Think-J: Learning to Think for Generative LLM-as-a-Judge

Hui Huang[1]*, Yancheng He[1]*, Hongli Zhou[1]*, Rui Zhang[1], Wei Liu[1], Weixun Wang[1], Wenbo Su[1], Bo Zheng[1], Jiaheng Liu[2]✉
[1]Alibaba Group, China [2]Nanjing University, China
hh456524@taobao.com, liujiaheng@nju.edu.cn

Abstract

LLM-as-a-Judge refers to the automatic modeling of preferences for responses generated by Large Language Models (LLMs), which is of significant importance for both LLM evaluation and reward modeling. Although generative LLMs have made substantial progress in various tasks, their performance as LLM-Judge still falls short of expectations. In this work, we propose Think-J, which improves generative LLM-as-a-Judge by learning how to think. We first utilized a small amount of curated data to develop the model with initial judgment thinking capabilities. Subsequently, we optimize the judgment thinking traces based on reinforcement learning (RL). We propose two methods for judgment thinking optimization, based on offline and online RL, respectively. The offline method requires training a critic model to construct positive and negative examples for learning. The online method defines rule-based reward as feedback for optimization. Experimental results showed that our approach can significantly enhance the evaluation capability of generative LLM-Judge, surpassing both generative and classifier-based LLM-Judge without requiring extra human annotations[1].

1 Introduction

As the capabilities of generative LLMs continue to advance, accurately evaluating the response quality has emerged as a crucial challenge. This is not only vital for more efficient model development and comparison but also essential in the context of Reinforcement Learning from Human Feedback (RLHF), which relies on precise preference modeling as guidance (Wang et al., 2024a). However, traditional evaluation methods for generative models, such as BLEU (Papineni et al., 2002), are based on predefined reference answers, which are often unavailable in open-ended scenarios.

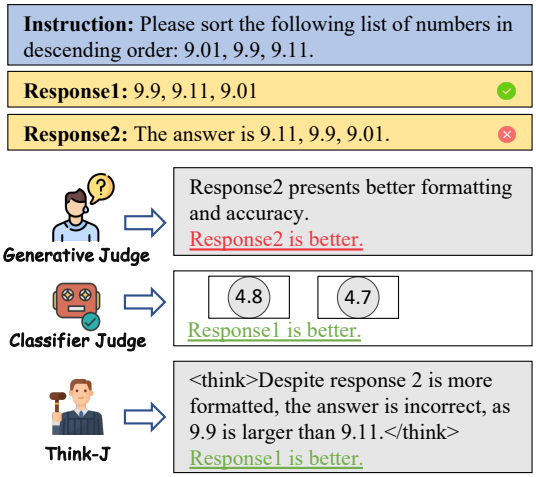

Figure 1: Comparison of different judge models. Our proposed Think-J takes into account both accuracy and interpretability based on thinking optimization.

Some studies have proposed LLM-as-a-Judge (Zheng et al., 2023), which leverages the generative capabilities of LLMs for evaluating response quality. These work either directly leverage proprietary LLMs or fine-tune a smaller judge based on preference data (Gu et al., 2024). However, the accuracy of their judgments remains unsatisfactory as revealed by recent benchmarks (Lambert et al., 2024). Other research has suggested fine-tuning a classifier based on preference data (Liu et al., 2024a). While this method can achieve higher judgment accuracy, it lacks interpretability due to its scalar output, and the performance is highly dependent on the data quality (Wang et al., 2024b).

Inspired by recent reasoning models such as o1 (Jaech et al., 2024) and Deepseek-R1 (Guo et al., 2025), in this work, we propose Thinking-enhanced Generative Judge (Think-J). Think-J aims to train a better generative judge by optimizing the model's judgment thinking capabilities. Specifically, our method consists of two steps:

1) Judgment Thinking Initialization. We care-

* The first three authors contributed equally.
[1]Codes and data are openly available at https://github.com/HuihuiChyan/Think-J

fully curated 707 samples from preference data, considering various aspects such as accuracy, difficulty, and diversity. After that, thinking trace is annotated by proprietary models to initialize the thinking capabilities of the judge.

2) Judgment Thinking Optimization. Due to the lack of high-quality critique annotation in preference datasets, we opt to optimize judgment thinking ability based on reinforcement learning (RL). Specifically, we adopted two methods: a) Critic-guide Offline Learning, leveraging an additional critic model to generate corresponding thinking traces based on provided judgment results, thus constructing positive and negative examples for offline RL. b) Rule-based Online Learning, defining rule-based rewards based on the correctness of judgment results, thus optimizing the thinking trace by online RL (Shao et al., 2024).

We conducted experiments on three open-source models, and results showed that our proposed Think-J significantly outperformed existing LLM-judges with only limited training data. We also verify the effectiveness of our method compared with generative and classifier-based preference modeling methods. Our contributions are as follows:

1. We propose to stimulate the judgment thinking ability of generative models with a small amount of carefully curated data.
2. We propose to optimize the judgment thinking ability of generative models with the correctness of judgment results as feedback.
3. Our proposed Think-J significantly outperforms previous generative and classifier-based LLM-judges for preference modeling.

2 Related Work

After the emergence of LLMs, numerous efforts have been made to design a more effective method for LLM evaluation (Chang et al., 2023). One of the most scalable and effective method is LLM-as-a-Judge (Li et al., 2023b; Zheng et al., 2023), namely utilizing proprietary LLMs, especially GPT4 (Achiam et al., 2023), to evaluate the LLM's response. For example, AlpacaEval (Li et al., 2023b) used the win rate compared with baseline response determined by GPT-4 as the evaluation result. MT-Bench (Zheng et al., 2023) automatically scored the model's answers using GPT-4 as the results. The GPT-4-based evaluator is proven to presents comparable or even better consistency and accuracy compared with human.

However, relying on external API for evaluation may introduce consideration about privacy leakage, and the opacity of API models also challenges the evaluation reproducibility. Therefore, follow-up works suggest fine-tuning language models locally for evaluations, including JudgeLM (Zhu et al., 2023b), Auto-J (Li et al., 2023a), Prometheus (Kim et al., 2023), Prometheus-2 (Zhu et al., 2023a), OffsetBias (Park et al., 2024), etc. These work typically construct preference data with judgment annotations and then finetune open-sourced LLMs to generate the judgment. Despite these fine-tuned judge models all achieve comparable accuracy with proprietary models, the evaluation is mostly conducted on the in-domain testsets, and these works are verified with a low scalability on more general benchmarks (Huang et al., 2024).

Another group of work fine-tunes a classifier on preference data based on the Bradley-Terry model, which is more commonly used on reward modeling. This approach is simple yet effective, as demonstrated on RewardBench (Lambert et al., 2024) where most top-performing models are trained in a classification style (Liu et al., 2024a). However, this method does not fully leverage the generative capabilities of LLMs, and is unable to provide rationales for its judgments, which is crucial for scalable evaluation. While recent work has begun to leverage the generative abilities of LLMs to combine critiques for scalar reward prediction (Ke et al., 2024; Ye et al., 2024), these critiques are often distilled from stronger proprietary models and hardly influence the final prediction (Liu et al., 2025b). Effectively integrating the generative abilities of LLMs into evaluation remains an open challenge (Chen et al., 2025a; Whitehouse et al., 2025; Chen et al., 2025b; Wang et al., 2025a).

