

# J1: Incentivizing Thinking in LLM-as-a-Judge via Reinforcement Learning

Chenxi Whitehouse<sup>†</sup>, Tianlu Wang<sup>‡</sup>, Ping Yu<sup>‡</sup>, Xian Li<sup>‡</sup>, Jason Weston<sup>‡</sup>, Ilia Kulikov<sup>‡</sup>, Swarnadeep Saha<sup>‡</sup>

<sup>†</sup>GenAI at Meta, <sup>‡</sup>FAIR at Meta

The progress of AI is bottlenecked by the quality of evaluation, and powerful LLM-as-a-Judge models have proved to be a core solution. Improved judgment ability is enabled by stronger chain-of-thought reasoning, motivating the need to find the best recipes for training such models to *think*. In this work we introduce **J1**, a reinforcement learning approach to training such models. Our method converts both verifiable and non-verifiable prompts to judgment tasks with verifiable rewards that incentivize thinking and mitigate judgment bias. In particular, our approach outperforms all other existing 8B or 70B models when trained at those sizes, including models distilled from DeepSeek-R1. J1 also outperforms o1-mini, and even R1 on some benchmarks, despite training a smaller model. We provide analysis and ablations comparing Pairwise-J1 vs Pointwise-J1 models, offline vs online training recipes, reward strategies, seed prompts, and variations in thought length and content. We find that our models make better judgments by learning to outline evaluation criteria, comparing against self-generated reference answers, and re-evaluating the correctness of model responses.

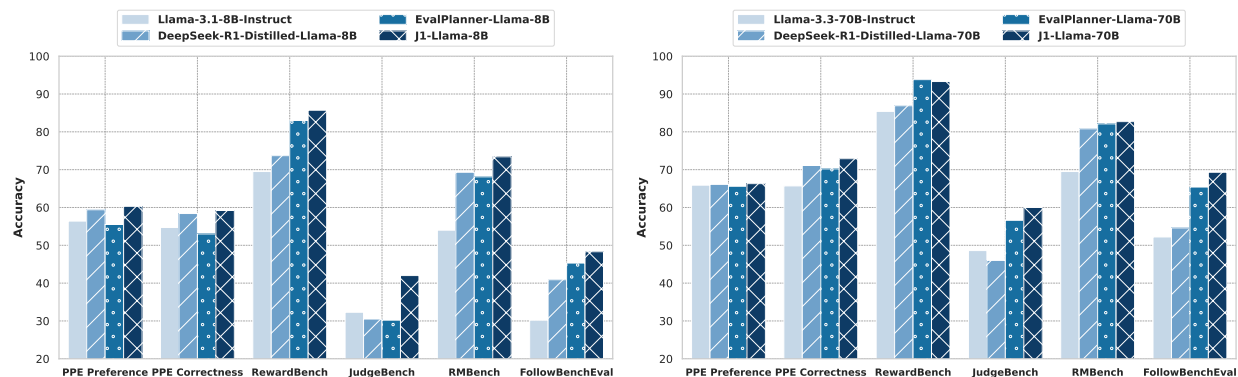
**Date:** May 16, 2025

**Correspondence:** [chenxwh@meta.com](mailto:chenxwh@meta.com), [swarnadeep@meta.com](mailto:swarnadeep@meta.com)



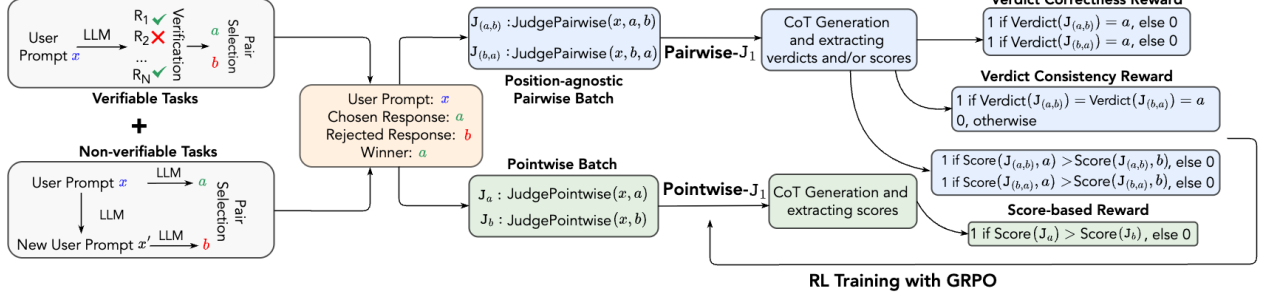
## 1 Introduction

Better judgments can be made by learning how to reason, which is observed in both humans and machines. For models, the ability to judge predictions is a vital process that is applied at all stages of development: during training and inference to provide a reward or verification signal, and during final benchmark evaluation to judge performance. Classical evaluation using reward models typically outputs a score directly (Ouyang et al., 2022) without having an explicit reasoning step. Using pre-trained and aligned language models to act as judges instead, i.e., LLM-as-a-Judge, allowed the possibility to generate chain-of-thought reasoning before making a decision, which was at first invoked by prompting (Zheng et al., 2023; Gu et al., 2024; Saha et al., 2024). Subsequently, iterative finetuning and direct preference optimization (DPO) methods were developed to improve these reasoning steps (Mahan et al., 2024; Wang et al., 2024d; Saha et al., 2025). In this work, we investigate recipes for further improvements to judgment reasoning via online Reinforcement Learning (RL).



**Figure 1** Comparison of Pairwise-J1 with state-of-the-art LLM-as-a-Judge models on five reward modeling benchmarks at 8B and 70B scale. Pairwise-J1 outperforms all baselines despite being trained only on synthetic preference pairs.





**Figure 3** Reinforcement Learning recipes for training Pairwise-J1 and Pointwise-J1 models. We generate synthetic preference pairs for both verifiable and non-verifiable tasks to create position-agnostic training batches. Rewards based on verdict correctness, consistency, and score alignment jointly optimize evaluation thoughts and verdicts using online RL (GRPO). Pointwise-J1 is trained *only* via distant supervision from pairwise labels.

thought tokens  $t$ , before deriving the final verdict. In the following subsections, we describe our training data and the Pairwise-J1 training recipe. Then in Section 2.5, we present alternative LLM-as-a-Judge settings and discuss how we derive RL recipes for them. An overview of our modeling setup is provided in Figure 3.

## 2.2 Synthetic Training Data Generation

The goal of J1 is to train a generalist judge that can evaluate LLM responses for both verifiable and non-verifiable tasks. With that motivation, we adopt the same synthetic training data used by Saha et al. (2025) to train EvalPlanner. Unlike most previous works (Liu et al., 2025b; Chen et al., 2025b), the use of synthetic data removes the reliance on human-annotated preference pairs since it can be costly to obtain (Wang et al., 2024a). It also ensures that J1 has a fair comparison with a prior state-of-the-art Thinking-LLM-as-a-Judge recipe which used DPO, allowing us to directly assess the benefits of our online RL-based training strategy.

