Universiteit Leiden

# Master Computer Science

Towards Unified Evaluation in Open QA: A Multi-Level LLM-as-a-Judge Approach


| | |
|---|---|
| Name: | Ziyou Hu |
| Student ID: | s3963616 |
| Date: | 23/06/2025 |
| Specialisation: | Artificial Intelligence |
| 1st supervisor: | Zhaochun Ren |
| 2nd supervisor: | Qinyu Chen |
| External supervisor: | Zhengliang Shi |


Master's Thesis in Computer Science


Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

Real-life evaluation tasks are complex and require that evaluators have solid professional knowledge and fair judgment. Traditional manual methods are limited by their time-consuming nature, potential evaluator bias, and difficulty in allocating expert resources for large-scale tasks. With LLMs advancing in semantic understanding, this thesis introduces UnifiedJudge, an evaluation model trained with reinforcement learning (RL) mainly aiming for open QA scenarios, its training data includes both point-wise and pair-wise responses of other models, enabling it to handle detailed single-sample evaluations and comparative judgments. In the RL training phase, we designed different reward functions for different inputs. Despite its small size, UnifiedJudge performs well in multiple benchmarks. It outperforms state-of-the-art methods on some and nearly matches optimal performance on others, proving that RL training effectively boosts the evaluation ability while maintaining high computational efficiency.

# 1 Introduction

Evaluating things in real life is often tricky. It means making difficult decisions that rely on someone's personal opinion and their deep knowledge of a subject. The people who do the evaluation need to have a strong understanding of their field. They also have to deal with words or ideas that are not perfectly clear and understand how different pieces of information connect to each other. On top of that, they ought to make judgments that are fair and ethical. The usual way we evaluate things, where people do it by hand, has several problems. For one, it takes up a lot of time from highly experienced experts, and their time is very valuable, even the topics which are not complex and just daily conversation, it still needs time to evaluate. In addition, different evaluators may have their own personal biases, which can lead to different opinions. When you need to evaluate a lot of things, it becomes hard to get many experts together at the same time because their schedules might clash. However, new developments in large language models (LLMs) have shown their ability to understand what words mean and to figure out how information relates. Due to this progress, people are now considering using these LLMs to help with evaluation tasks.

Gu et al. [1] review the use of LLMs in auto-evaluation tasks. They cover model performance in different scenarios, metric selection, and the limitations of current methods. Their work shows that more studies are using LLMs as evaluation tools, which have demonstrated an ability to tell the difference between good and bad, much like human experts do. However, it is important to understand the difference between LLMs acting as "judges" and traditional "reward models." Reward models simply need to choose the better option from a set of choices. But LLMs acting as judges have a much harder job. They need to create detailed explanations for their evaluations, such as how they assigned scores and what the good and bad points are. This requires the models to have stronger reasoning skills and knowledge specific to the area they are evaluating. Currently, most studies train these evaluation models using supervised fine-tuning (SFT), such as JudgeLM [2], PROMETHEUS [3] [4] and Auto-j [5]. This means that they teach the LLMs by showing them examples of evaluations that humans have already done. But there is a clear drawback to the methods trained in this way: The models can only copy existing evaluation examples. They cannot improve their own way of evaluating things.

To overcome this problem, we introduce UnifiedJudge-3B. UnifiedJudge-3B is an evaluation model trained by the reinforcement learning (RL) algorithm. By interacting with its environ-

ment, UnifiedJudge can learn different ways to evaluate. This means that it can not only give scores, but also improve the evaluation over time as it goes through more learning steps. Specifically, our training data includes point-wise (single-sample independent rating) and pair-wise (paired-sample preference comparison) evaluation, which means that the model can do both detailed single-sample evaluation and comparative judgment. UnifiedJudge-3B is specifically designed to excel in open QA scenarios, where diverse and nuanced evaluations are crucial. Our model's ability to perform both detailed single-sample ratings and comparative judgments makes it particularly effective for assessing the quality and relevance of responses in these conversational environments.

In the RL training phase, we designed different reward functions for different evaluation tasks. For rating tasks, the reward mechanism focuses on the reasonableness and consistency of the rating criteria. For comparison tasks, it emphasizes the model's ability to identify slight differences. After training, we systematically tested UnifiedJudge-3B on multiple benchmarks. The results show that despite the small size of the model, UnifiedJudge-3B performs competitively. It outperformed current SOTA methods on some benchmarks and nearly matched optimal performance on others. Notably, it even outperformed some open source models with ten times more parameters, which shows the effectiveness of RL training in improving evaluation ability while maintaining high computational efficiency. Furthermore, we carried out a series of analytical experiments to dive deeper into the performance of the model and to reflect on the current method.

In Sec.2, we will deeply survey prior work in current and related fields, especially the research status of reasoning models and LLM as a Judge. In Sec.3, we will introduce our training data, covering prompt organization and sampling methods. In Sec.4, we will define the problem to be solved and present our designed method. Sec.5 will comprehensively display our model's performance and more in-depth analyses. Finally, in Sec.6 and Sec.7, we will examine the current results and draw conclusions.

# 2 Related work

## 2.1 LLM Reasoning

Large language models are getting much better at thinking, which means that new AI systems can now clearly show how they arrive at their answers. This evolution, driven primarily by the widespread adoption of Chain-of-Thought (CoT) mechanisms [6], has transformed the way models approach complex problems. Currently, the rise of Reinforcement Learning from Human Feedback (RLHF) has further enhanced these capabilities, enabling models to develop sophisticated reasoning skills through iterative optimization. As models like OpenAI's O1 [7] begin to emerge, we are seeing a growing trend of AI systems that can now engage in internal thought and reasoning processes before generating an answer. When fine-tuned with reinforcement learning, these models demonstrate remarkable abilities to perform logical analysis, draw causal inferences, and systematically decompose multi-faceted problems by leveraging their internal knowledge representations and generation architectures. This dual advancement in explicit reasoning and reinforcement-based learning has led to measurable improvements across challenging domains, particularly in mathematical problem-solving, code generation, and other tasks requiring multi-step reasoning. The combination of CoT's transparent reasoning pathways and RLHF's ability to refine decision-making processes has created models

that not only reach correct conclusions, but can do so through verifiable, human-like reasoning steps, substantially enhancing their reliability and practical utility in complex cognitive tasks.

