + \sigma t},$$

and focus on the most unstable azimuthal mode $m = 1$. For this mode the growth rate can be written in the form

$$\sigma = \frac{U_0}{a}\Big(\mathrm{Rm}\, F\big(q, k_n a\big) - \lambda_1^2\Big),$$

where $\mathrm{Rm} = \dfrac{U_0 a}{\eta}$ is the magnetic Reynolds number, $q = \Omega a/U_0$ is the dimensionless pitch, $\lambda_1 \approx 3.8317$ is the first zero of the Bessel function J_1, and $F(q, k_n a)$ is a positive dimensionless function that satisfies $0 < F \le 1$ for all admissible q and k_n.
Derive the **necessary condition** for dynamo action (i.e. $\sigma > 0$) that must hold **for every admissible axial mode** n. Express your result as a single inequality involving only the magnetic Reynolds number Rm, the constant λ_1, and the geometric ratio a/L.

Reference Answer: $\mathrm{Rm} \;>\; \lambda_1^2$

Model Response:
To determine the necessary condition for dynamo action, we begin by analyzing the given expression for the growth rate:

$$\sigma = \frac{U_0}{a}\left(\mathrm{Rm}\, F\left(q, k_n a\right) - \lambda_1^2\right).$$

For the magnetic field to grow (i.e., to exhibit dynamo action), the growth rate σ must be positive. Therefore:

$$\sigma > 0 \Rightarrow \mathrm{Rm}\, F(q, k_n a) > \lambda_1^2.$$

Thus, the necessary condition for dynamo action is: $\mathrm{Rm} > \frac{\lambda_1^2}{F(q,k_n a)}$.
. . . [analysis of the function $F(q, k_n a)$ and worst-case scenarios] . . .
To ensure that $\mathrm{Rm} > \frac{\lambda_1^2}{F(q,k_n a)}$ holds for all n, we must consider the worst-case value of $F(q, k_n a)$. Since $F(q, k_n a)$ is bounded above by 1, the maximum of $\frac{\lambda_1^2}{F(q,k_n a)}$ over all n occurs when $F(q, k_n a) = 1$.
Therefore, the necessary condition for dynamo action is:

$$\boxed{\mathrm{Rm} > \lambda_1^2}$$

GPT-OSS-120B Judgment: The response derives the same inequality as the reference answer and correctly concludes that the most restrictive necessary condition is obtained when $F = 1$. Both answers are equivalent. **CORRECT**

Figure 10 Example of a synthetically-generated correct training example for LM-as-RM training. Given a (question, reference answer), we sample a response sampled from the policy and check whether the answer matches the reference answer using GPT-OSS-120B.

Pointwise LM-as-RM Training Example (Incorrect)

Question:
A conducting cylinder of radius a and finite length L ($0 \leq z \leq L$) is filled with the steady helical flow

$$\mathbf{U}(r,\theta,z) = U_0\,\hat{\mathbf{z}} + \Omega\,r\,\hat{\boldsymbol{\theta}}, \qquad 0 \leq r \leq a.$$

The magnetic field obeys the induction equation with magnetic diffusivity η. Assume that the end caps at $z = 0$ and $z = L$ are perfectly conducting, so that the axial dependence of any normal-mode perturbation must satisfy the boundary condition $\mathbf{B} = 0$ at the end caps. Consequently the axial wavenumber is quantised as

$$k_n = \frac{n\pi}{L}, \qquad n = 1, 2, \ldots.$$

Adopt the normal-mode ansatz

$$\mathbf{B}(r,\theta,z,t) = \mathbf{b}(r)\,e^{\mathrm{i}m\theta + \mathrm{i}k_n z + \sigma t},$$

and focus on the most unstable azimuthal mode $m = 1$. For this mode the growth rate can be written in the form

$$\sigma = \frac{U_0}{a}\Big(\mathrm{Rm}\,F\big(q, k_n a\big) - \lambda_1^2\Big),$$