J1’s final 22K training data consists of 17K WildChat (Zhao et al., 2024) and 5K MATH (Hendrycks et al., 2021) prompts, along with their corresponding preference pairs. For WildChat, rejected responses are obtained by prompting an LLM to first generate a “noisy” variant of the original instruction and then produce a response to this noisy instruction (Wang et al., 2024d). For MATH, rejected responses are sampled from generations by an LLM that do not lead to the gold answer. Given these preference pairs, we are thus able to convert the evaluation task into a verifiable task (i.e., predicting the better response), enabling the use of RL with verifiable rewards.

Pairwise LLM judges frequently suffer from a phenomenon called position bias, wherein the verdict changes if the order of the responses is swapped (Wang et al., 2024c; Chen et al., 2024). To mitigate this issue, we construct training data with both orders of response pairs –  $(x, a, b)$  and  $(x, b, a)$  – which are then fed into the corresponding thinking seed prompt (Figure 7 in Appendix A). Crucially, we construct position-agnostic batches of training data i.e., for the same input, we process both orders of responses in one batch (see Figure 3). In Section 2.3, we describe how this helps us design verdict consistency rewards for training J1 that can mitigate position bias in pairwise judges.

## 2.3 Reward Modeling

Following the training data construction, we describe J1’s reward modeling for RL training. We adopt a rule-based reward system that consists of two main types of rewards.

**Verdict Correctness Reward.** J1’s primary reward is based on the accuracy of the final verdict. If a judge correctly predicts the better response, it receives a reward of 1 and 0, otherwise.

**Verdict Consistency Reward.** To further encourage consistent reasoning and judgment, we introduce verdict (positional) consistency reward. In particular, we only assign a reward of 1 when the model produces correct judgments for both input orders of the same response pair (i.e.,  $(x, a, b)$  and  $(x, b, a)$ ).

Similar to DeepSeek-R1, we also experiment with format rewards to ensure that the thought tokens are

enclosed in “<think>” and “</think>” tags, but do not observe noticeable improvements. In [Section 4.2](#), we provide a detailed analysis of the effect of these reward types.

## 2.4 Primary J1 Recipe and Models

J1 jointly optimizes thoughts and judgments using GRPO, which makes RL training more efficient by removing the dependency on a separate critic model and instead estimating the baseline from group scores.

**Pairwise LLM-as-a-Judge with Verdict (PaV).** Our primary recipe,  $J1_{\text{PaV}} : \text{prompt}_{\text{PaV}}(x, a, b) \rightarrow (t, y)$ , receives a user question and a response pair, and generates thought tokens and the preferred response (as the final verdict). Using this recipe, we develop two models, J1-Llama-8B and J1-Llama-70B, trained starting from Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct ([Grattafiori et al., 2024](#)), respectively. Although we report our main benchmark results with  $J1_{\text{PaV}}$  in [Section 4.1](#), we also consider modified recipes for training other Thinking-LLM-as-a-Judge models that vary in problem formulation, thinking prompts, or reward modeling.

## 2.5 Other RL Training Recipes for Thinking-LLM-as-a-Judge

In addition to our primary ‘Pairwise-J1 with Verdict’ setting described above, we also introduce three other Thinking-LLM-as-a-Judge settings and develop their corresponding training recipes. In [Section 4.2](#), these modified recipes are compared to our primary  $J1_{\text{PaV}}$  recipe.

**Pairwise LLM-as-a-Judge with Scores (PaS).** Instead of directly generating a verdict, our pairwise score-based variant  $J1_{\text{PaS}} : \text{prompt}_{\text{PaS}}(x, a, b) \rightarrow (t, s_a, s_b)$ , generates real-valued scores  $s_a, s_b$  for response  $a$  and  $b$ , respectively. The response that obtains the higher score is chosen as the final verdict. [Figure 9](#) in [Appendix A](#) shows the corresponding thinking prompt. To train a model with this recipe, we replace the verdict-based reward with a score-based reward: a binary reward is assigned depending on whether the predicted scores are consistent with the gold verdict.

**Pairwise LLM-as-a-Judge with Scores&Verdict (PaVS).** In another pairwise variant  $J1_{\text{PaVS}} : \text{prompt}_{\text{PaVS}}(x, a, b) \rightarrow (t, s_a, s_b, y)$ , the model generates scores for both responses as well as the final verdict, where the generated scores are interpreted as observations of unknown latent variables to help with the pairwise judgment task; therefore  $y$  is directly used as the final verdict. Consequently, the reward is also computed using only the final verdict (and not the intermediate scores). [Figure 10](#) in [Appendix A](#) shows the corresponding thinking prompt.

**Pointwise LLM-as-a-Judge (PoS).** Finally, we also introduce a pointwise judge,  $J1_{\text{PoS}} : \text{prompt}_{\text{PoS}}(x, a) \rightarrow (t, s)$ , which takes an instruction  $x$  and a single response  $a$  as input, and outputs a score  $s$  that reflects the quality or reward of the response. Unlike pairwise judges, pointwise judges are inherently *consistent*. We train  $J1_{\text{PoS}}$  via *distant supervision* with the same pairwise training data as used for all our pairwise variants. Each preference pair is split into two separate pointwise samples (see [Figure 3](#)) and the model is trained with a seed thinking prompt that assigns a score between 0 and 10 to a given response (see [Figure 11](#) in [Appendix A](#)). Both scores are evaluated jointly, and the model receives a reward of 1 if the scores are consistent with the gold verdict. Since preference rankings are significantly easier to obtain than pointwise annotations, Pointwise-Thinking-LLM-as-a-Judge represents one of the novel contributions of our work.

## 3 Experimental Setup

**Training.** We implement J1 on top of `ver1` ([Sheng et al., 2024](#)). All variants are trained on the 22K synthetic preference pairs described in [Section 2.2](#). [Appendix B](#) includes other details of our experimental setup.

**Evaluation.** We evaluate J1 on five pairwise judgment benchmarks, covering both verifiable and non-verifiable tasks. These benchmarks include multilingual instructions and responses from a wide range of LLMs.