More specifically, LLMs have demonstrated strong reasoning abilities in multiple domains. In mathematical problem solving, by incorporating Chain-of-Thought (CoT) prompt strategies and Reinforcement Learning from Human Feedback (RLHF), models can gradually parse algebraic equations, geometric proofs, and calculus computations. Zayne Sprague et al. [8] shows that introducing reasoning steps is effective in mathematical and symbolic problem-solving scenarios. In code generation and debugging, incorporating code data at different training stages can enhance the model's code generation and reasoning abilities [9]. In general NLP reasoning tasks, such as multi-step reasoning, LLMs can generate reasoning traces and task-specific actions alternately to complete multi-step reasoning which helps to overcome issues such as hallucinations and error propagation, to be specific, Yao et al. [10] suggest combining reasoning and acting in an alternating way. The model generates reasoning paths and task-specific actions, enabling better handling of edge cases. It allows access to external information, such as knowledge bases or the environment. During the execution of an action, the model can dynamically reason. This helps in creating, maintaining, and adjusting high-level action plans (reason to act) and interacting with the external environment to integrate additional information. All of these improve the model's interpretability and trustworthiness.

In addition to chain-of-thought, Yao et al. [11] proposed a tree - based thinking method. The problem solving process is broken down into a series of connected "thought" steps, which serve as intermediate steps. At each state point, several possible follow-up thoughts are generated and evaluated for effectiveness. This approach combines language-based generation and evaluation with search algorithms like BFS or DFS, systematically exploring the thought tree, allowing forward-looking and back-tracking as needed. This method improves performance on novel tasks that require non-trivial planning or search. Zhang et al. [12] proposed the Auto-CoT method to eliminate the need for manually created CoT prompts. It uses LLMs to automatically generate problem and reasoning chain examples, instead of relying on human-written ones. Through clustering techniques, problems are divided into different categories. Representative problems from each category are selected to generate reasoning chains. Simple heuristic rules are used to filter and optimize these generated reasoning chains, reducing errors, and improving usability. This shows that LLMs can perform CoT reasoning by automatically building examples. Similarly to this, Zhou et al. [13] proposed a novel prompting strategy. Inspired by the "least-to-most prompting" technique in educational psychology, this method uses a series of gradually more difficult prompts to help students learn new skills. Break down complex problems into a sequence of simpler sub-problems. These sub-problems are solved in order, with each solution building on the answer to the previous sub-problem to enhance the model's ability to handle complex problems. Sel et al. [14] also proposed a new strategy called the "Algorithm of Thoughts" (AoT). This strategy aims to guide LLMs through algorithmic reasoning paths by providing algorithmic examples in the context. These examples capture the exploration process from initial candidate solutions to verified ones, enabling LLMs to imitate algorithmic iterative thinking and complete tasks via one or few shot queries, which allows LLMs to explore different solutions during generation and backtrack when needed to find the optimal solution.

Although LLMs have shown remarkable abilities in understanding and generating natural language, they still have shortcomings in deep reasoning and open-domain Q&A where there are no standard answers. They struggle with vague or incomplete instructions that lead to logi-

cal breaks and hallucinations. This makes their output unreliable, especially in scenarios that require high factual accuracy. Given these challenges, using LLM as a judge to evaluate the output of other models or humans may be a solution. Using their reasoning abilities, LLMs can assess and compare various outputs. This approach compensates for their weaknesses in independent complex reasoning and utilizes their strengths in understanding and evaluating complex language outputs. Provides an efficient and scalable evaluation mechanism for different applications.

As research is growing on the use of LLM as a judge, there is less exploration of the use of LLMs' own reasoning to guide judgment. Current methods mainly use supervised fine-tuning to enhance judging abilities. Few works explore how to fully and purposefully engage LLMs' deep reasoning from different input angles to make more accurate and interpretable judgments. Our work aims to fill this gap. We propose a model that can flexibly draw on LLMs' reasoning abilities from different input levels. This enables the model to go beyond imitating surface-level human judgments. It can deeply understand and analyze information for more reliable and transparent evaluations.

## 2.2 Reward Models

In the field of artificial intelligence, reward models play a crucial role in improving the learning and behavior of models. They are particularly important in Reinforcement Learning from Human Feedback (RLHF), a method used to train AI models to conform to human preferences. These reward models serve two main purposes: to help train AI systems by providing feedback on their behavior, and to improve the efficiency of the testing phase by quickly evaluating the output. A recent and thorough study by Jialun et al. [15] has provided a comprehensive overview of the current state of reward models. Their work explores various aspects, including taxonomy, applications, and challenges.

Training data for reward models are typically obtained in two ways. First, human-labeled data is collected through methods such as active collection and data enhancement. Second, preference data are generated using AI systems (LLMs), which can supplement or replace human-labeled data, reducing annotation costs. However, data generated by AI systems may sometimes deviate from human preferences and are not entirely reliable.

Reward models can be classified into three types: Discriminative, Generative, and Implicit. Discriminative reward models, the most common type, consist of a base model(often a pre-trained language model or a similar neural network) and a specialized reward head. Their function is straightforward: they take certain inputs (like a generated text response or an action taken by an AI agent) and then produce a single reward scalar as an output. For example, the work by cai et al. [16]. introduced a family of multidimensional reward models trained on large datasets, supporting RLHF. These models are often based on the principles of the Bradley–Terry model [17]. This statistical model is particularly well suited for situations involving pairwise comparisons, such as human preferences, which operates on the assumption that the probability of one "competitor" (in this case, one AI response or action) being preferred over another is directly related to the ratio of their underlying "strength parameters" making it suitable for selecting human preferences. On this basis, the Direct Preference Optimization (DPO) [18] algorithm has emerged. The DPO algorithm offers a more efficient way to train language models directly from human preference data. Unlike traditional methods that might involve training a separate reward model and then using reinforcement learning techniques, DPO simplifies the

entire process, using a simple classification loss function. Reduces the complexity of aligning language models with human preferences and eliminates the need for complicated training procedures. Furthermore, it also reduces the need for extensive hyperparameter tuning, which can be a time-consuming and computationally expensive part of the model development process. Although discriminative reward models provide a single numerical score or preference, Generative reward models leverage the inherent generative abilities of large language models (LLMs). This approach is connected to the emerging and increasingly popular concept of LLM as a Judge. The detailed feedback provided by generative reward models directly aligns with this idea, enabling more informative assessments, which will be elaborated on in subsequent discussions.

Unlike the previous types, which mainly depend on the final output to determine a reward, the Implicit Reward Model focuses on providing reward signals for each individual step or process taken to reach an outcome. Yuan et al. [19] have provided a good explanation of this concept. They introduced a method to implicitly obtain a Preference Rating Model (PRM) using only response-level labels, which avoids the high cost of annotating intermediate steps; their method works by transforming the reward based on the final result into the log-likelihood ratio between the policy model and the reference model. This method allows for simultaneous training of Outcome Reward Models (ORMs) and implicit learning of PRM.