3 Approach

3.1 Judgment Thinking Initialization

Recent studies have shown that LLMs inherently possess long chain-of-thought (CoT) reasoning capabilities, which can be activated with a small amount of data (Muennighoff et al., 2025; Ye et al., 2025). In this work, we also curate high-quality preference data, **LIMJ707**, to initialize the thinking capability of the judge model. LIMJ707 are selected based on three principles:

- **Accuracy**: The judgment annotation should be correct. We leverage the high-quality preference

data Skywork-Preference-v0.2[2], which has been carefully validated to ensure accuracy.

- **Difficulty**: The sample should be sufficiently challenging. We apply the judge models to perform judgment for the sample three times, and select those samples where at least one judgment is failed, as these samples are likely more difficult and reflect the insufficiency of the judge.
- **Diversity**: The instruction should encompass various types to enhance judgment thinking capabilities in different aspects. We represent the instructions with an embedding model and then merge samples that are too close to each other[3].

The data statistics during constructing LIMJ707 are shown in Table 1. Based on LIMJ707, we construct judgment thinking trace by Deepseek-R1. After that, the annotated samples are used to initialize the model with judgment thinking capability by Supervised Fine-tuning (Ouyang et al., 2022).

Skywork-Reward-Preference-v0.2	Data Num
- initially selected	10K
- filtered by difficulty	1496
- filtered by diversity	870
- filtered by accuracy	707

Table 1: Data statistics during constructing LIMJ707.

Despite its superior reasoning capability, the thinking trace generated by R1 are excessively long and could possibly cause difficulty for further optimization and increase computational overhead. Specifically, for our case, we hypothesize that general LLM judgment does not necessarily require a detailed, step-by-step reasoning process. On the contrary, identifying one key factor is sufficient to make correct judgment in most cases.

Therefore, we introduce **Trace Clipping** to enhance the effectiveness of thinking initialization[4]. Specifically, we observed that the CoTs generated by R1 are composed of two parts: a lengthy reasoning process, and an explanation which summarizes the reasoning process. Therefore, we remove the first part and use the second part as the output trace. This not only facilitates subsequent optimization but also enhances the judgment readability.

3.2 Judgment Thinking Optimization

To further enhance the alignment between the judge and human preference, the initialized thinking trace should be further optimized on preference data. However, preference data typically only includes binary labels without thinking trace annotations. Therefore, we propose judgment thinking optimization based on reinforcement learning (RL). Specifically, we propose two optimization approaches based on offline and online RL methods, respectively, as detailed in the following sections.

3.2.1 Critic-guided Offline Learning

Offline RL methods represented by Direct Preference Optimization (DPO) (Rafailov et al., 2023) has been widely applied to LLM pipelines due to their efficiency and simplicity (Grattafiori et al., 2024). In this work, we also aim to optimize the thinking ability based on offline learning.

Due to the lack of golden thinking annotation in preference dataset, we propose to train an additional critic model[5], to help to construct training samples. Both the critic and the judge models are trained on the same data (i.e., LIMJ707), with the following distinctions:

- **Judge Model:** Given instruction-responses, it generates the thinking trace and judgment result.
- **Critic Model:** Given instruction-responses and judgment result, it generates the thinking trace.

Based on the two models, we can perform thinking optimization with the following steps:

1. First, leverage the judge to evaluate the input to generate the thinking traces and results.
2. If the result is correct, use the critic to generate an incorrect trace as the negative sample. Conversely, if the result is incorrect, use the critic to generate a correct trace as the positive sample.
3. Based on the positive and negative samples, optimize the judgment thinking ability with offline learning objective.
4. These steps can be iterated to continuously enhance the judgment thinking capability.

We adopt a combination of SFT and DPO as our training objective in this step:

$$\mathcal{L}_{\text{offline}}(\pi_\theta; \mathcal{D}) = - \mathbb{E}_{x\sim\mathcal{D},(y_w,y_l)\sim\pi_\theta(y|x)} \left[\log \pi_\theta(y_w \mid x) + \log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{ref}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{ref}(y_l \mid x)} \right)\right] \quad (1)$$

[2] huggingface.co/datasets/Skywork/Skywork-Reward-Preference-80K-v0.2

[3] For more details please refer to Appendix B.1.

[4] An example is presented in Appendix B.1.

[5] Notice the critic model here differs from the critic model in traditional RL which is used for advantage estimation.