- **Preference Proxy Evaluations (PPE)** ([Frick et al., 2025](#)) is a large-scale benchmark that links reward models to real-world human preference performance. It consists of two subsets: (i) **PPE Preference** (10.2K samples), human preference pairs from Chatbot Arena featuring 20 LLMs in 121+ languages, and (ii) **PPE Correctness** (12.7K samples), response pairs from four models across popular verifiable benchmarks (MMLU-Pro, MATH,

Models	#Training Pref. Pairs	PPE Overall	PPE Preference	Overall	MMLU-Pro	PPE Correctness			
						MATH	GPQA	MBPP-Plus	IFEval
Open and Closed LLM-as-a-Judge									
Llama-3.1-8B-Instruct	–	55.5	56.4	54.7	56.3	62.9	51.4	50.1	52.8
GPT-4o <sup>†</sup>	–	62.3	67.1	57.6	–	–	–	–	–
Llama-3.3-70B-Instruct	–	65.8	65.9	65.7	72.1	73.1	61.2	59.6	62.3
SOTA Scalar Reward Models									
Armo-8B-v0.1 <sup>†</sup>	1000K	60.9	60.6	61.2	66.0	71.0	57.0	54.0	58.0
Skywork-Reward-Gemma-2-27B <sup>†</sup>	80K	55.6	56.6	54.7	55.0	46.2	44.7	69.1	58.3
DeepSeek-BTRM-27B <sup>†</sup>	237K	67.5	<b>68.3</b>	66.7	68.8	73.2	56.8	68.8	66.0
SOTA Generative Reward Models									
DeepSeek-GRM-27B <sup>†</sup>	237K	62.2	64.7	59.8	64.8	68.8	55.6	50.1	59.8
DeepSeek-GRM-27B (MetaRM voting@32) <sup>†</sup>	237K	65.2	67.2	63.2	68.1	70.0	56.9	50.8	70.4
EvalPlanner-Llama-8B	22K <sup>‡</sup>	54.1	55.5	52.8	57.0	59.0	50.3	47.7	50.0
EvalPlanner-Llama-70B	22K <sup>‡</sup>	67.9	65.6	70.2	78.4	81.7	64.4	62.2	64.3
<b>J1-Llama-8B</b>	22K <sup>‡</sup>	59.8	60.3	59.2	65.6	70.0	53.2	53.1	54.0
<b>J1-Llama-8B (SC@32)</b>	22K <sup>‡</sup>	61.3	60.6	61.9	67.5	76.6	55.7	54.6	54.9
<b>J1-Llama-70B</b>	22K <sup>‡</sup>	69.6	66.3	72.9	79.0	86.0	65.9	66.0	<b>67.3</b>
<b>J1-Llama-70B (SC@32)</b>	22K <sup>‡</sup>	<b>70.4</b>	67.0	<b>73.7</b>	<b>79.9</b>	<b>88.1</b>	<b>66.5</b>	<b>66.5</b>	67.2

**Table 1** Results on PPE comparing Pairwise J1-Llama-8B (trained from Llama-3.1-8B-Instruct) and J1-Llama-70B (trained from Llama-3.3-70B-Instruct) with state-of-the-art LLM-as-a-Judge and reward models. <sup>†</sup>: Results taken from Liu et al. (2025b) and Frick et al. (2025). <sup>‡</sup>: Training data only contains synthetically constructed preference pairs.

GPQA, MBPP-Plus, IFEval). The first subset evaluates subjective preferences, while the second tests alignment in Best-of-N tasks.

- **JudgeBench** (Tan et al., 2025) (350 preference pairs) contains challenging response pairs that span knowledge, reasoning, math, and coding categories. Following Tan et al. (2025), we report results on the subset where the responses are generated by GPT-4o.
- **RM-Bench** (Liu et al., 2025a) (4K samples) assesses the robustness of reward models based on their sensitivity and resistance to subtle content differences and style biases.
- **FollowBenchEval** (Saha et al., 2025) (205 preference pairs) tests reward models for their ability to validate multi-level constraints in LLM responses (e.g., “Write a one sentence summary (less than 15 words) for the following dialogue. The summary must contain the word ‘stuff’...”).
- **RewardBench** (Lambert et al., 2024) (3K samples), similar to JudgeBench, consists of preference pairs from 4 categories of prompts: chat, chat-hard, safety, and reasoning.

Consistent with prior work, we report accuracy over a single random ordering of paired responses for PPE, RewardBench, and RM-Bench. For JudgeBench and FollowBenchEval, we instead report *position-consistent accuracy*, where a sample is deemed correct only if the judge produces the correct verdict under both response orders. A more detailed analysis of positional consistency is presented in Section 4.2. Model selection is based on overall accuracy on RewardBench. Unless otherwise specified, inference is performed using vLLM (Kwon et al., 2023) with greedy decoding.

**Baselines.** We compare J1 to different categories of baselines: (i) LLMs acting as judges in a zero-shot manner (e.g., Llama-3.3-70B-Instruct, GPT-4o (Hurst et al., 2024)), (ii) state-of-the-art scalar reward models (e.g., DeepSeek-BTRM-27B (Liu et al., 2025b), Armo (Wang et al., 2024a), Skywork-Reward-Gemma-2-27B (Shiwen et al., 2024)), (iii) state-of-the-art generative reward models that belong to the same category as J1 (e.g., EvalPlanner (Saha et al., 2025), DeepSeek-GRM-27B (Liu et al., 2025b)), and (iv) general Reasoning/Thinking-LLMs (e.g., DeepSeek-R1-Distilled-Llama (Guo et al., 2025), DeepSeek-R1, and OpenAI-o1-mini).

## 4 Results

### 4.1 Main Results on Benchmarks with Pairwise-J1

**Comparison of J1 with state-of-the-art methods on PPE.** Table 1 compares our Pairwise-J1 models with different baselines on the PPE benchmark. J1-Llama-70B, our larger model, obtains an overall accuracy of 69.6, outperforming all previous methods, including those trained on much more data (see column 2).



Models	PPE Preference	PPE Correctness	RewardBench	RM-Bench	JudgeBench <sup>†</sup>	FollowBenchEval <sup>†</sup>
Llama-3.1-8B-Instruct	56.4	54.7	69.5	54.0	32.3	30.2
DeepSeek-R1-Distilled-Llama-8B	59.4	58.4	73.7	69.3	30.5	40.9
EvalPlanner-Llama-8B	55.5	53.0	83.0	68.1	30.2	45.3
<b>J1-Llama-8B</b>	<b>60.3</b>	59.2	<b>85.7</b>	<b>73.4</b>	42.0	48.3
Llama-3.3-70B-Instruct	65.9	65.7	85.4	69.5	48.6	52.2
DeepSeek-R1-Distilled-Llama-70B	66.1	71.1	86.9	80.8	46.0	54.6
EvalPlanner-Llama-70B	65.6	70.2	<b>93.8</b>	82.1	56.6	65.4
<b>J1-Llama-70B</b>	<b>66.3</b>	<b>72.9</b>	93.3	<b>82.7</b>	<b>60.0</b>	<b>69.3</b>
OpenAI-o1-mini	65.7	71.3	87.1	80.8	64.2	62.9
DeepSeek-R1-671B	68.0	76.5	90.6	88.6	68.9	71.7

**Table 2** Results on five reward modeling benchmarks comparing Pairwise-J1 at 8B and 70B scale to a state-of-the-art Thinking-LLM-as-a-Judge model (EvalPlanner) and general Thinking-LLMs (o1-mini, distilled-R1, and R1). We report the default metric for each benchmark, where †: JudgeBench and FollowBenchEval use positional consistent accuracy.