Some studies propose alternative methods to avoid explicit reward model training. Richemond et al. [20] present an approach to address the alignment issue of LLMs on single trajectory datasets. These datasets are different from the more common paired preference datasets. Instead, in a single-trajectory dataset, each prompt has only one associated completion (response) and a scalar reward (a single numerical score). Specifically, they train a reward model directly on these single-trajectory data using a simple mean squared error objective function. Other works focus on enhancing DPO's robustness. Xu et al. [21] propose a method to tackle the distribution shift problem in RLHF. Building on DPO, they develop algorithms that account for uncertainty in the training data distribution. By solving a minimax optimization problem to minimize expected loss, their approach improves the alignment performance of LLMs when preference distribution shifts occur.

Reward models have wide-ranging and practical applications; one important application is seen in the work of Dong et al. [22], they use reward models for training data selection, where reward models are used as a filtering mechanism, allowing them to identify and select high-quality data to be used in the training process. Another innovative application is demonstrated by Yuan et al. [23], They use reward models before the main training process begins to construct better datasets, by sampling responses from various sources. These sources can include the model itself, other large language models, and human experts (to ensure high-quality human-aligned examples), and they then use ranking loss to align the model's probabilities with human preferences.

During the training stage, reward models use reward signals to reinforce desired model behavior or constrain undesired actions. However, they face the challenge of reward hacking, Skalse et al. [24] have provided a clear description of this issue. Reward hacking occurs when models learn to exploit factors or loopholes in the reward function to maximize their perceived reward, rather than truly achieving the intended objective. For example, a model might learn that generating longer responses consistently yields higher rewards, even if the extra length does not add value, which means that the model gains unrealistic and inaccurate rewards and prevents the

model from capturing the true causal relationships between its actions or generated content and the desired outcome. Given the persistent challenge of reward hacking, Chen et al. [25] proposed an evaluation protocol and conducted large-scale studies on mitigating length bias in RL. They introduced a method that involves jointly training two distinct linear heads within the reward model, both designed to predict rewards. One head is linked to length, and the other is decoupled from it and focuses on actual content quality. During the RL phase, only the quality-related head is used, discarding length-related rewards; this approach prevents the model from exploiting length-based biases. Wang et al. [26] proposed a method that integrates causal inference to mitigate spurious correlations. By applying counterfactual invariance, it ensures that the reward model predictions remain consistent when intervening on input aspects unrelated to outcomes. Similarly, Liu et al. [27] presented a causal framework to differentiate between genuine quality signals from prompts and artifacts from responses. This approach improves the robustness of reward models.

In the inference stage, reward models can rank multiple outputs to identify the one that best aligns with human preferences. Ma et al. [28] proposed a heuristic greedy search algorithm based on Process-Supervised Reward Models (PRMs). This approach enhances model performance in the inference path search by providing real-time feedback on each reasoning step through PRMs. Jiang et al. [29] proposed a reward-guided tree search framework. Integrates policy models, reward models, and search algorithms to enhance the reasoning capabilities of LLMs.

## 2.3 LLM as a Judge

The concept of LLM as a Judge can be traced back to Ouyang's work [30]. Letting LLMs follow instructions to produce desired output is the basic paradigm of LLM as a Judge. The work of Wei et al. [31]. formally established this field. They innovatively used LLMs for evaluation and introduced benchmarks like MT-Bench and the Chatbot Arena platform. For the first time, they showed that LLMs can effectively evaluate other LLMs' outputs on a large scale, especially in judging the quality, relevance, and usefulness of generated text. The results align well with human assessments, proving that LLMs can be a feasible alternative for automated evaluation.

Building on this, using LLMs as judges requires evaluating multiple aspects of the output. Helpfulness, which evaluates the usefulness and informativeness of responses in addressing user queries or tasks, is one such aspect. A helpful response should provide accurate and comprehensive information that clearly answers the question. Auto-J [5] offers a thorough evaluation in this regard, taking helpfulness into full account. Harmlessness is another key criterion in LLM evaluation. It assesses whether the generated content is safe, moral, and free from bias, toxicity, or harmful stereotypes. LLM Guard [32] addresses this by creating an LLM-based input-output protection model. It evaluates safety risks in prompts and responses during human-AI conversations. However, safety risks can not be fully covered in one go. Managing newly emerging risk content remains a major challenge. Reliability is crucial for assessing the factual accuracy and consistency of a model's information. A reliable response must be based on verifiable facts and avoid presenting speculative or incorrect information as truth. RAIN [33] introduces a method where the model self-assesses its outputs during generation to ensure they align with human preferences. If misalignment occurs, the model retraces and attempts different outputs. This approach offers significant insights into verifying model reliability.

Relevance in LLM outputs measures how directly the generated text relates to the user's prompt or question. This concept also extends to using LLMs to judge the relevance between two pieces of input content, such as assessing the connection between two legal cases. Such a task demands high-level domain expertise, including a deep understanding of legal facts and their application. To address this challenge, Ma et al. [34] proposed a few-shot workflow that mimics how human experts perform relevance judgments. This method breaks down the annotation process into several stages, with each stage offering detailed explanations of expert reasoning as guidance. This stepwise guidance, requiring minimal expert input, helps activate the LLM's internal domain-specific knowledge, enabling it to conduct relevance annotation using the same standards as human experts. This approach has been proven effective and achieves high consistency with expert annotations. Regarding the feasibility of LLM outputs, this dimension assesses whether a model can provide a practical decision or workable solution. This is especially crucial when using LLMs for tasks involving strategic foresight or initial decision-making, where the ability to propose actionable plans is paramount. Yao et al. [11] enhanced LLMs' decision-making capabilities by conceptualizing the problem-solving process as a search through a Tree of Thoughts (ToT). In this framework, each node represents a partial solution, and each branch signifies a modification to that solution. Specifically, the problem-solving is broken down into a series of coherent "thought" steps. At each tree node, multiple potential subsequent thoughts are generated using specific prompts. A state evaluator is then designed to assess the progress of different states toward solving the problem, thereby enabling the exploration of multiple reasoning paths and facilitating self-evaluation of choices. In the end, the overall quality of model outputs should be evaluated. This is a comprehensive assessment covering coherence, fluency, conciseness, and style appropriateness. High-quality responses should be well-written, easy to understand, and convey information effectively. Jain's work [35] proposes a context-learning-based evaluation framework for multi-dimensional NLG assessment. It uses a few input - output examples, designs prompts for different dimensions, and samples context examples to assess model outputs. Tom Kocmi's work [36] presents an evaluation method using different prompt templates to leverage LLMs in assessing translation quality, revealing the potential of LLMs as evaluators.