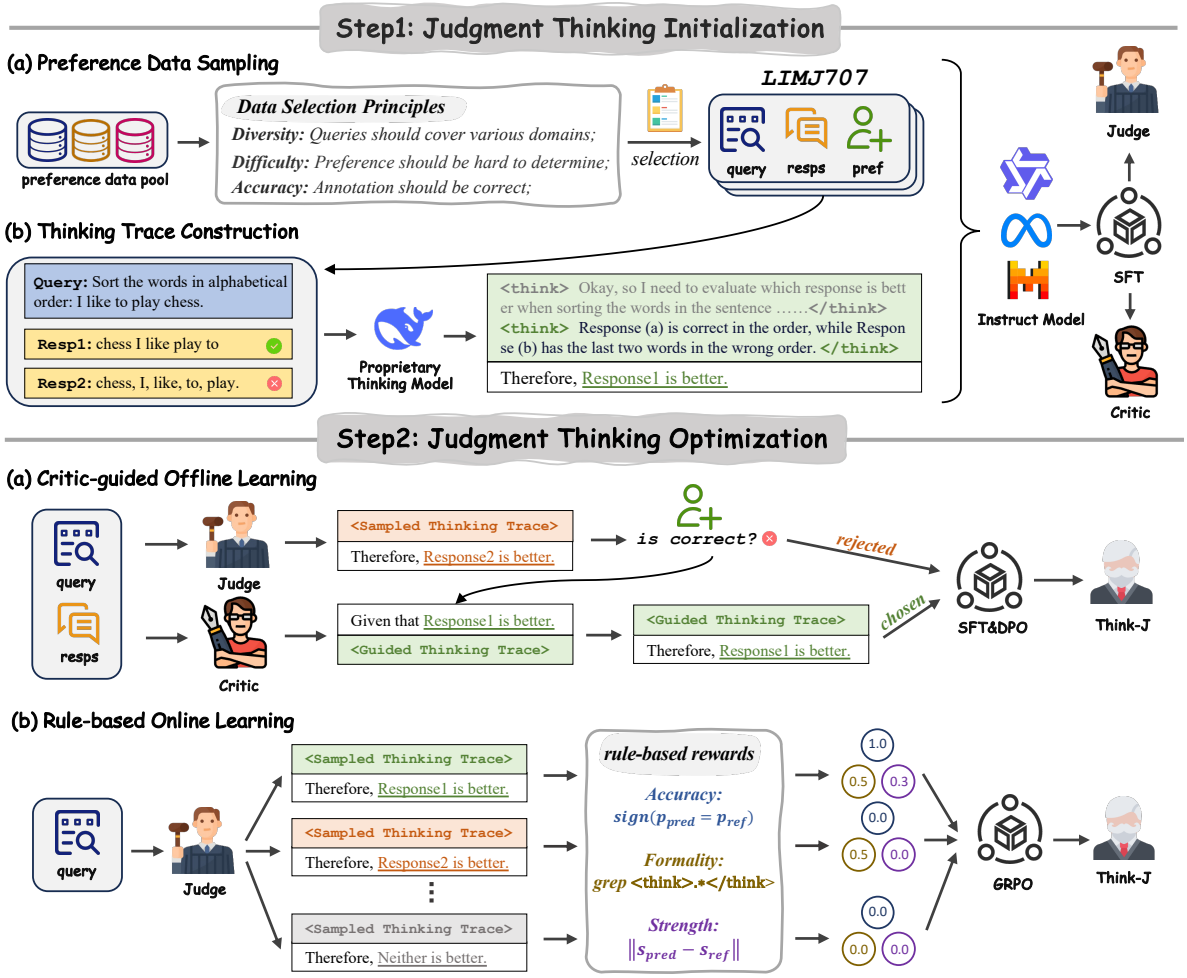

Figure 2: The illustration of our proposed framework. We begin by constructing high-quality judgment thinking traces using curated principles and proprietary thinking models. Based on this data, we initialize a judge model and a critic model, both equipped with judgment thinking capability. After that, we optimize the capability of the judge model through two methods: Critic-guided Offline Learning and Rule-based Online Learning, resulting in Think-J.

where π_θ and π_{ref} represent the policy model and the reference model, and y_w and y_l denotes the positive and negative judgment traces, respectively.

With the help of critic model, samples with thinking annotation are constructed based on the correctness of judgment result. Therefore, the judge model will be enhanced to generate more accurate thinking, leading to more accurate judgment.

3.2.2 Rule-based Online Learning

The recent success of R1-style methods have demonstrated the effectiveness of online RL using discrete, rule-based rewards (Shao et al., 2024). In this work, we also apply online rule-based RL approach to optimize the judgment thinking capability. More specifically, we mainly utilize the GRPO algorithm, whose optimization objective is:

$$J_{\text{online}}(\pi_\theta; \mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(y|x)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)} A_i, \text{clip}\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}}) \right] \tag{2}$$

where G is group size, and A_i is advantage. The reward function is designed as follows[6]:

$$r_{\text{accuracy}} = \begin{cases} 1, & \text{if judgement} = \text{label} \\ 0, & \text{if judgement} \neq \text{label} \end{cases} \tag{3}$$

$$r_{\text{format}} = \begin{cases} 0, & \text{if format is right} \\ -0.5, & \text{if format is wrong} \end{cases} \tag{4}$$

$$r_{\text{strength}} = ||\text{s}_{\text{pred}} - \text{s}_{\text{golden}}||, s \in \{1, 2, 3\} \tag{5}$$

$$r_{\text{final}} = \alpha \cdot r_{\text{accuracy}} + \beta \cdot r_{\text{format}} + \gamma \cdot r_{\text{strength}} \tag{6}$$

[6]We do not include a length penalty in rewards to encourage longer thinking, as we have observed that longer thinking does not necessarily lead to better accuracy in our case.

Notice we incorporate a reward for assessing preference strength, which is defined as the degree to which the judge favors one response over another[7]. The final judgment is also adjusted as:

`<think> {thinking trace} </think>`
`Therefore, Response (a) is better, and the preference strength is [[2]].`

Preference strength helps to perceive the relative quality of response pairs, without uniformly providing the same reward for different pairs despite the quality gap. We employ a comparatively simple scale of reward scores of reward, as during actual training, we observed that the model tends to manipulate scores towards extreme values [8].

While annotating an absolute score for a given response is challenging, annotating preference strength is relatively easy, making strength annotation more readily accessible (Wang et al., 2024d, 2025b). On the other hand, if absolute score annotation is absent in the preference data, we can firstly fine-tune a BT-classifier based on the data, then leverage the classifier to assess the relative strength between chosen and rejected responses, as explained in Wang et al. (2024b).

4 Experiments

4.1 Set-up

We mainly conducted experiments on two popular open-sourced models with their instruction version: Qwen-2.5 (Qwen et al., 2025) and Llama-3 (Grattafiori et al., 2024).

The optimization is conducted on two datasets: HelpSteer2-Preference[12] and HH-RLHF[13]. Notice while there are larger preference datasets available, we did not include them in our training, as our primary objective was to explore the most effective method for training LLM judges.

We compare Think-J with the following generative judgment approaches:

- **Direct Prompt** Leverage the LLM to directly generate judgment.
- **SFT (w/o CoT)** Train a generative model based on Supervised Fine-tuning to perform judgment without generating CoTs.
- **SFT (w/ CoT)** Train a generative model on correct judgment thinking traces and results, which can also be deemed as Rejection Sampling.

We also compare Think-J with the following classifier-based approaches:

- **BT Classifier** Feed the instruction and responses into the model and added a classification head, training it according to Bradley-Terry model.
- **CLoud** (Ke et al., 2024) First train the model to generate critiques, and then leverage the critiques as additional input to improve the BT Classifier.
- **SynRM** (Ye et al., 2024) Leverage critiques generated by a proprietary model as additional input to improve the BT Classifier. We use Deepseek-R1 to generate the critiques.

To further showcase Think-J's evaluation capabilities, we also compared its 32B version against leading proprietary judge models, including closed-source options like Claude 3.5 Sonnet, GPT-4o, and Gemini 1.5 Pro, as well as open-source fine-tuned judges such as JudgeLM, Prometheus, and CompassJudger. These models are widely employed as LLM-as-a-Judge across diverse tasks.

We mainly perform evaluation on RewardBench (Lambert et al., 2024), which is the most popular benchmark for evaluating preference modeling ability. We also evaluate our model on RMBench (Liu et al., 2025a) and Auto-J-test (Li et al., 2023a).

We report the best results achieved by either offline or online learning in the experiment results by default. We also mixed LIMJ707 into all training sets to ensure a fair comparison.

4.2 Main Experiment

As demonstrated in Table 2, Think-J achieved the best performance across all benchmarks, surpassing both close-sourced and fine-tuned judges. Notably, our method required only 9832 training samples, but still achieve marginal improvement on 32B-sized models. This underscores the effectiveness of Think-J, which leverages thinking optimization with the correctness of judgment as feedback to enhance preference modeling capability.

Furthermore, as Table 3 illustrates, starting from the same base model and training data, our proposed method consistently outperforms other approaches for fine-tuning LLM judges. This demonstrates the effectiveness of our judgment thinking optimization. In contrast, both naive and fine-tuned generative methods yield inferior results. Notably, the rejection sampling method, despite being

[7] For example, a strength of 1 means the chosen response is only slightly better than the rejected, while a strength of 3 means the chosen is much better than the rejected.

[8] For more details please refer to Appendix A.1.