where $\mathrm{Rm} = \dfrac{U_0 a}{\eta}$ is the magnetic Reynolds number, $q = \Omega a/U_0$ is the dimensionless pitch, $\lambda_1 \approx 3.8317$ is the first zero of the Bessel function J_1, and $F(q, k_n a)$ is a positive dimensionless function that satisfies $0 < F \leq 1$ for all admissible q and k_n.
Derive the **necessary condition** for dynamo action (i.e. $\sigma > 0$) that must hold **for every admissible axial mode** n. Express your result as a single inequality involving only the magnetic Reynolds number Rm, the constant λ_1, and the geometric ratio a/L.

Reference Answer: $\mathrm{Rm} > \lambda_1^2$

Model Response:
To determine the necessary condition for dynamo action, we analyze the growth rate. For dynamo action, we require $\sigma > 0$:

$$\sigma = \frac{U_0}{a}\left(\mathrm{Rm}\,F(q, k_n a) - \lambda_1^2\right) > 0$$

Rewriting:

$$\mathrm{Rm}\,F(q, k_n a) > \lambda_1^2$$

So, for every n, we must have:

$$\mathrm{Rm} > \frac{\lambda_1^2}{F(q, k_n a)}$$

To ensure this is true for all n, we must consider the worst-case scenario, i.e., the minimum value of $F(q, k_n a)$, since it will make $\frac{\lambda_1^2}{F(q,k_n a)}$ as large as possible.
… [analysis of the function $F(q, k_n a)$ and geometric considerations] …
The function $F(q, k_n a)$ is positive and less than or equal to 1 for all admissible q and k_n. The problem explicitly states that we are to express the necessary condition in terms of Rm, λ_1, and the geometric ratio a/L.
The necessary condition for $\sigma > 0$ for every n is:

$$\boxed{\mathrm{Rm} > \lambda_1^2 \cdot \frac{a}{L}}$$

This is the most restrictive condition, since it ensures that the growth rate is positive for the worst-case value of $F(q, k_n a)$.

Judgment: The reference answer states $\mathrm{Rm} > \lambda_1^2$ with no dependence on the geometric ratio a/L. The response incorrectly introduces a factor of $\frac{a}{L}$ in the final answer. This additional factor changes the condition qualitatively and is not present in the reference answer. The response's condition is therefore not equivalent. **INCORRECT**

Figure 11 Example of a synthetically-generated incorrect training example for LM-as-RM training. Given a (question, reference answer), we sample a response sampled from the policy and check whether the answer matches the reference answer using GPT-OSS-120B.

Example of a hard-to-verify math problem

Question: Find all real polynomials $p(x)$ such that

$$p(x^2 + 1) = p(x)^2 + 1$$

for all $x \in \mathbb{R}$.

Reference Answer: $p_n(x) = ((\cdots((x^2 + 1)^2 + 1)^2 + \cdots)^2 + 1$ (nested n times)

Figure 12 Example of a hard-to-verify math problem for policy training where the reference answer is a mathematical expression.

Example of an easy-to-verify math problem

Question: In $\triangle ABC$ with $BC = a$, $CA = b$, and $AB = c$, given that $(b + c)(b - c) = ca$ and $\angle BAC = 75°$, find $\angle ABC$.

Reference Answer: $70°$

Figure 13 Example of an easy-to-verify math problem for policy training where the reference answer is an integer.