At the time of writing, several studies have started to integrate LLMs' reasoning abilities into judging tasks, and these works were mostly published in 2025, indicating that LLM as a judge still has great potential to be explored. JudgeLRM [37] defines an evaluation task at the pair-wise level. It is trained with reinforcement learning (RL) to enhance the model's reasoning ability and devises a reward function that combines structure and content. This ensures the model's outputs include structured reasoning processes and accurate scores. Evaluation on benchmark datasets shows JudgeLRM [37] can effectively learn structured and credible reasoning paths, achieving better performance in evaluation tasks. J1 [38] also uses RL training. It mainly constructs synthetic data to transform judging tasks into verifiable ones and uses online RL algorithms for training, thus utilizing training data more efficiently. Both works propose model families where models of different sizes perform well in evaluation tasks.

## 2.4 Rule-based reward in LLM

The concept of rule-based reward (RBR) in LLMs traces back to work proposed by OpenAI [39], marking a significant evolution from prior alignment methods. Unlike earlier approaches that heavily relied on extensive human annotation or undifferentiated AI feedback, RBR directly leverages fine-grained, composable, and LLM-scored few-shot prompts to generate reward

signals during Reinforcement Learning (RL) training. This innovative approach enables greater control, accuracy, and ease of updates by ranking model completions based on their adherence to predefined rules, which are then used to assign rewards. By doing so, RBR provides an efficient and controllable pathway for aligning LLMs, particularly excelling in scenarios with clear, formalizable rules, such as detecting hate speech, ensuring specific output formats, or verifying the correctness of mathematical solutions. This inherent adaptability and efficiency explain why RBR has been swiftly adopted in modern RL algorithms like GRPO [40] and integrated into the reward models of widely used algorithms such as PPO, signifying its growing role as a key component in contemporary LLM development and safety alignment efforts.

### 2.5 User Simulation for Evaluating

User Simulation for Evaluating Information Access Systems is a technology for assessing the overall effectiveness of systems in interactively supporting users to accomplish tasks. It determines whether content meets human preferences by creating user simulators.User Simulation for Evaluating Information Access Systems assesses how effectively a system interactively aids users in task completion. It checks if content aligns with human preferences via user simulators. Balog et al. [41] offer a comprehensive description of evaluating information access systems. As user simulation is a broad concept with diverse applications, using simulators to mimic human behavior is one effective evaluation method.

Owoicho et al. [42] proposed a user simulator based framework. Given information-need descriptions, it can provide feedback on system responses. Its natural language responses are LLM-generated, helping maintain contextual coherence in multi-turn dialogues and mimicking real user behavior. Zhang et al. [43] developed a user simulator that generates realistic user-like reactions. This enabled an automated evaluation of conversational recommender systems. Unlike LLM-based methods, it uses personal knowledge graphs to represent user preferences. This ensures that all preference statements align with the user's historical behavior. Li et al. [44] proposed adversarial evaluation, training a discriminator to distinguish human-generated from machine-generated dialogues, assessing dialogue quality.The generator aims to produce human-like responses, while the discriminator seeks to tell human- and machine-dialogues apart. They are jointly optimized via adversarial training: the generator tries to fool the discriminator with its responses, and the discriminator strives to accurately identify the source of the responses. This method is insightful. If the discriminator can precisely determine whether the content is human-generated, it indicates a high alignment with human preferences. Thus, the discriminator can be used in the subsequent task training for fine-grained input evaluation.

Although user simulation can assess content, it mainly focuses on simulating human behavior for responses rather than specialized multidimensional evaluation and judgment. However, these studies show that content evaluation does not rely solely on generative or reward models; human-like simulation can also work. However, fully replicating human thinking remains a challenge.

## 3 Data

In order to train and evaluate a unified judge and enable it to provide corresponding feedback and ratings from both a point-wise and pair-wise perspective, we have looked at the datasets

mentioned or used in papers from the past 1-2 years and also searched for relevant open–source datasets. Although many papers do not clearly specify their training sets and only a few are open–sourced, we have still been able to find some high–quality datasets, some of which have become industry standards. To balance the quantity of datasets, we sampled different datasets and adapted them according to their different data structures. The specific methods will be elaborated below.

## 3.1 Point-wise dataset

For the point-wise dimension, we first collected the Feedback-Collection dataset used in the current SOTA model paper [3]. The dataset offers a complete rating standard for model evaluation, which mainly consists of five parts, and the size of the dataset is 100k. The structure example is as follows:

- **Instruction**: An instruction that users would prompt the LLM.
- **Response to Evaluate**: The response generated by the model based on the instruction.
- **Customized Score Rubric**: Scoring criteria proposed by users, which should be comprehensively considered when evaluating the model's output, which includes a general guideline for evaluating and scoring, along with specific decisions where scores from 1 to 5 should be assigned.
- **Reference Answer**: A response that can achieve a score of 5, by providing such responses, the model can understand the characteristics of a good answer and learn to develop its evaluation capabilities.
- **Score**: An integer score for the provided response that ranges from 1 to 5.

It should be especially noted that the score is a crucial part of our training data. It will serve as the basis for the reward information in the RL algorithm. By integrating the above data structures, we have organized the training prompt shown in the Fig.1

In addition to the above data, we have incorporated the dataset used by Auto–J [5], which encompasses a variety of real-world scenarios. It is primarily sourced from the chatbot-arena-conversations dataset [31]. The structure is as follows:

- **usermsg**: The raw input text for the model before being wrapped in a specific prompt or template. It includes the query, response, and instructions.
- **target output**: The target output for the given usermsg, which serves as the judgment to evaluate the response.
- **pred score**: The rating of the response given by GPT-4.
- **scenario**: The real-world scenario to which the query of this sample belongs.
- **source dataset**: The origin dataset of this sample.

During actual training, we did not use the target output field. We want the model to reason out the output on its own. Instead, the pred score, is a key basis for calculating rewards.

### 3.2 Pair-wise dataset

For the pair-wise dimension, we have also conducted a thorough investigation. Currently, many open–source training sets are designed or more suitable for DPO training, which usually only provide outputs from two different models and a preference choice. However, we prefer training sets with ratings. After careful evaluation, we selected the Preference–Collection [4] and JudgeLM–100K [2] datasets, sized at 200k and 100k respectively. The Preference-Collection dataset [4] contains 8 sections:

- **orig instruction**: The instruction that would be prompt to the LLM
- **orig response A/B**: Responses from models A and B.
- **orig reference answer**: A reference answer to the orig instruction.
- **orig criteria**: The scoring criteria for evaluating the orig response.
- **orig preference**: The selected model.
- **orig score A/B**: The score of response A or B.