[12] huggingface.co/datasets/nvidia/HelpSteer2

[13] huggingface.co/datasets/Anthropic/hh-rlhf

Model	Sample Num	RewardBench					RMBench	Think-J
		Chat	Hard	Safety	Reason	Overall	Overall	Agreement
Claude-3-5-Sonnet-20240620[9]	—	96.4	74.0	81.6	84.7	84.2	68.9	70.7
Qwen-2.5-32B-Instruct	—	96.2	74.0	88.7	86.9	86.5	68.3	59.6
GPT-4o-2024-08-06[10]	—	96.1	76.1	88.1	86.6	86.7	68.8	69.8
Gemini-1.5-Pro-0514[11]	—	92.3	80.6	87.9	92.0	88.2	74.4	68.1
JudgeLM-33B (Zhu et al., 2023b)	100K	90.1	51.0	85.7	39.7	66.6	49.6	45.3
Prometheus-7b-v2.0 (Zhu et al., 2023a)	40K	83.9	49.2	72.8	72.0	69.5	52.4	63.1
Prometheus-8x7b-v2.0 (Zhu et al., 2023a)	40K	93.0	47.1	80.5	77.4	74.5	57.4	68.5
Llama-3-OffsetBias-8B (Park et al., 2024)	276K	92.5	80.3	86.8	76.4	84.0	66.0	68.7
CompassJudger-32B (Cao et al., 2024)	2041K	97.4	65.6	85.1	87.1	83.8	69.4	**80.7**
STE-Llama3.1-70B (Wang et al., 2024c)	20K	96.9	85.1	89.6	88.4	90.0	65.3	72.0
SynRM-Command-R-35B (Ye et al., 2024)	5K	97.5	76.8	88.5	86.3	87.3	—	—
CLoud-Llama3-70B (Ke et al., 2024)	350K	98.0	75.6	87.6	89.0	87.6	—	—
Think-J-Qwen-2.5-32B	9.8K	96.7	83.2	90.1	92.0	**90.5**	**79.8**	**75.8**

Table 2: Experiment results of different LLM judges on RewardBench, RMBench and Auto-J.

Model	Method	HH-RLHF RewardBench					HelpSteer2-Preference RewardBench				
		Chat	Hard	Safety	Reason	Overall	Chat	Hard	Safety	Reason	Overall
Llama3-8B -Instruct	Direct Prompt (w/o CoT)	90.4	44.7	76.5	63.4	68.8	90.4	44.7	76.5	63.4	68.8
	Direct Prompt (w/ CoT)	84.6	40.9	50.8	58.8	58.8	84.6	40.9	50.8	58.8	58.8
	SFT on LIMJ707	91.8	67.1	83.0	64.2	76.5	91.8	67.1	83.0	64.2	76.5
	SFT (w/o CoT)	88.1	50.6	81.0	69.1	72.2	88.0	41.1	41.5	47.8	54.6
	SFT (w/ CoT)	78.8	67.0	81.0	62.1	72.2	84.5	71.3	80.8	62.6	74.8
	BT Classifier	81.8	72.6	79.3	85.2	79.7	88.6	73.9	81.2	91.8	83.9
	CLoud	87.4	69.3	87.3	77.6	80.4	93.3	74.8	80.8	73.7	80.6
	SynRM	87.2	74.3	81.1	80.2	80.7	91.9	68.4	80.4	87.9	82.2
	Think-J	92.2	71.7	84.8	75.6	**81.1**	93.9	74.3	90.3	77.8	**84.1**
Qwen2.5-7B -Instruct	Direct Prompt (w/o CoT)	94.4	56.9	81.0	77.3	77.4	94.4	56.9	81.0	77.3	77.4
	Direct Prompt (w/ CoT)	93.0	58.1	81.2	77.8	77.5	93.0	58.1	81.2	77.8	77.5
	SFT on LIMJ707	89.5	75.2	80.7	74.5	80.0	89.5	75.2	80.7	74.5	80.0
	SFT (w/o CoT)	89.8	74.3	82.5	61.3	77.0	95.0	75.9	87.2	75.8	83.5
	SFT (w/ CoT)	94.1	74.0	86.0	64.7	79.7	88.7	69.7	84.5	76.1	79.8
	BT Classifier	82.4	69.7	87.0	76.9	79.0	95.0	67.8	85.0	61.4	77.3
	CLoud	85.2	79.2	86.6	57.9	77.2	94.7	73.3	82.3	59.0	77.3
	SynRM	90.5	65.1	77.8	70.6	76.0	96.7	61.4	82.8	61.4	75.6
	Think-J	94.4	70.2	83.8	79.8	**82.0**	96.1	78.6	85.9	80.4	**85.3**

Table 3: Experiment results of different LLM judge training methods on RewardBench.

trained on the same data constructed by the critic, also underperforms. This underscores the critical role of learning from negative samples when modeling human preference (Liu et al., 2024b).

On the other hand, the classifier-based method achieves relatively higher results but can only produce numerical outputs that lack interpretability. Additionally, the reasoning-enhanced classifiers, including CLoud and SynRM, performs worse than expected. This suggests that combining generative CoT into classifier may introduce noise rather than useful information for classification.

5 Analysis

5.1 Less is More for Thinking Initialization

We compared the performance of different data source for judgment thinking initialization in Figure 3. Our findings indicate that a small amount of thinking trace annotated with Deepseek-R1 can significantly enhance the model's judgment capabilities. In contrast, using thinking trace annotated with Deepseek-V3, or removing the trace from the data would results in a substantial decline in performance. This highlights the importance of high-quality trace for thinking initialization.

We also compare the impact of different data selection strategies in Table 5. The results show that data quality is crucial for effective model initialization. For instance, a model initialized with chatbot-arena[14], which contains relatively noisy data, achieves minimal improvement. Conversely, selecting the longest traces proves detrimental, as the longest traces are often code or math-related, which can negatively impact the data diversity.

We further investigated the impact of differ-

[14] huggingface.co/datasets/lmsys/chatbot_arena_conversations

Method	Llama-3-8B-Instruct RewardBench					Qwen-2.5-7B-Instruct RewardBench				
	Chat	Hard	Safety	Reason	Overall	Chat	Hard	Safety	Reason	Overall
baseline	90.4	44.7	76.5	63.4	68.8	94.4	56.9	81.0	77.3	77.4
offline (Sampling)	92.9	75.0	86.9	70.1	81.2	92.9	72.4	85.3	76.2	81.7
offline (Critic-guided)	95.0	73.3	87.4	77.2	83.2	94.8	74.2	83.5	80.5	83.3
offline (w/o SFT)	95.1	63.8	88.3	77.6	81.2	95.7	73.8	86.0	74.6	82.5
online (PPO)	70.4	75.3	77.2	70.0	73.2	88.7	69.7	84.5	76.1	79.8
online (Reinforce++)	93.9	74.3	90.3	77.8	84.1	94.7	70.6	90.4	82.3	84.5
online (GRPO)	93.9	74.3	90.3	77.8	**84.1**	96.1	78.6	85.9	80.4	**85.3**

Table 4: Experiment results of different RL methods.