Compared to related competing approaches that are generative reward models, J1-Llama-70B outperforms both EvalPlanner (Saha et al., 2025) by 1.7% (67.9  $\rightarrow$  69.6), and DeepSeek-GRM-27B (Liu et al., 2025b) by 7.1% (62.2  $\rightarrow$  69.3). First, this shows the effectiveness of J1’s training methodology and use of online RL (GRPO), compared to EvalPlanner, which is trained on the same data but with two iterations of DPO. Second, this also highlights the effectiveness of J1’s high-quality synthetic preference pairs, compared to the data used to train DeepSeek-GRM-27B. The latter is first fine-tuned on 1270K judge data, followed by stages of Reinforcement Learning on 237K samples and further scaling at test time with a meta reward model across 32 generations. J1 shows particularly strong results on the PPE Correctness subset that spans popular reasoning benchmarks, thereby highlighting its utility as a reward model for Best-of-N alignment. At a smaller scale, J1-Llama-8B is competitive with Armo-8B (scalar RM) and outperforms EvalPlanner-8B and a larger Skywork-Reward-Gemma-2-27B by significant margins (54.1  $\rightarrow$  59.8 and 55.6  $\rightarrow$  59.8, respectively).

**Test-time Scaling of J1.** We further scale J1 up at test time with techniques such as self-consistency (Wang et al., 2023). As shown in Table 1 (see “w/ SC@32” rows), we obtain further improvements of up to 1.5% at both scales of J1 by sampling 32 CoTs (with a temperature of 1.0) and using self-consistency to obtain the final verdict. In subsection 4.2, we conduct a more detailed analysis of test-time scaling of J1.

**Comparison of J1 with Thinking-LLMs.** Given the superiority of the earlier Thinking-LLM method EvalPlanner over other baselines, Table 2 compares J1 to EvalPlanner on all other benchmarks at both scales (8B/70B). Additionally, we compare J1 to general Thinking-LLMs (e.g., DeepSeek-R1-Distill-Llama, DeepSeek-R1, and OpenAI-o1-mini). DeepSeek-R1-Distilled-Llama is SFT-ed with 800K long CoTs from DeepSeek-R1 (a much larger 671B MoE model), starting from the same base models as J1. Thus, in a head-to-head comparison where the base models are the same, J1 outperforms DeepSeek-R1-Distilled-Llama across all benchmarks by large margins. In particular, on the harder JudgeBench benchmark that requires evaluating GPT-4o responses, J1-Llama-70B obtains an absolute 14% improvement (46.0  $\rightarrow$  60.0) over R1-Distilled-Llama-70B. J1 also closes the gap to the much larger R1 model, even outperforming it on RewardBench by 2.7%.

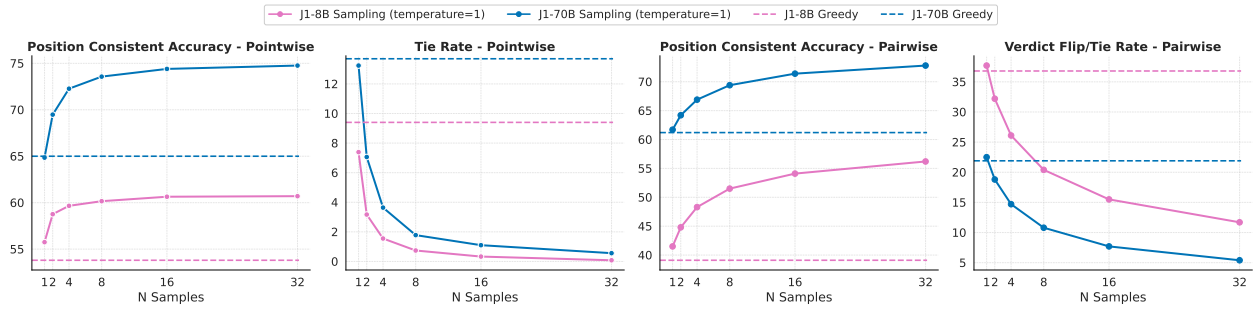
To further analyze this observation, Table 8 in Appendix C shows a detailed breakdown of the results for the individual categories in RewardBench. Although R1 expectedly excels in reasoning, J1 outperforms it in evaluating non-verifiable prompts like those in Chat-Hard and Safety. Overall, our results suggest that J1 is a generalist judge that can be used for evaluating responses to both verifiable and non-verifiable prompt tasks, and in different stages of the LLM development process.

## 4.2 Ablations and Analyses of Pairwise-J1 and Pointwise-J1

In Section 2.5, we introduced several alternative training recipes for Thinking-LLM-as-a-Judge models, including the Pointwise-J1 variant. In the following experiments, we systematically analyze the impact of these recipes, focusing on three key research questions: (i) How does Pointwise-J1 compare to Pairwise-J1 in mitigating position bias? (ii) How can we design effective rewards for reinforcement learning in Thinking-LLM-as-a-Judge models? (iii) What is the impact of different thinking prompts on model behavior? Unless otherwise specified, we use Pairwise-J1 to refer to the model trained under our primary *Pairwise with Verdict* setting. Other variants are discussed explicitly where relevant.

Models	Type	PPE Correctness				JudgeBench			
		(a, b) Acc $\uparrow$	(b, a) Acc $\uparrow$	Consistent Acc $\uparrow$	Verdict- Flip/Ties $\downarrow$	(a, b) Acc $\uparrow$	(b, a) Acc $\uparrow$	Consistent Acc $\uparrow$	Verdict- Flip/Ties $\downarrow$
Llama-3.1-8B-Instruct	Pairwise	54.7	54.1	30.2	44.1	67.4	42.3	32.3	37.4
EvalPlanner-Llama-8B	Pairwise	53.0	52.4	37.3	25.7	38.2	62.8	30.3	30.5
<b>J1-Llama-8B</b> ( <i>random single-order data</i> )	Pairwise	58.3	57.6	38.3	36.7	48.3	59.4	36.6	32.9
<b>J1-Llama-8B</b> ( <i>both-order data</i> )	Pairwise	59.2	58.4	39.1	36.8	63.1	51.4	42.0	27.7
<b>J1-Llama-8B</b> ( <i>verdict consistency reward</i> )	Pairwise	58.4	58.2	43.9	28.7	52.3	64.6	45.4	26.0
<b>J1-Llama-70B</b> ( <i>both-order data</i> )	Pairwise	72.9	72.3	61.2	21.9	68.9	72.6	<b>60.0</b>	20.6
Llama-3.1-8B-Instruct	Pointwise	—	—	42.9	25.1	—	—	46.3	26.9
<b>J1-Llama-8B</b>	Pointwise	—	—	53.8	9.4	—	—	57.7	8.0
<b>J1-Llama-70B</b>	Pointwise	—	—	<b>65.0</b>	13.7	—	—	<b>60.0</b>	16.6

**Table 3** Results on PPE Correctness and JudgeBench, comparing various position-bias mitigation strategies (for Pairwise-J1) and Pointwise-J1. *Consistent Accuracy* measures the proportion of examples where the pairwise judge gives the correct verdict in both response orders. *Verdict Flip* shows the percentage of cases where a *pairwise* judge’s verdict flips when the order of responses is swapped, regardless of correctness. *Ties* indicate the proportion of examples where a *pointwise* judge assigns equal scores to both responses. All the generations are with greedy decoding.