Similarly, the orig criteria field offers a general scoring guideline, but it is important to note that the criteria vary across different data entries, the training prompt is shown in the Fig.1 For the JudgeLM–100K [2] dataset, we primarily use the following structure:

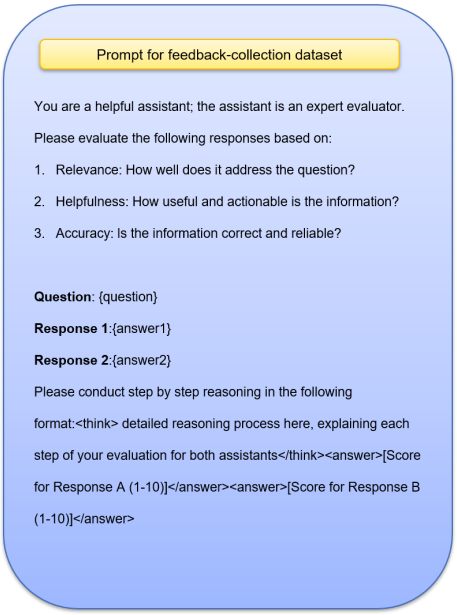
- **question body**: The text of the initial questions, which serve as instructions input into the model.
- **answer1 body / answer2 body**: Responses from different models.
- **score**: A list representing GPT-4's ratings for the two responses.

Given the large size of the two datasets and the potential training burden, we first performed cluster sampling on the instructions' embeddings to reduce computational costs while preserving data quality. To ensure data diversity, we carefully sampled from each cluster rather than taking random samples from the whole dataset. This approach allowed us to maintain a final dataset size of 100K without losing important patterns in the data.

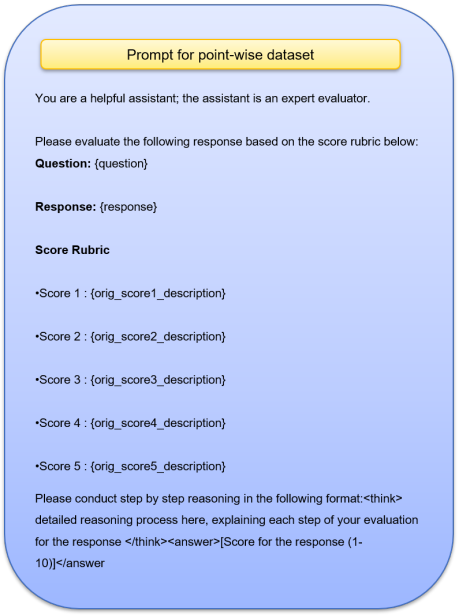
More specifically, we first sorted all the clusters by their size to understand their distribution. Then, we assigned smaller sampling weights to relatively larger clusters and higher weights to smaller clusters. This weighting strategy ensures that smaller clusters are adequately represented, enhancing data diversity while maintaining a balanced overall distribution. If the final sample size is short of 100K after cluster-based sampling, we supplemented it with additional random samples from the remaining data to reach the desired size. This two-step sampling process helped us achieve a more representative and diverse dataset for training.

## 4 Method

In this section, we begin by formalizing the problem setting, clearly defining the task objectives, input-output spaces. With the problem structure established, we then introduce the core algorithmic components, focusing on the Proximal Policy Optimization (PPO) method as our baseline approach. We outline its standard optimization objectives and policy update mechanisms, while noting key adaptations made to suit our specific task. The central contribution of




(a) Prompt for preference-collection dataset




(b) Prompt for point wise training

Figure 1: When presenting prompts for point-wise and pair-wise training, we didn't force them to be identical to enhance the model's generalization. The prompts for other datasets are in the Appendix.

our work lies in the redesign of the reward function, where we depart from conventional reward model in favor of a rule-based reward. We dedicate the final part to reward design, detailing the rule formulation process, its alignment with desired policy behaviors, and the practical considerations involved in its integration with PPO's training process.

## 4.1 Problem setup

We propose a unified large language model (LLM) capable of serving as an automated judge for both point-wise and pair-wise evaluation tasks. This dual–capability design allows the model to handle two distinct evaluation paradigms within a single system.

In the point-wise evaluation mode, when presented with a single instruction $x$ and a model response $r$, the model performs two key operations: (1) it assigns a scalar quality score $s$ and (2) it generates a textual judgement $J$ that provides explanatory feedback justifying the given score. For pair-wise comparisons, the model extends its functionality to evaluate two competing model responses. It produces: (1) individual scores ($s1$ and $s$) for each item in the pair, and (2) a comparative judgement $J$ that not only declares a preference but also explains the relative strengths and weaknesses between the two options. This dual-scoring approach ensures backward compatibility with point-wise evaluation while enabling relative quality comparisons. An overview of the problem and format is provided in Fig.2

## 4.2 RL training algorithm

We use the Proximal Policy Optimization (PPO) algorithm for training. Unlike traditional reward model methods, we use a rule-based reward approach. This has two advantages. First,

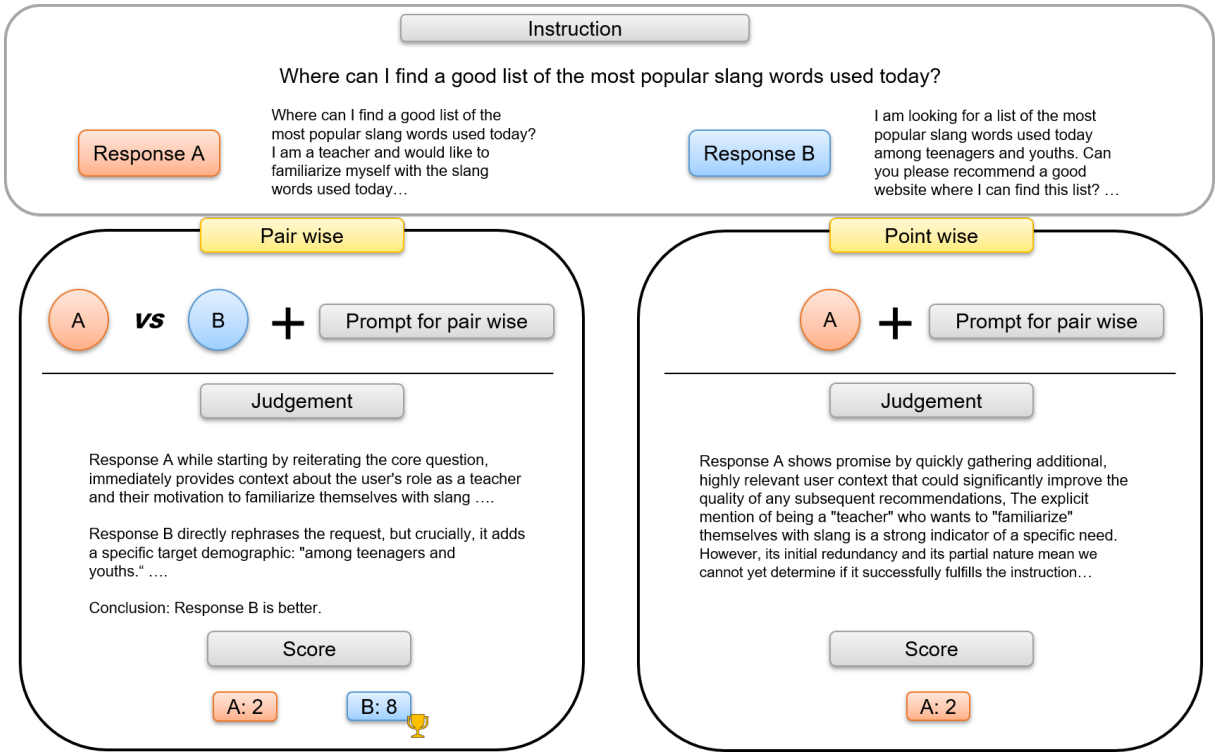



Figure 2: The models responds differently to the distinct inputs of point-wise and pair-wise data.