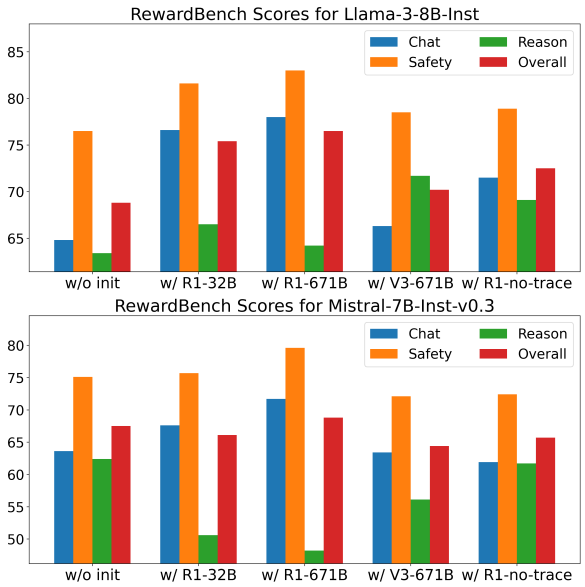

Figure 3: Experiment results of different data source for judgment thinking initialization.

Method	RewardBench			
	Chat	Safety	Reason	Overall
Llama-3-8B-Inst	64.8	76.5	63.4	68.8
Init with 1000 samples on chatbot-arena				
random-sampled	66.0	74.3	64.4	68.3
Init with 1000 samples on skywork-preference-v3				
random-sampled	75.1	82.0	66.4	75.4
longest	77.2	79.3	61.8	74.2
Llama3-failed	76.4	85.0	66.5	**76.1**

Table 5: Experiment results of different data selection strategies for judgment thinking initialization.

ent initialization strategies on the subsequent optimization process, as shown in Table 6. The results indicate that even without thinking initialization, RL-based methods can still achieve substantial improvements, validating their effectiveness. Moreover, initializing the model with a few R1-annotated traces leads to a more structured and effective reasoning pattern, resulting in further enhancements in performance. In contrast, when initialization is performed without trace clipping, the optimization effect deteriorates, particularly for online RL. This finding supports our earlier hypothesis that for general LLM judgment, longer thinking do not necessarily correlate with better accuracy.

Init	RL	RewardBench			
		Chat	Safety	Reason	Overall
Llama-3-8B-Inst		64.8	76.5	63.4	68.8
no init	offline	80.3	73.2	79.4	79.1
	online	80.8	86.8	71.8	80.8
w/o clip	offline	82.6	88.1	70.2	81.4
	online	79.6	85.1	67.5	78.7
w/ clip	offline	82.8	87.4	77.2	83.2
	online	82.9	90.3	77.8	**84.1**

Table 6: Experiment results of different initialization strategies for RL optimization on HelpSteer2.

Method	RewardBench			
	Chat	Safety	Reason	Overall
Llama-3-8B-Inst	64.8	76.5	63.4	68.8
Offline learning on Helpsteer2				
iteration 1	81.8	88.3	77.8	83.2
iteration 2	82.5	87.2	74.8	82.5
iteration 3	82.8	87.4	77.2	**83.2**
Offline learning on HH-RLHF				
iteration 1	81.2	83.2	67.0	77.3
iteration 2	81.1	84.6	70.5	78.4
iteration 3	78.3	83.1	72.6	**78.9**

Table 7: Experiment results of iterative offline learning.

5.2 Best Practice for Thinking Optimization

With new RL algorithms continue to emerge in LLM training (Zhang et al., 2025), in this section, we compare different RL methods and strategies for judgment thinking optimization.

As shown in Table 4, for offline RL, constructing traces based on sampling (with a large beam size) would lead to performance degradation. This verifies the critical role of critic model for constructing both positive and negative examples. On the other hand, for online RL methods, we find

Setting	Qwen2.5-32B-Instruct RewardBench					Meta-Llama-3-8B-Instruct RewardBench				
	Chat	Hard	Safety	Reason	Overall	Chat	Hard	Safety	Reason	Overall
β=1.0	96.7	83.2	90.1	92.0	**90.5**	93.9	74.3	90.3	77.8	**84.1**
β=0.5	96.9	85.8	91.2	87.4	90.3	95.0	73.3	87.4	77.2	83.2
β=0.0	92.9	71.3	86.3	67.3	79.4	96.2	65.5	88.0	71.5	80.3
γ=0.5	91.2	75.2	71.2	61.5	74.8	82.7	65.5	80.4	66.4	73.7
γ=0.2	96.7	83.2	90.1	92.0	**90.5**	96.5	66.8	88.7	73.2	81.3
γ=0.0	96.0	83.7	90.5	87.7	89.5	93.9	74.3	90.3	77.8	**84.1**

Table 8: Experiment results of different reward function settings on Helpsteer2-Preference.

that PPO (Schulman et al., 2017) significantly underperforms compared to GRPO and Reinforce++ (Hu et al., 2025). This discrepancy suggests that incorporating a value model for thinking optimization would decrease training stability. We argue that natural language generation (NLG) tasks is distinct from the sequential decision-making tasks in traditional RL. Therefore, the introduction of an additional value model is not only unnecessary but may also hinder training efficiency.

Finally, perform offline learning in an iterative manner, as shown in Table 7, can achieve further improvement. However, this will result in a more complex and cumbersome training pipeline.

5.3 Reward Design for Online Learning

In this section, we aimi to analyze the contribution of different components of the reward function as defined in 6. We fixed α as 1.0 and varied the weights β, and γ, with the results detailed in Table 8. Our findings indicate that the formatting reward r_{format} is crucial for both models, as its removal leads to significant performance degradation. Conversely, the influence of the strength reward r_{strength} differs between the models. For Llama3-8B, removing r_{strength} results in a performance improvement, while it offers a slight enhancement for the more capable Qwen2.5-32B. We hypothesize that Qwen2.5-32B's stronger inherent abilities allow it to effectively learn the correlation between preference strength and judgment. In contrast, for the weaker model Llama-3-8B, this additional learning objective may introduce confusion.

5.4 Thinking Makes the Judge More Robust

In real applications, it is common that there exists noise in the training set, or there is a distributional difference between the training and test sets. In such cases, Think-J demonstrates superior robustness as a result of its judgment thinking ability.

To verify this, we adopted the approach from Wang et al. (2024b) and divided HH-RLHF into four groups with different data quality[15]. We then trained judges on the data based on classifier-judge or Think-J. As shown in Figure 4, for the two groups with higher data quality, classifier-based judge achieve comparable or even better performance. However, for the groups with lower quality, the accuracy of classifier-based methods drops significantly, even falling below random guessing. In contrast, Think-J maintains relative stability, verifying its robustness to varied data quality.

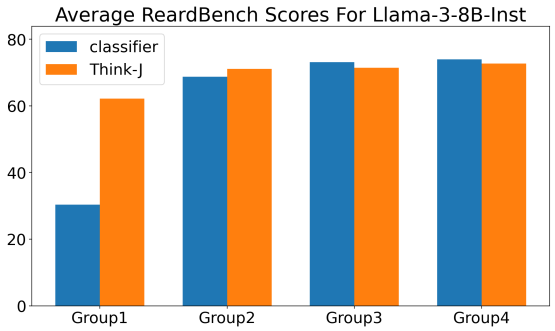

Figure 4: Comparison of different methods on data groups with different quality. Group 1 is with lower quality and Group 4 is with higher quality.