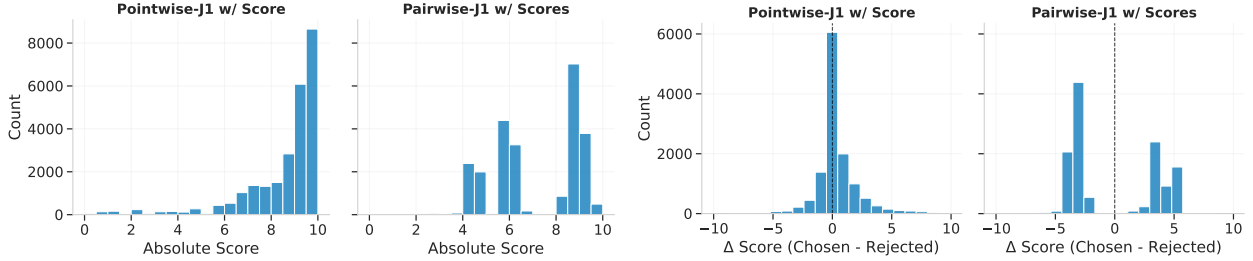


**Figure 4** Test-time scaling of Pairwise-J1 and Pointwise-J1 on the PPE Correctness benchmark. We show greedy decoding results in dotted lines for comparison. As we sample more  $N$ , (i) position-consistent accuracy increases and (ii) tie rate decreases, for both pairwise and pointwise models, at both 8B and 70B scales.

**Pairwise-J1 vs Pointwise-J1 and their Effects on Positional Consistency.** There are two main ways of mitigating position bias in judgments – either by improving a pairwise judge itself (e.g., by training on both orders of data, adding consistency-based rewards, etc., like we do in J1) or by training a *Pointwise-J1* model that, by design, is position-consistent. Pointwise judges, while consistent, are score-based and thus could suffer from ties if both responses receive the exact same score.

To understand this long-standing issue of judgment bias, in [Table 3](#), we report (i) individual accuracy for both orders of responses, (ii) position-consistent accuracy, and (iii) verdict-flips/ties. We use “verdict flip” and “tie” to refer to the same metric which is the fraction of samples where either the verdict changes based on the response order (for pairwise judges) or both responses are scored equally (for pointwise judges). A *consistently correct* judge is thus expected to show higher position-consistent accuracy and lower verdict-flips/ties. We observe that when judgment quality is measured by a stricter position-consistency accuracy, *Pointwise-J1* outperforms *Pairwise-J1*. The same trend holds between Pointwise and Pairwise-Llama-Instruct as well. While our bias mitigation techniques on *Pairwise-J1* help, our best *Pairwise-J1* model still suffers from a significant fraction of inconsistencies (up to 20%) whereas for *Pointwise-J1*, the fraction of ties is as low as 9%, leading to improvement in position-consistent accuracy. Scaling up our *Pointwise-J1* model to 70B improves position-consistent accuracy further.

**Effect of Test-time Scaling on Position-Consistent Accuracy and Ties.** In [Table 1](#), we showed results with one kind of test-time scaling of J1 – self-consistency over multiple verdicts. For a *Pointwise-J1* model, we can also scale it up at test-time by averaging scores across multiple generations to get a more accurate estimate of these scores. In [Figure 4](#), we plot position-consistent accuracy and ties as a function of number of generations  $N$  (refer to [Table 7](#) in [Appendix C](#) for accuracy numbers). We observe improved position-consistent accuracy and reduction in ties for both Pairwise- and Pointwise-J1 at both 8B and 70B scales. To the best of our knowledge, this is one of the first comprehensive analyses of Pointwise versus Pairwise judges, when both are



**Figure 5** Distribution of Absolute Scores and  $\Delta\text{Score}$  ( $\text{Chosen} - \text{Rejected}$ ) generated by the 8B Pairwise-J1 (w/ Scores) and Pointwise-J1 models on the PPE Correctness benchmark. Pairwise-J1 exhibits sparser score distribution and larger score differences between Chosen and Rejected (ground-truth) responses.

Pairwise-J1 8B Variants	PPE Preference	PPE Correctness	RewardBench	JudgeBench	RM-Bench	FollowBenchEval
<i>with Different Rewards</i>						
Positive Reward for Correct Verdict	60.3	59.2	85.7	42.0	73.4	48.3
+ Negative Reward for Incorrect Verdict	60.0	59.1	85.4	44.9	70.8	42.0
+ Format Reward	60.2	58.3	85.6	40.3	71.8	49.3
<i>with Different Seed Prompts</i>						
Thinking (default - Figure 8)	60.3	59.2	85.7	42.0	73.4	48.3
Plan & Execution (Figure 7)	59.1	58.9	85.8	44.3	71.8	50.2

**Table 4** Results of Pairwise-J1 models trained with different reward schemes and seed prompts.

trained using the same data.

**Effect of Reward Schemes and Seed Prompts for Training J1.** In Table 4, we first study the effect of different rewards for Pairwise-J1 models. We obtain best results when only assigning positive rewards to correct verdicts – adding additional format rewards or negative rewards for incorrect verdicts marginally degrades performance. Next, we also analyze the effect of two different seed prompts that are used to elicit “thinking” in J1 models. Our default J1 *Thinking* prompt is motivated by DeepSeek-R1. Additionally, similar to EvalPlanner (Saha et al., 2025), we experiment with a prompt that instructs the model to first *plan* for the evaluation recipe, then *execute* the evaluation according to that recipe and the response(s), before generating the final verdict (Figure 8 in Appendix A). We find that J1 is robust to such choices, performing comparably with both prompts. In fact, with a simpler Thinking prompt, the model tends to generate richer reasoning traces, including evaluation criteria, reference answers, re-evaluations, and detailed comparisons (see Figure 2).