it eliminates the cost of a reward model (in terms of time, money, and equipment). Second, by integrating dataset labels, it allows for better control of reward granularity and better perception of the model's learning ability.The Eq.1 shows the PPO loss function. It optimizes the policy using advantage values while preventing large policy updates via a "clipping" mechanism. This ensures training stability. The advantage estimate $\mathbf{A}_t$ indicates how much better an action $\mathbf{a}_t$ is than the average in state $\mathbf{s}_t$, determined by rewards from the reward model.

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[ \min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t, \text{clip} \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A_t \right) \right] \tag{1}$$

PPO aims to maximize this loss function (or minimize its negative value). Its goals are:

- If a token has a high advantage value $\mathbf{A}_t > 0$, PPO tries to moderately increase its generation probability in the current policy.
- If a token has a low advantage value $\mathbf{A}_t < 0$, PPO tries to moderately decrease its generation probability.

The scalar reward measures how well the response aligns with human preferences. This reward is used to calculate the advantage estimate $\mathbf{A}_t$ for each timestep (each generated token). A value function is often used alongside to reduce variance and stabilize the advantage estimate. In simple terms, a higher reward tends to result in a higher advantage value $\mathbf{A}_t$ for the corresponding action (generating that token). Thus rewaro plays an important role in the training process.

## 4.3 Reward design

Our rule-based reward approach requires careful preparation from the data construction phase onward, as described in Sec.3. The core process involves decoding the model's generated output and comparing it against pre-defined standard answers to calculate precise reward signals. Since our system handles both point-wise and pair-wise evaluation tasks, we've developed distinct reward mechanisms for each mode. For point-wise inputs, the reward is computed through direct comparison with a single reference answer, assessing absolute quality across multiple dimensions. The pair-wise evaluation, on the other hand, requires comparing two model outputs against each other and against reference standards to determine relative quality differences. This dual reward scheme allows us to maintain evaluation consistency while adapting to different assessment needs. The rule-based design eliminates the need for training separate reward models, instead relying on transparent, pre-defined criteria that offer better control over reward granularity and more interpretable training signals.

### 4.3.1 Point-wise reward

To ensure consistent reward scaling across different evaluation prompts with varying score ranges, we implement a linear normalization procedure that processes the raw scores into a standardized [0, 1] range. This normalization begins by calculating the absolute discrepancy between the model's predicted score (extracted through regular expression pattern matching) and the ground truth reference score. We then apply a linear transformation that proportionally maps these error values onto our target reward scale, where 0 represents the maximum observed error (lowest reward) and 1 indicates perfect alignment with the ground truth (highest reward). Importantly, this normalization maintains the ordinal relationships between different quality levels while making the reward magnitudes directly comparable across all training data, and the formula is:

$$R_{\text{final}} = \max\left(0.0, \min\left(1.0, 1.0 - \frac{|P - T|}{S_{\max} - S_{\min}}\right)\right) \tag{2}$$

In Eq.2, $P$ is the predicted score, $T$ is the ground truth, we chose $S_{max} = 5$, $S_{min} = 1$ to keep the reward into the [0,1] range.

### 4.3.2 Pair-wise reward

For pair-wise evaluation, we implement a graduated reward system that combines strict ranking requirements with continuous quality assessment. The core mechanism first verifies the relative ordering between the two predictions compared to the ground truth - completely reversing the correct order results in zero reward, while perfect matches receive the maximum reward of 1.0. In intermediate cases where the ordering is correct but predictions aren't perfect, we calculate a quality-adjusted reward through a multi-step process. For each prediction, we compute its closeness to the ground truth using the formula

$$R = \max\left(0.0, 1.0 - \frac{|\text{difference}|}{\tau}\right) \tag{3}$$

where tau serves as our tolerance parameter set to 1.0. The final reward then combines a base score of 0.6 for correct ordering with a weighted quality component (40% of the average closeness between the two predictions), yielding values smoothly distributed between 0.6 and 1.0. This design ensures models must first satisfy the basic ranking requirement

before fine-tuning their predictions for higher rewards through improved numerical accuracy. More specifically, we categorize different situations: Let $(P_1, P_2)$ be prediction score 1 and 2, $(T_1, T_2)$ be truth 1 and 2, we denote the ranking order with an indicator, as shown in the Eq.4 and Eq.5:

$$O_P = \begin{cases} 1 & \text{if } P_1 \geq P_2, \\ 0 & \text{if } P_1 < P_2. \end{cases} \tag{4}$$

$$O_T = \begin{cases} 1 & \text{if } T_1 \geq T_2, \\ 0 & \text{if } T_1 < T_2. \end{cases} \tag{5}$$

As shown in Eq.6, our pair-wise reward mechanism employs a composite structure consisting of a base reward and an accuracy-based bonus. The base reward of 0.6 serves as the minimum guaranteed reward when the relative ranking is correct but predictions are not perfect, ensuring the model receives substantial feedback even for partially correct responses. The additional reward component is determined by the absolute error between the predicted score difference and the true difference, we create a smooth reward gradient that proportionally reflects prediction accuracy while maintaining the 0.6 baseline for correct rankings. The total reward combines these two components, with the base reward ensuring fundamental ranking correctness and the accuracy-based bonus refining the model's scoring precision. The reward is capped at 1.0 for cases where predictions exactly match the ground truth, establishing a clear upper bound for perfect performance.

$$R = \begin{cases} 1 & \text{if } P_1 = T_1 \quad \text{and} \quad P_2 = T_2, \\ 0 & \text{if } O_P \neq O_T, \\ 0.6 + 0.4 \cdot \max\left(0.0, 1.0 - \frac{|(P_1 - P_2) - (T_1 - T_2)|}{\tau}\right) & \text{otherwise.} \end{cases} \tag{6}$$