6 Conclusion

In this paper, we propose Think-J to enhance generative LLM-as-Judge with judgment thinking optimization. Experiment results verify the effectiveness of Think-J compared with both classifier-based and generative LLM judges.

With the increasing popularity of RL-based test-time scaling methods, it is crucial to develop a reliable and stable feedback system that aligns well with real-world human preferences. In future, we will continue to explore generative judges for more accurate preference modeling and optimization.

[15]Data quality is indicated by the difference in scores assigned to response pairs by the reward model. For more details, please refer to Wang et al. (2024b).

Limitations

Our work still has several limitations: 1) Due to time and resource constraints, we only validated our method on 32B-sized models with a relatively small dataset. Fine-tuning larger models with more comprehensive preference data warrants future investigation. 2) While LLM-as-a-Judge and Reward Modeling are two inter-correlated tasks, the lack of absolute scoring data currently prevents us from extending our pairwise-trained LLM-Judge to a pointwise setting, hindering it for direct RL guidance. Acquiring such data is a key direction for future work. 3) The increasing prevalence of thinking-enhanced models like Qwen-3 necessitates investigating the continued effectiveness of our method on these architectures. Scaling our approach to reasoning-enhanced models would further strengthen our findings.

Ethical Considerations

Our research aims to model human preference, ultimately making LLM outputs less harmful and more helpful. While the judge model might occasionally produce relatively harmful outputs during experimentation, its core purpose is to provide unbiased judgment towards helpfulness and harmlessness, thereby reducing the occurrence of such outputs. Furthermore, our method leverages existing datasets rather than creating new ones. Based on these points, we believe our work is ethical.

References

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

Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. 2024. Compassjudger-1: All-in-one judge model helps model evaluation and evolution. *Preprint*, arXiv:2410.16256.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *Preprint*, arXiv:2307.03109.

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

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. 2025b. Rm-r1: Reward modeling as reasoning. *Preprint*, arXiv:2505.02387.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jian Hu, Jason Klein Liu, and Wei Shen. 2025. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *Preprint*, arXiv:2501.03262.

Hui Huang, Yingqi Qu, Xingyuan Bu, Hongli Zhou, Jing Liu, Muyun Yang, Bing Xu, and Tiejun Zhao. 2024. An empirical study of llm-as-a-judge for llm evaluation: Fine-tuned judge model is not a general substitute for gpt-4. *Preprint*, arXiv:2403.02839.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Pei Ke, Bosi Wen, Andrew Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. 2024. CritiqueLLM: Towards an informative critique generation model for evaluation of large language model generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13034–13054, Bangkok, Thailand. Association for Computational Linguistics.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling. *Preprint*, arXiv:2403.13787.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023a. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*.

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

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

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2024b. Statistical rejection sampling improves preference optimization. In *The Twelfth International Conference on Learning Representations*.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2025a. RM-bench: Benchmarking reward models of language models with subtlety and style. In *The Thirteenth International Conference on Learning Representations*.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Junsoo Park, Seungyeon Jwa, Ren Meiying, Daeyoung Kim, and Sanghyuk Choi. 2024. OffsetBias: Leveraging debiased data for tuning evaluators. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1043–1067, Miami, Florida, USA. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024a. Secrets of rlhf in large language models part ii: Reward modeling. *Preprint*, arXiv:2401.06080.

Binghai Wang, Rui Zheng, Lu Chen, Zhiheng Xi, Wei Shen, Yuhao Zhou, Dong Yan, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024b. Reward modeling requires automatic adjustment based on data quality. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4041–4064, Miami, Florida, USA. Association for Computational Linguistics.

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

Yibin Wang, Zhimin Li, Yuhang Zang, Chunyu Wang, Qinglin Lu, Cheng Jin, and Jiaqi Wang. 2025a. Unified multimodal chain-of-thought reward model through reinforcement fine-tuning. *Preprint*, arXiv:2505.03318.

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024d. Helpsteer2-preference: Complementing ratings with preferences. *Preprint*, arXiv:2410.01257.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Daniel Egert, Ellie Evans, Hoo-Chang Shin, Felipe Soares, Yi Dong, and Oleksii Kuchaiev. 2025b. Dedicated feedback and edit models empower inference-time scaling for open-ended general-domain tasks. *Preprint*, arXiv:2503.04378.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *Preprint*, arXiv:2505.10320.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gallé. 2024. Improving reward models with synthetic critiques. *Preprint*, arXiv:2405.20850.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. 2025. What, how, where, and how well? a survey on test-time scaling in large language models. *Preprint*, arXiv:2503.24235.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, YeYanhan YeYanhan, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2023a. Promptbench: A unified library for evaluation of large language models. *arXiv preprint arXiv:2312.07910*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023b. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*.

A Additional Experiments

A.1 Margin-based Reward for GRPO

As we have explained in Section 3.2.2, we design the reward function as the combination of three parts: r_{accuracy}, r_{format}, r_{strength}. Referring to the work of Chen et al. (2025a), for the purpose of more fine-grained judgment prediction, we also tested the following reward function design:

$$r_{\text{margin}} = \begin{cases} -||s_{\text{resp1}} - s_{\text{resp2}}||, & \text{if judgement} = \text{label} \\ ||s_{\text{resp1}} - s_{\text{resp2}}||, & \text{if judgement} \neq \text{label} \end{cases}$$

where the judgment prediction should also be formatted as:

```
<think> {thinking trace} <think>
Therefore, the quality scores for Response (a) and Response (b) are [[30]] and
[[50]], respectively.
```

Method	Llama-3-8B-Instruct RewardBench					Qwen-2.5-7B-Instruct RewardBench				
	Chat	Hard	Safety	Reason	Average	Chat	Hard	Safety	Reason	Average
online (GRPO w/ margin)	96.5	66.8	88.7	73.2	81.3	96.1	72.3	83.4	76.1	82.0
online (GRPO w/o margin)	93.9	74.3	90.3	77.8	**84.1**	96.1	78.6	85.9	80.4	**85.3**

Table 9: Experiment results of the impact of r_{margin} on Helpsteer2-Preference.

This design enables fine-grained absolute response scoring, which is more useful for LLM preference optimization pipelines. However, as shown in Table 9, the introduction of margin-based reward results in performance degradation. By further inspection, we revealed a consistent issue of reward hacking, where the assigned scores were always driven to extreme values (e.g., 0 or 100) to maximize the reward. Despite experimenting with various formulations of r_{margin}, we were unable to effectively mitigate this problem. We believe that mitigating this reward hacking for r_{margin} necessitates preference data annotated with absolute scores, which we leave for future investigation.

A.2 Statistical Indicators of Judgment Thinking Optimization

In this section, we present the detailed statistical indicators during Judgment Thinking Optimization. The statistical indicators of offline learning, GRPO training, PPO (Schulman et al., 2017) training and Reinforce++ (Hu et al., 2025) training are presented in Figure 5, 6, 8, 7, respectively.