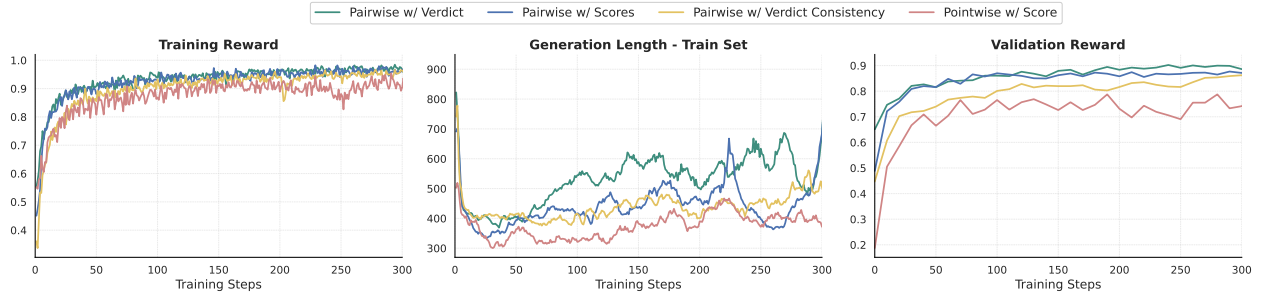
**Score Distribution of Pointwise-J1 and Pairwise-J1.** In Figure 5, we show the distribution of (i) absolute scores and (ii) score difference between the chosen and rejected (ground-truth) responses, as generated by Pairwise-J1 w/ Scores and Pointwise-J1 models. Pairwise-J1 exhibits a sparser distribution of scores (clustered around a few scores) and a larger gap compared to Pointwise-J1. This difference arises from their training objectives: Pairwise-J1 directly contrasts both responses within the same input, allowing it to more clearly differentiate between them, which is reflected in the score assignments. In contrast, Pointwise-J1 is trained with distant supervision that attends to only one response at a time, making it harder to learn fine-grained comparative judgments.

**Reward and Thought Length Analysis of J1.** Figure 6 illustrates the training and validation reward, as well as average generation length throughout different stages of J1 training. As training progresses, the thought lengths of most pairwise judges converge at around 500 tokens, while the pointwise judge tends to generate shorter outputs, typically between 300 and 400 tokens due to the absence of *comparison*-style tokens. Training rewards for the pairwise variants exhibit a similar steady increase before converging. In Table 5 of Appendix C, we provide a comparison of different Pairwise-J1 variants, all of which show comparable performance.

## 5 Related Work

**Reward Models.** Reward models have been instrumental in both training-time (Ouyang et al., 2022; Lambert et al., 2024) and test-time (Snell et al., 2025) alignment of Large Language Models. Traditional reward models are typically trained with the Bradley-Terry objective and output a scalar score indicating the reward of





**Figure 6** Reward and average generation length during training for different J1-Llama-8B models. Pointwise-J1 is trained via *distant supervision* derived from pairwise preference data.

the response. This design frequently results in poor calibration and generalization across different prompts and responses (Sun et al., 2025; Zhang et al., 2025). Furthermore, such discriminative models do not fully leverage the generative capabilities of LLMs and therefore cannot be scaled up at test time, e.g., with long chain-of-thought or multiple generations (Wang et al., 2024b; Shiwen et al., 2024). As a potential solution, generative reward models have emerged, which we discuss below.

**LLM-as-a-Judge and Generative Reward Models.** LLM-as-a-Judge and Generative Reward Models (GRMs) conceptually share a similar motivation – the language modeling head in LLMs is used to also output chain-of-thought (CoT) reasoning (in the form of critiques) before generating preference judgments or rewards (Kim et al., 2024a,b; Ankner et al., 2024; Mahan et al., 2024; Ye et al., 2024; Yu et al., 2025; Zhang et al., 2025; Saha et al., 2025). Rewards in such models could either come from training a separate reward head (typically done in GRMs) or from the LM head itself generating real-valued scores as tokens (typically done in LLM-as-a-Judge). Prior work building LLM judges has mostly relied on either prompting (Zheng et al., 2023; Saha et al., 2024), rejection finetuning on self-generated CoTs (Wang et al., 2024d), or preference finetuning on CoT pairs using DPO (Mahan et al., 2024; Trivedi et al., 2024; Saha et al., 2025).

Recently, in some concurrent studies, DeepSeek-GRM (Liu et al., 2025b), JudgeLRM (Chen et al., 2025a), and RM-R1 (Chen et al., 2025b) use Reinforcement Learning for building reasoning judge models. We compare J1 to these methods and show that J1 achieves superior performance with  $10\times$  less data. This is achieved via J1’s methodical novelty that span several axes. First, it is a training recipe that is based only on *synthetic* data. Second, it focuses on mitigating position bias (a long-standing issue in LLM-as-a-Judge development), leading to the proposal of novel consistency rewards and Pointwise-J1 models trained with *pairwise supervision only*. Consequently, we are able to comprehensively study different J1 variants that vary across sizes, modeling choices, seed prompts, and reward strategies, enabling us to draw important conclusions on building generalist thinking-judge models with state-of-the-art results.

**Reinforcement Learning with Verifiable Rewards.** J1 draws inspiration from the recent advancements in improving reasoning through Reinforcement Learning with verifiable rewards. Online optimization algorithms such as GRPO, when combined with accurate and robust rewards, have been shown to elicit enhanced *reasoning* in LLMs (Guo et al., 2025; Team et al., 2025). In our approach, we construct preference pairs and assign verifiable rewards based on the correctness of the model’s judgments. By optimizing over the generated thinking traces, J1 encourages LLMs to spend more test-time compute before deriving scores and judgments.

## 6 Conclusion

We proposed J1, an RL recipe for training Thinking-LLM-as-a-Judge models. Our key innovation was in converting the judgment task into a verifiable task for all kinds of task prompts, themselves both verifiable and non-verifiable, and then optimizing the thoughts and judgments using an online RL method. We trained J1-Llama-8B and J1-Llama-70B, two generalist judge models that outperformed all baselines at their respective model sizes, o1-mini, and even a much larger R1 model on non-verifiable tasks. Using only pairwise supervision, we also trained Pointwise-J1 models that proved to be effective in mitigating position bias, thereby highlighting the potential of both Pairwise and Pointwise Thinking-LLM-as-a-Judge.

## References

- 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=CljYUv1lRW>.
- Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or LLMs as the Judge? A Study on Judgement Bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8301–8327, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.474. URL <https://aclanthology.org/2024.emnlp-main.474>.
- 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*, 2025a. URL <https://arxiv.org/abs/2504.00050>.
- 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*, 2025b. URL <https://arxiv.org/abs/2505.02387>.
- 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=cbttLt094Q>.
- 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>.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A Survey on LLM-as-a-Judge. *arXiv preprint arXiv:2411.15594*, 2024. URL <https://arxiv.org/abs/2411.15594>.
- 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. URL <https://arxiv.org/abs/2501.12948>.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.
- 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>.
- Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing Fine-grained Evaluation Capability in Language Models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL <https://openreview.net/forum?id=8euJaTveKw>.
- Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An Open Source Language Model Specialized in Evaluating Other Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.248. URL <https://aclanthology.org/2024.emnlp-main.248>.
- 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, 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. URL <https://arxiv.org/abs/2403.13787>.

- Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. RM-Bench: Benchmarking Reward Models of Language Models with Subtlety and Style. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL <https://openreview.net/forum?id=QEHRmQPBdd>.
- 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>.
- 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>.
- 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://proceedings.neurips.cc/paper\\_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html](https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html).
- Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-Solve-Merge Improves Large Language Model Evaluation and Generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8345–8363, 2024. URL <https://aclanthology.org/2024.naacl-long.462>.
- Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. Learning to Plan & Reason for Evaluation with Thinking-LLM-as-a-Judge. *arXiv preprint arXiv:2501.18099*, 2025. URL <https://arxiv.org/abs/2501.18099>.
- 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. URL <https://arxiv.org/abs/2402.03300>.
- 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>.
- Tu Shiwen, Zhao Liang, Chris Yuhao Liu, Liang Zeng, and Yang Liu. Skywork Critic Model Series. <https://huggingface.co/Skywork>, September 2024. URL <https://huggingface.co/Skywork>.
- Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=4FWAwZtd2n>.
- Hao Sun, Yunyi Shen, and Jean-Francois Ton. Rethinking Reward Modeling in Preference-based Large Language Model Alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=rfdble10qm>.
- 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>.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. URL <https://arxiv.org/abs/2501.12599>.
- Prapti Trivedi, Aditya Gulati, Oliver Molenschot, Meghana Arakkal Rajeev, Rajkumar Ramamurthy, Keith Stevens, Tanveesh Singh Chaudhery, Jahnvi Jambholkar, James Zou, and Nazneen Rajani. Self-Rationalization Improves LLM as a Fine-grained Judge. *arXiv preprint arXiv:2410.05495*, 2024. URL <https://arxiv.org/abs/2410.05495>.
- Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.620. URL <https://aclanthology.org/2024.findings-emnlp.620>.
- Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. Direct Judgement Preference Optimization, 2024b. URL <https://arxiv.org/abs/2409.14664>.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large Language Models are not Fair Evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume*

- 1: *Long Papers*), pages 9440–9450, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.511. URL <https://aclanthology.org/2024.acl-long.511>.
- 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*, 2024d. URL <https://arxiv.org/abs/2408.02666>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gall . Improving Reward Models with Synthetic Critiques. *arXiv preprint arXiv:2405.20850*, 2024. URL <https://arxiv.org/abs/2405.20850>.
- Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, Melanie Kambadur, Dhruv Mahajan, and Rui Hou. Self-Generated Critiques Boost Reward Modeling for Language Models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11499–11514, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL <https://aclanthology.org/2025.naacl-long.573>.
- 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=B18u7ZR1bM>.
- 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](https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html).

### Thinking Seed Prompt Template for Training Pairwise J1 with Verdict

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.

First, provide your reasoning within `<think>` and `</think>` tags. This should include your evaluation criteria for a high-quality response, a detailed comparison of the two responses, and when helpful, a reference answer as part of your evaluation. Be explicit in your thought process, referencing your criteria and explaining how each response aligns with or deviates from them.

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.

Finally, provide your verdict within `<answer>` and `</answer>` tags, strictly following this format:

- `<answer> [[A]] </answer>` if Assistant A is better
- `<answer> [[B]] </answer>` if Assistant B is better

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 7** Thinking seed prompt template for Pairwise-J1 with Verdict.

## A Prompt Templates

Figure 7 shows the seed prompt to train our primary J1 recipe: Pairwise-J1 with verdict. Figure 8 shows an alternative seed prompt for training a similar Pairwise-J1 with Verdict setup. Motivated by EvalPlanner (Saha et al., 2025), this prompt instructs the model to first *plan* the evaluation recipe and then *execute* it as part of the thinking process. Figure 9 and Figure 10 show our prompts for “Pairwise-J1 with Scores” and “Pairwise-J1 with Scores&Verdict” variants respectively. Finally, we adapt our pairwise prompts to a pointwise prompt used to train our Pointwise-J1 model that instructs the model to think and assign real-valued scores between 0 and 10, shown in Figure 11.

## B Experimental Setup

For training, the policy actor generates 5 rollouts per prompt using ancestral sampling with temperature 1.0. Training regime uses a learning rate of  $1e-6$  (decayed to  $3e-7$  in later steps for pairwise J1-Llama-70B), and a total batch size of 512. The maximum sequence length is set to 4096 tokens for both inputs and outputs.

We experimented with different KL coefficients from  $\{0.1, 0.01, 0.001, 0\}$  for J1-Llama-8B, and selected 0.01 as the best-performing value based on development set accuracy. For J1-Llama-70B, we set the KL coefficient to 0 to encourage more exploration. Preliminary experiment with entropy bonus during training showed degraded performance: the model tends to generate longer but more repetitive output. See Table 6 in Appendix C for comparison of KL penalty and entropy bonus.

We use  $8 \times A100$  and  $64 \times A100$  GPUs to train J1-Llama-8B, and J1-Llama-70B respectively, and  $8 \times A100$  for inference. Tensor parallelism is set to 8 for both training and inference. During inference, we maintain a maximum generation length of 4096 tokens. The default generation uses greedy decoding, and for inference-time scaling we use sampling with top-p of 0.95 and temperature of 1.



### EvalPlanner-style Seed Prompt Template for Training Pairwise J1 with Verdict

You are given a user question and two responses from AI assistants. Your task is to act as an impartial judge and determine which response better follows the user's instructions and provides a higher-quality answer.

First, build an evaluation plan that can then be executed to assess the response quality. Whenever appropriate, you can choose to also include a step-by-step reference answer as part of the evaluation plan. Enclose your evaluation plan between the tags `<plan>` and `</plan>`.

Next, execute the plan step-by-step to evaluate the two responses. Avoid copying the plan when doing the evaluation. Please also only stick to the generated plan and provide an explanation of how the plan is executed to compare the two responses. 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. Enclose your plan execution between the tags `<execution>` and `</execution>`.

Finally, output your final verdict by strictly following this format:

- `<answer> [[A]] </answer>` if Assistant A is better
- `<answer> [[B]] </answer>` if Assistant B is better

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** EvalPlanner-style (plan + execution) Prompt template for Pairwise-J1 with Verdict.

### Thinking Seed Prompt Template for Training Pairwise-J1 with Scores

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.

First, provide your reasoning within `<think>` and `</think>` tags. This should include your evaluation criteria for a high-quality response, a detailed comparison of the two responses, and when helpful, a reference answer as part of your evaluation. Be explicit in your thought process, referencing your criteria and explaining how each response aligns with or deviates from them.