# 5 Experiments

## 5.1 Experimental Setup

For training, we train our model on top of verl [45], which is a flexible and efficient RL training library for LLMs, to be more specific, we use Torch FSDP [46] for parallel training and employed vllm [47] as the model inference engine. All the training data and the data preparation method are as described in Sec.3. We implement our method by creating a file of custom functions. For evaluation, we collect and self-source some highly popular and widely used benchmarks from current SOTA papers, as listed below:

- **Feedback-Bench** [3] is a point-wise dataset, which has 1k samples and its structure is consistent with the Feedback-Collection described in Sec.3
- **JudgeLM-Bench** [2] is a pair-wise dataset consisting of 5,000 judge samples, where all judge samples contain high-quality judgements made by GPT-4, and its data structure is consistent with the JudgeLM–100K [2] described in Sec.3.
- **Auto-J Eval** [5] is a pair-wise dataset with 1,392 samples, covering 58 different real-world scenarios. Hence, it serves as a solid platform for evaluating the abilities of different assessors. In the original dataset, the labels are categorized as 0 (the first response is selected), 1 (the second response is selected), and 2 (a tie). To enhance interpretability, we excluded tie data in practice.

- **Judge-Bench** [48] is a pair-wise benchmark for evaluating LLM-based judges on challenging response pairs, which has 620 samples in total, and these pairs span knowledge, reasoning, math, and coding. It is highly challenging. All responses were generated by GPT-4o and Claude-3.5-Sonnet. Many strong models, such as GPT-4o, perform only slightly better than random guessing.
- **PandaLM Eval** [49] is a human-labeled test dataset that is reliable and aligns with human preference for text, where the data are generated and sampled from the human evaluation data of Self-Instruct [50]
- **Reward-Bench** [51] is a dataset evaluates capabilities of reward models over the multiple categories and sampled from several popular bench.

To compare performance of our model, we collected various baseline models: (1) Zero-shot LLMs as judges, such as the Llama series, GPT-3.5 [52], and GPT-4 [53]. (2) State-of-the-art reward models such as Auto-J [5]. (3) State-of-the-art generative models fine-tuned for judge tasks, including models trained with supervised fine-tuning and RL, such as JudgeLRM [37].

## 5.2 Judge Performance

### 5.2.1 Point-wise

To directly assess our model's performance on a point-wise basis, we conduct an evaluation using the relevant benchmark dataset. The primary metric employed for this comparison is the Pearson correlation coefficient. This coefficient is computed by comparing our model's outputs against the ground-truth annotations provided by GPT-4. A higher Pearson correlation coefficient consistently indicates superior performance, demonstrating a stronger alignment between our model's predictions and the established golden standard.

| **Dataset** | *Feedback-Bench (GPT-4 as ground truth)* | | |
|---|---|---|---|
| **Criteria** | **Pearson** | **Spearman** | **Kendall-Tau** |
| *Existing Baseline. (* from original paper)* | | | |
| LLAMA2-CHAT 7B* | 0.485 | 0.478 | 0.422 |
| LLAMA2-CHAT 13B* | 0.441 | 0.452 | 0.387 |
| LLAMA2-CHAT 70B | 0.572 | 0.564 | 0.491 |
| GPT-3.5-TURBO* | 0.636 | 0.617 | 0.536 |
| AUTO-J (13B)* | 0.637 | - | - |
| MISTRAL-INSTRUCT-7B* | 0.586 | - | - |
| *Base Models.* | | | |
| PROMETHEUS-1-7B | 0.893 | 0.890 | 0.813 |
| PROMETHEUS-2-7B | 0.826 | 0.819 | 0.743 |
| *Ours.* | | | |
| UnifiedJudge-3B | **0.914** | **0.914** | **0.852** |

Table 1: Pearson, Kendall-Tau, Spearman correlation with data generated by GPT-4-0613, the data marked with * are from the original paper, and we tested the models from the Base Models and used each model's corresponding prompt template during evaluation.

In Table.1, it clearly shows that un-fine-tuned models, when used as zero-shot judges, deliver only mediocre performance. Interestingly, Llama2-13B performs worse than its smaller counterpart, Llama2-7B, which is an unexpected outcome. GPT-3.5-TURBO and AUTO-J (13B) exhibit similar performance levels, suggesting that they have comparable capabilities in this

zero-shot setting,in contrast, PROMETHEUS-1-7B and PROMETHEUS-2-7B, which were trained using supervised fine-tuning, show a noticeable improvement over the un-fine-tuned models. This highlights the significant benefit of fine-tuning for this task. However, it should be noted that PROMETHEUS-2-7B unexpectedly underperforms PROMETHEUS-1-7B, indicating that more parameters or a different fine-tuning approach do not always guarantee better results. Finally, our model slightly outperforms PROMETHEUS-1-7B, achieving a gain of $+0.02$, demonstrating a marginal but positive improvement in performance.

### 5.2.2 Pair-wise

For a comprehensive pair-wise performance comparison, we rigorously evaluated our model using the Auto-J Eval benchmark. The crucial metric for this assessment was the accuracy of preference predictions, where GPT-4's established preferences are served as ground truth. This specific metric allowed us to precisely measure how well our model's selections of preferred responses aligned with those determined by GPT-4, thus providing a direct indication of its ability to replicate human-like judgment in comparative scenarios.

| **Dataset** | *Auto-J Eval (GPT-4 as ground truth)* | | |
|---|---|---|---|
| **Criteria** | **Accuracy** | **Precision** | **Recall** |
| *Existing Baseline. (* from original paper)* | | | |
| LLAMA2-CHAT 7B* | 0.457 | - | - |
| LLAMA2-CHAT 13B* | 0.433 | - | - |
| LLAMA2-CHAT 70B | 0.506 | - | - |
| GPT-3.5-TURBO* | 0.711 | - | - |
| AUTO-J (13B)* | 0.766 | - | - |
| GPT-4-1106-PREVIEW* | 0.83 | - | - |
| ULTRA-RM (13B)* | 0.598 | - | - |
| PAIR RM (0.4B)* | 0.590 | - | - |
| *Base Models.* | | | |
| JudgeLRM-3B | 0.723 | 0.778 | 0.675 |
| PROMETHEUS-2-7B | 0.771 | 0.797 | 0.715 |
| *Ours.* | | | |
| UnifiedJudge-3B | **0.713** | **0.731** | **0.748** |

Table 2: The data marked with * are from the original paper, which does not provide Precision and Recall. ULTRA-RM [54] is a reward model trained on preference datasets. PAIR RM [55] is a smaller model trained on mixed data.