As the results indicate, GRPO demonstrates consistent and substantial increases in predicted rewards, a strong signal of learning to generate desirable outputs, and the critic's low and stable value function loss suggests reliable outcome evaluation. While DPO effectively aligns with preferences, and Reinforce++ and PPO show progress in reward acquisition and stability, GRPO's robust reward improvement combined with a well-functioning critic positions it as the most robust algorithm based on these indicators.

Notably, GRPO's response length exhibits a consistent gradual increase, similar to prior work of Shao et al. (2024), while other methods start to fluctuate after a certain point. This suggests GRPO effectively learns to generate more detailed reasoning, enhancing judgment accuracy.

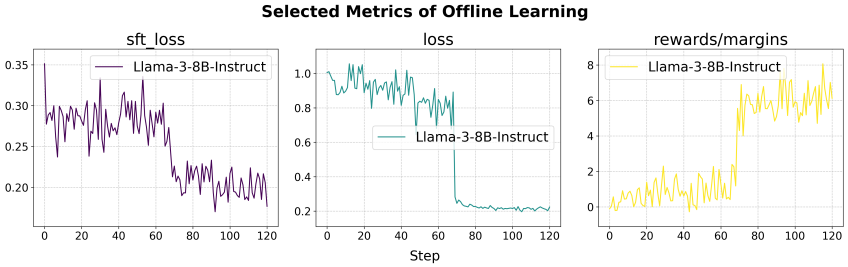

Figure 5: The variation of statistical metrics during offline learning trained on Helpsteer2-Preference.

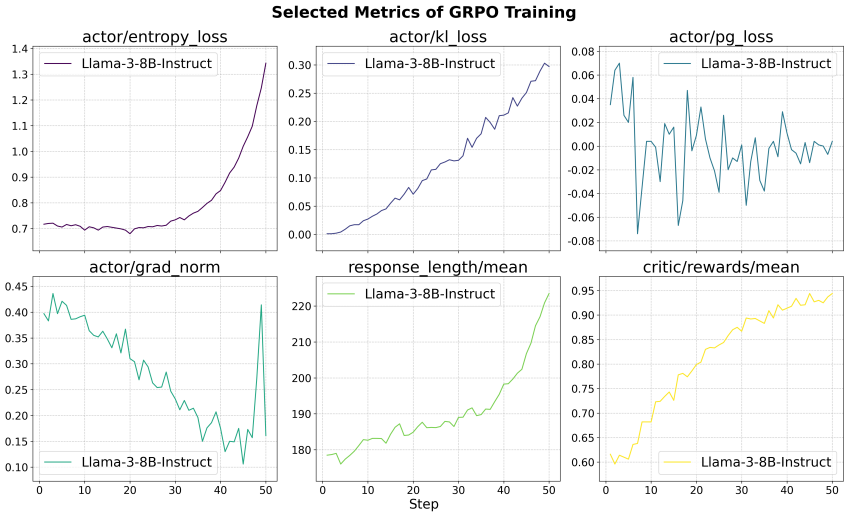

Figure 6: The variation of statistical metrics during GRPO training on Helpsteer2-Preference.

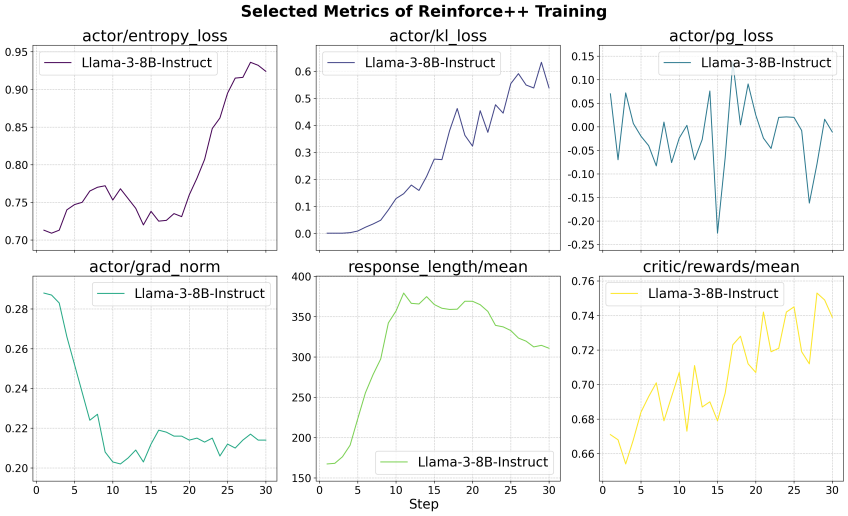

Figure 7: The variation of statistical metrics during Reinforce++ training on Helpsteer2-Preference.

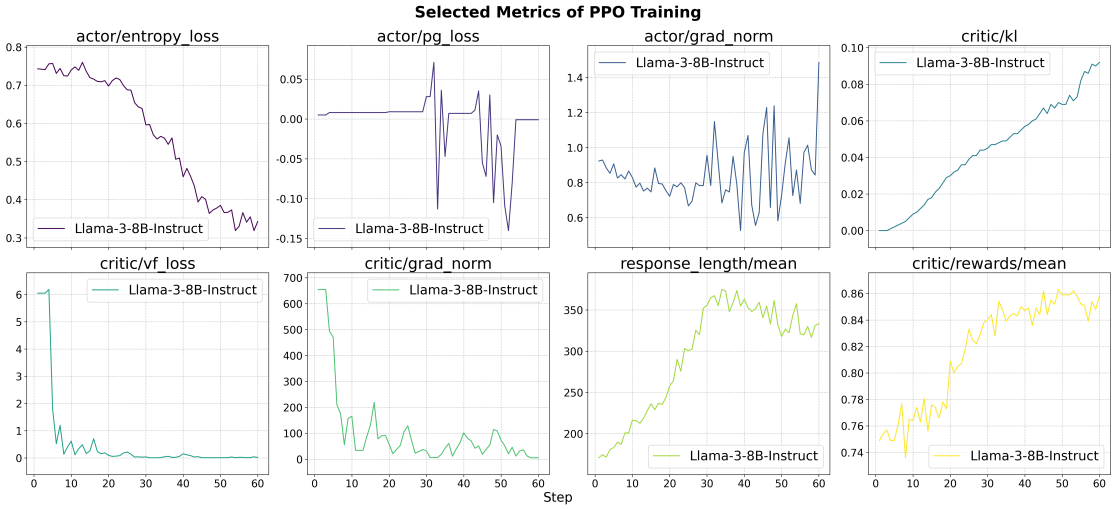

Figure 8: The variation of statistical metrics during PPO training on Helpsteer2-Preference.

B Implementation Details

B.1 Judgment Thinking Initialization

As we have explained in Section 3.1, we apply deduplication to ensure the diversity of LIMJ707. For this purpose, we leverage the algorithm defined in Algorithm 1 to create instruction data pool, where the embeddings of instructions are derived based on the model of instructor-large[16], and repetitive instructions are removed based on the similarity of instruction semantics.

We leverage the prompt templates in Figure 11 and 13 for constructing thinking trace based on proprietary thinking models. The temperature is set as 0.0 to enable more accurate judgment. We also present an example of thinking trace calibration in 10, which can ensure training efficiency and stability.