J1-Qwen3-1.7B vs J1-Qwen3-32B Evaluation Traces

Question:

Consider two open Heisenberg antiferromagnetic spin-1/2 chains, one of length $N_1 = 100$ and one of length $N_2 = 200$. Using the density matrix renormalization group (DMRG) with three different numbers of kept states m, the following total ground-state energies (in units of the exchange constant J) and discarded weights ε_m are obtained: | Chain length | m | Energy E_m | Discarded weight ε_m | |————|——|————–|——————————| | $N_1 = 100$ | 50 | -44.12 | 3.2×10^{-4} | | $N_1 = 100$ | 100 | -44.23 | 8.5×10^{-5} | | $N_1 = 100$ | 200 | -44.28 | 1.9×10^{-5} | | $N_2 = 200$ | 50 | -88.30 | 3.2×10^{-4} | | $N_2 = 200$ | 100 | -88.44 | 8.5×10^{-5} | | $N_2 = 200$ | 200 | -88.51 | 1.9×10^{-5} |
The DMRG algorithm is variational, i.e. each E_m is an upper bound to the exact ground-state energy for the corresponding chain length: $E_0(N) \le E_m(N)$.
In addition, the following rigorous bound on the energy error is known:

$$|E_m(N) - E_0(N)| \le K\,\varepsilon_m, \qquad K = 1.5 \times 10^4.$$

The exact ground-state energy per site of the infinite open Heisenberg chain is

$$e_\infty = -0.4431471806\,J,$$

and for any open chain of length N the exact energy satisfies the finite-size inequality

$$N\,e_\infty \;\le\; E_0(N) \;\le\; N\,e_\infty + \frac{C}{N}, \qquad C = 12.0.$$

Using **all** the information above (variational property, discarded-weight error bound, infinite-chain reference, finite-size bound, and the data for both chain lengths), give a **rigorous interval** that is guaranteed to contain the exact total ground-state energy $E_0(N_1 = 100)$ of the 100-site chain. Express your answer **only** in interval notation (e.g. $[a,\, b]$) without any additional text or equations.

Response (Truncated):

From DMRG with $m = 200$: - $E_0(100) \in [-44.28 - 0.285, -44.28 + 0.285] = [-44.565, -43.995]$ - Intersect with finite-size bound:

$$[-44.31471806, -44.19471806]$$

—

[...]

Final Interval The **exact** ground-state energy $E_0(100)$ is guaranteed to lie in the **intersection** of the DMRG error bounds and the finite-size bound. This is the **tightest and most rigorous** interval that contains $E_0(100)$.

$$\boxed{[-44.31471806, -44.19471806]}$$

Reference Answer:

[-44.31471806, -44.28]

Expected Evaluation: Incorrect

<score> 0 </score>

J1-Qwen3-1.7B-RM's Incorrect Thinking Trace (Truncated):

...the assistant's final interval is [-44.31471806, -44.19471806], which is the same as the finite-size bound. But the assistant also mentioned that the DMRG results with m=50, 100, 200 give energies -44.12, -44.23, -44.28...So the intersection of those is [-44.3147, -44.1947]. So the assistant's final interval is [-44.3147, -44.1947], which is the intersection of the DMRG error bound and the finite-size bound...Therefore, the assistant's answer is correct. So the score should be 1.

Evaluation Score: <score> 1 </score>

J1-Qwen3-32B-RM's Correct Thinking Trace (Truncated):

...the DMRG results provide upper bounds. For example, the DMRG with m=200 gives $E_m = -44.28$, which is an upper bound...Since $E_0 \le E_{200} = -44.28$, the tightest upper bound is -44.28, not -44.1947...the exact $E_0(100)$ must lie in [-44.3147, -44.28]. Because the upper bound is the minimum of the DMRG upper bounds (which is -44.28) and the finite-size upper bound (which is -44.1947)...the assistant's interval is too wide. The correct interval should be the intersection of the finite-size lower bound and the DMRG upper bounds...Therefore, the rigorous interval should be [-44.31471806, -44.28]. The assistant's answer is wrong.

Evaluation Score: <score> 0 </score>

Figure 14 Qualitative comparison of a weaker LM-as-RM J1-Qwen3-1.7B's thinking trace with a stronger J1-Qwen3-32B's thinking trace. The stronger LM-as-RM correctly identifies the response as incorrect and also specifies the correct answer (which matches the reference answer).