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.

Finally, assign the assistant's 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 9** Thinking seed prompt template for training Pairwise-J1 with Scores.

### Thinking Seed Prompt Template for Training Pairwise J1 with Verdict and Score

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.

First, provide your reasoning within `<think>` and `</think>` tags. This should include your evaluation criteria for a high-quality response, a detailed comparison of the two responses, and when helpful, a reference answer as part of your evaluation. Be explicit in your thought process, referencing your criteria and explaining how each response aligns with or deviates from them.

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.

Finally, assign the assistant's 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>`.

Finally, provide your verdict within `<answer>` and `</answer>` tags, strictly following this format:

- `<answer> [[A]] </answer>` if Assistant A is better
- `<answer> [[B]] </answer>` if Assistant B is better

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 10** Thinking seed prompt template for training Pairwise-J1 with Verdict and Score.

### Thinking Seed Prompt Template for Training Pointwise J1

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 enclose your reasoning within `<think>` and `</think>` tags. Your reasoning should include your evaluation criteria, a clear understanding of what an ideal response would look like for this particular question, and a concrete example of such an ideal or reference answer if possible. Then compare the assistant's response to your ideal or reference answer, explaining how it aligns with or deviates from your expectations. Be specific and avoid vague or overly general judgments. Remain as objective as possible.

Finally, assign the assistant's response a score from 0 to 10, using either an integer or a decimal with up to 0.1 precision. A higher score should indicate a higher-quality response. Enclose the score within `<score>` and `</score>` tags.

Format your output like this:

`<think> your_thinking_process </think>`  
`<score> your_score </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 11** Thinking seed prompt template for training Pointwise-J1.

Pairwise-J1 8B Variants	PPE Preference	PPE Correctness	RewardBench	JudgeBench	RM-Bench	FollowBenchEval
w/ Verdict : $(x, a, b) \rightarrow (t, y)$	60.3	59.2	85.7	42.0	73.4	48.3
w/ Scores: $(x, a, b) \rightarrow (t, s_a, s_b)$	60.7	59.7	85.8	41.7	72.3	46.3
w/ Scores&Verdict: $(x, a, b) \rightarrow (t, s_a, s_b, y)$	59.7	59.3	85.1	41.4	71.5	41.0

**Table 5** Results of Pairwise-J1 models trained with different recipes.  $x$  : input instruction,  $a, b$  : pair of responses,  $t$  : intermediate thought,  $y$ : verdict,  $s_a, s_b$ : scores.

Pairwise-J1 8B Variants	PPE Preference	PPE Correctness	RewardBench	JudgeBench	RM-Bench	FollowBenchEval
w/ KL Penalty	60.3	59.2	85.7	42.0	73.4	48.3
w/o KL Penalty	59.8	59.4	84.9	42.9	69.8	48.3
w/o Entropy Bonus	60.3	59.2	85.7	42.0	73.4	48.3
w/ Entropy Bonus	59.3	58.9	84.1	39.4	71.7	42.4

**Table 6** Ablation studies on Pairwise-J1 with verdict with KL penalty and entropy bonus.

## C Additional Results

**Comparison of Different Pairwise-J1 Models.** In Table 5, we compare the three Pairwise-J1 variants, that as part of the final answer, generate either: (i) only the final verdict, (ii) only real-valued scores for both responses, or (iii) both. We observe that predicting the verdict (without the scores) performs as well as other variants. Having access to scores, however, has other advantages e.g., in quantifying the degree of preference or to rank across multiple responses.

**Effect of KL Penalty and Entropy Bonus in GRPO for training J1.** In Table 6, we study the effect of KL Penalty and Entropy Bonus in GRPO when training a Pairwise J1-Llama-8B model. In our experiments, we find that more exploration generally leads to some degradation in performance.

Pointwise-J1 Variants	PPE	PPE	Overall	MMLU-Pro	PPE Correctness			
	Overall	Preference			MATH	GPQA	MBPP-Plus	IFEval
J1-Llama-8B - Greedy	54.5	54.9	53.8	57.0	63.2	52.1	49.8	46.9
J1-Llama-8B - Sampling	55.8	55.7	55.8	58.4	66.5	52.4	52.4	49.8
J1-Llama-8B (Mean-Score@32)	60.4	60.1	60.7	62.0	71.6	56.6	60.6	52.8
J1-Llama-70B - Greedy	57.8	50.6	65.0	73.4	77.4	58.9	60.7	54.6
J1-Llama-70B - Sampling	57.9	50.8	64.9	74.0	79.7	58.2	55.7	59.7
J1-Llama-70B (Mean-Score@32)	70.0	65.1	74.8	81.2	87.6	67.3	81.9	70.8

**Table 7** Test-time scaling of Pointwise-J1 models. Judgments are made based on the average scores of the responses.

Models	#Pref. Pairs	Overall	Chat	Chat-Hard	Safety	Reasoning
<i>Open and Closed LLM-as-a-Judge</i>						
Llama3.1-8B-Instruct <sup>†</sup>	—	69.5	92.7	46.1	64.4	74.7
Llama3.3-70B-Instruct	—	85.4	96.9	77.4	77.6	89.6
Llama3.1-405B-Instruct <sup>†</sup>	—	84.1	97.2	74.6	77.6	87.1
Claude-3.5-Sonnet <sup>†</sup>	—	84.2	96.4	74.0	81.6	84.7
GPT-4o <sup>†</sup>	—	86.7	96.1	76.1	88.1	86.6
Gemini-1.5-Pro-0514 <sup>†</sup>	—	88.2	92.3	80.6	87.9	92.0
OpenAI-o1-mini	—	87.1	94.4	78.7	80.9	94.2
DeepSeek-R1	—	90.6	95.3	83.6	86.4	97.4
<i>SOTA Generative Reward Models</i>						
Skywork-Critic-Llama-3.1-70B <sup>†</sup> (Shiwen et al., 2024)	80K	93.3	96.6	87.9	93.1	95.5
DeepSeek-GRM-27B <sup>†</sup> (Liu et al., 2025b)	237K	86.0	94.1	78.3	88.0	83.8
DeepSeek-GRM-27B (MetaRM voting@32) <sup>†</sup> (Liu et al., 2025b)	237K	90.4	95.3	85.7	89.5	91.0
EvalPlanner-Llama-8B	22K	83.0	85.5	84.0	83.4	79.3
EvalPlanner-Llama-70B	22K	93.8	97.7	89.5	91.7	96.1
<b>J1-Llama-8B</b>	22K	85.7	92.9	80.3	85.6	83.9
<b>J1-Llama-70B</b>	22K	93.3	96.1	90.1	91.9	94.9

**Table 8** Results on RewardBench comparing our Pairwise-J1 models with other state-of-the-art models. <sup>†</sup>: Results taken from either RewardBench leaderboard or the corresponding paper.