Algorithm 1 Diversity-verified Sampling

Input: Instruction Dataset D with m samples, number of samples to select n.
Output: Selected Dataset D' with n samples.

Derive the embeddings for each sample in D.
Random sample one data point x from D.
Delete x from D, add x to D'.
for $i = 1, 2, ..., n-1$ **do**
    Calculate the cosine similarity score between x and each sample from D.
    Select the least similar sample x' from D'.
    Let $x = x'$.
    Delete x from D, add x to D'.
end for

Figure 9: The sampling algorithm for LIMJ707 used for diversity verification.

Judgment Input

```
Please act as an impartial judge and evaluate the quality of the responses provided by two AI
assistants to the user question displayed below.
[The Start of User Question]
Generate a 3-word phrase that summarizes the following text A budding professional photographer
looking to make a career out of taking creative photos.
[The End of User Question]
[The Start of Response (a)]
Find creative photos.
[The End of Response (a)]
[The Start of Response (b)]
Pictures capture moments.
[The End of Response (b)]
Please provide an evaluation by first offering a brief explanation, and then end your answer
with "Therefore, Response (a) is better." or "Therefore, Response (b) is better.
```

Judgment Output with Thinking Trace Clipping

```
<think>\nOkay, let's tackle this evaluation. The user wants a 3-word phrase that summarizes the
given text about a budding professional photographer aiming to make a career from creative
photos.\n\nFirst, looking at Response (a): "Find creative photos." The original text mentions
the photographer is looking to make a career out of taking creative photos. "Find" might not
capture the essence of building a career. It sounds more like searching for existing photos
rather than creating them. However, "creative photos" does align with the text's focus on
creativity.\n\nResponse (b): "Pictures capture moments." While this is a nice phrase, it's more
generic. The original text emphasizes a career built on creativity, not just capturing moments.
"Capture moments" is a common photography phrase but doesn't highlight the professional and
creative aspects mentioned.\n\nSo, Response (a) includes "creative," which is key, but "Find"
isn't ideal. Response (b) misses the career and creativity focus. Even though neither is
perfect, (a) better addresses the creative aspect mentioned in the text.\n</think>
<think>\nThe user requested a 3-word phrase summarizing a budding photographer's career focus
on *creative* photography. Response (a) includes "creative," directly reflecting the text's
emphasis on creativity, though "Find" is less precise than terms like "Create" or "Craft."
Response (b) is generic and omits the career-driven creativity central to the original text.
While neither response is perfect, (a) better aligns with the key themes.</think>
Therefore, Response (a) is better.
```

Figure 10: The illustration of Trace Clipping. The first part of thinking trace is removed, while the second part is used as the output thinking trace.

[16] https://huggingface.co/hkunlp/instructor-large

B.2 Judgment Thinking Optimization

For judgment thinking optimization, we leverage the prompts in Figure 11 and 12 for offline learning, and the prompts in Figure 13 and 14 for online learning. The offline learning is implemented based on Llama-Factory (Zheng et al., 2024), and online learning is implemented based on verl (Sheng et al., 2024). We present the hyper-parameter settings in Table 10 and 11.

hyper-parameter	SFT	offline	BT classifier
learning_rate	5.00E-06	1.00E-06	2.00E-06
per_device_train_batch_size	64	64	64
cutoff_len	4096	4096	4096
lr_scheduler	cosine	cosine	cosine
train_epoch	3	2	2
kl_beta	—	0.1	—
deepspeed_stage	zero3	zero2	—
precision	bf16	bf16	bf16
flash_attn	fa2	fa2	—

Table 10: Hyper-parameter settings for SFT, offline learning and BT classifier.

hyper-parameter	GRPO	PPO	Reinforce++
data.train_batch_size	1024	1024	1024
data.max_prompt_length	1024	1024	1024
data.max_response_length	3072	3072	3072
actor_rollout_ref.actor.optim.lr	1.00E-06	1.00E-06	3.00E-06
actor_rollout_ref.actor.ppo_mini_batch_size	256	256	1024
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu	16	4	8
actor_rollout_ref.actor.use_kl_loss	true	—	true
actor_rollout_ref.actor.kl_loss_coef	0.001	—	0.001
actor_rollout_ref.actor.kl_loss_type	low_var_kl	—	mse
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu	16	4	8
actor_rollout_ref.rollout.n	8	—	4
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu	16	4	8
critic.optim.lr	—	1.00E-05	—
critic.ppo_micro_batch_size_per_gpu	—	4	—
algorithm.kl_ctrl.kl_coef	0.001	0.001	—
trainer.n_gpus_per_node	8	8	8
trainer.nnodes	2	2	2
trainer.total_epochs	10	5	5

Table 11: Hyper-parameter settings for online learning algorithms.

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

[The Start of User Question]
{instruction}
[The End of User Question]

[The Start of Response (a)]
{response1}
[The End of Response (a)]

[The Start of Response (b)]
{response2}
[The End of Response (b)]

Please provide an evaluation by first offering a detailed explanation, and then end your answer with
"Therefore, Response (a) is better." or "Therefore, Response (b) is better.
```

Figure 11: The prompt template used for judgment prediction without $r_{strength}$.

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

[The Start of User Question]
{instruction}
[The End of User Question]

[The Start of Response (a)]
{response1}
[The End of Response (a)]

[The Start of Response (b)]
{response2}
[The End of Response (b)]

Given that {chosen} is better than {rejected}, please provide an evaluation by first offering a
detailed explanation, and then end your answer with "Therefore, Response (a) is better." or "Therefore,
Response (b) is better.".
```

Figure 12: The prompt template used for critique prediction without $r_{strength}$.

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

[The Start of User Question]
{instruction}
[The End of User Question]

[The Start of Response (a)]
{response1}
[The End of Response (a)]

[The Start of Response (b)]
{response2}
[The End of Response (b)]

Please provide an evaluation by first offering a detailed explanation. Then end your answer with the
following format: 'Therefore, Response (a/b) is better, and the strength is [[score]].'. For example:
'Therefore, Response (b) is better, and the strength is [[1]].'.
The strength denotes how much one response is preferred over the other, with the following scale:
- A strength of 1 indicates one response is slightly better than the other.
- A strength of 2 indicates one response is better than the other.
- A strength of 3 indicates one response is much better than the other.
```

Figure 13: The prompt template used for judgment prediction with $r_{strength}$.

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

[The Start of User Question]
{instruction}
[The End of User Question]

[The Start of Response (a)]
{response1}
[The End of Response (a)]

[The Start of Response (b)]
{response2}
[The End of Response (b)]

Given that {chosen} is better than {rejected}, please provide an evaluation by first offering a
detailed explanation. Then end your answer with the following format: 'Therefore, Response (a/b) is
better, and the strength is [[score]].'. For example: 'Therefore, Response (b) is better, and the
strength is [[1]].'.
The strength denotes how much one response is preferred over the other, with the following scale:
- A strength of 1 indicates one response is slightly better than the other.
- A strength of 2 indicates one response is better than the other.
- A strength of 3 indicates one response is much better than the other.
```

Figure 14: The prompt template used for critique prediction with $r_{strength}$.