As Table 2 illustrates, the LLAMA2 models struggled in this evaluation, with both the 7B and 13B versions even performing worse than random guessing. This suggests that without specific fine-tuning for preference prediction, these models are not effective as judges. In contrast, GPT-3.5 showed a significant improvement in performance, indicating its superior capability in understanding and predicting preferences. AUTO-J (13B), a leading model in this domain, also performed well. It is particularly insightful to note that even GPT-4, despite being the source of our ground truth labels, did not achieve a perfect self-assessment score. This highlights an inherent uncertainty within even the most advanced models, underscoring the complexity of preference judgment.

Among all the models we tested, PROMETHEUS-2-7B notably stands out with its strong performance. Our model, UnifiedJudge-3B, is basically the same as JudgeLRM-3B, which substantially demonstrates that our model can match the performance of sota models within

the 3B parameter class. Furthermore, the performance gap between our 3B model and the leading 7B models is not substantial, which strongly suggesting the considerable potential of 3B models to achieve high levels of performance with efficient resource utilization.

| **Dataset** | *JudgeLM-bench (GPT-4 as ground truth)* | | |
|---|---|---|---|
| **Criteria** | **Accuracy** | **Precision** | **Recall** |
| *Existing Baseline. (* from original paper)* | | | |
| Qwen2.5-3B-Instruct* | 0.723 | - | - |
| Qwen2.5-7B-Instruct* | 0.768 | - | - |
| JudgeLM-7B* | 0.811 | - | - |
| JudgeLM-13B* | 0.843 | - | - |
| AUTO-J (13B)* | 0.749 | - | - |
| PandaLM-7B* | 0.686 | - | - |
| *Base Models.* | | | |
| JudgeLRM-3B | 0.823 | 0.865 | 0.815 |
| PROMETHEUS-2-7B | 0.746 | 0.791 | 0.714 |
| *Ours.* | | | |
| UnifiedJudge-3B | **0.839** | **0.824** | **0.826** |

Table 3: The performance comparison of various models on the JudgeLM-bench, in which the PandaLM [49], a generative evaluation model, can efficiently assess responses from different models.

Table 3 shows that our model achieves excellent performance on the JudgeLM-bench. Specifically, its performance is second only to JudgeLM-13B, indicating its near-state-of-the-art capabilities. Furthermore, our model exhibits a slight but notable improvement over JudgeLRM-3B, highlighting its competitive edge within its parameter class.

In contrast, PROMETHEUS-2-7B performs below this benchmark, aligning more closely with the performance of AUTO-J (13B). This is a crucial observation as it indicates that our model surpasses current state-of-the-art (SOTA) models on this specific benchmark, effectively outperforming other 7B models and even larger models in this evaluation. This underscores the efficiency and effectiveness of the design and training of our model.

| **Dataset** | *PandaLM Eval* | | |
|---|---|---|---|
| **Criteria** | **Accuracy** | **Precision** | **Recall** |
| *Existing Baseline. (* from original paper)* | | | |
| Qwen2.5-3B-Instruct* | 0.685 | 0.509 | 0.561 |
| Qwen2.5-7B-Instruct* | 0.639 | 0.619 | 0.676 |
| JudgeLM-7B* | 0.651 | 0.669 | 0.719 |
| JudgeLM-13B* | 0.689 | 0.682 | 0.741 |
| PandaLM-7B* | 0.593 | 0.573 | 0.592 |
| *Base Models.* | | | |
| JudgeLRM-3B | 0.768 | 0.818 | 0.847 |
| PROMETHEUS-2-7B | 0.728 | 0.771 | 0.687 |
| *Ours.* | | | |
| UnifiedJudge-3B | **0.729** | **0.798** | **0.784** |

Table 4: The performance comparison of various models on the PandaLM Eval.

Table 4 shows the performance of various models in PandaLM Eval, where all labels are manually annotated by humans, better reflecting whether the models align with human preferences. Both JudgeLM-7B and PandaLM-7B perform moderately. Specifically, PandaLM-7B has a

low accuracy of 0.593, underperforming even the Qwen2.5 series models without fine-tuning. In contrast, our model performs well, matching PROMETHEUS-2-7B and nearly reaching JudgeLRM-3B levels. This indicates that our model better meets the human preference requirements in this evaluation.

| **Dataset** | *Judge-bench (GPT-4o and claude-3.5-Sonnet as ground truth)* | | |
|---|---|---|---|
| **Criteria** | **Accuracy** | **Precision** | **Recall** |
| *Existing Baseline. (* from original paper)* | | | |
| J1-Llama-8B (random single-order data)* | 0.483 | - | - |
| J1-Llama-8B (both-order data)* | 0.631 | - | - |
| J1-Llama-8B (verdict consistency reward)* | 0.523 | - | - |
| EvalPlanner-Llama-8B* | 0.382 | - | - |
| *Base Models.* | | | |
| JudgeLRM-3B | 0.577 | 0.537 | 0.645 |
| PROMETHEUS-2-7B | 0.555 | 0.514 | 0.500 |
| *Ours.* | | | |
| UnifiedJudge-3B | **0.530** | **0.532** | **0.604** |

Table 5: The performance comparison of various models on the Judge bench, in which the EvalPlanner [56], a thinking generative evaluation model.

Finally, we evaluated our model on the challenging Judge-bench. As Table 5 shows, none of the models, including our own, achieved remarkably high performance on this benchmark. This outcome is directly attributable to the inherently high difficulty of the Judge bench, primarily because it has multiple diverse fields and presents questions that demand specialized knowledge. This broad scope and specific knowledge requirement pose a significant challenge for all models.

Despite these considerable difficulties, our model still shows a relatively good performance. It notably outperformed EvalPlanner, another 3B model, indicating its superior capability within the same parameter class. The performance gap between our model and JudgeLRM-3B was observed to be small, further highlighting our model's competitive standing. Moreover, our model also approached the performance level of PROMETHEUS-2-7B. In a broader comparison with the J1-Llama-8B series, our model performed relatively well.

As shown in all tables, our model consistently shows strong overall performance on various benchmarks. Remarkably, it surpasses numerous larger models, including GPT-3.5, Llama2-70B, and even fine-tuned state-of-the-art models within the 7B to 13B parameter range, illustrating its exceptional capabilities despite its smaller size. Across multiple evaluation benches, our model proves its superiority.

Compared to the absolute optimal models on each specific benchmark, our model consistently either closely approaches or nearly matches their top performance. This highly competitive showing is significant as it highlights the strong competitiveness of our lightweight model. The maintains excellent performance while simultaneously offering higher efficiency and substantial resource savings.

## 5.3 Reward Curve, Loss and Score Analysis

### 5.3.1 Reward Curve

Fig.3 shows the clear trend of the reward curve throughout the training process. A higher reward value signifies that the model's outputs are better aligned with human preferences. During the initial phase of training, specifically within the first 200 steps, the curve exhibited a steep upward trajectory. This sharp increase indicates highly effective early learning and a significant improvement in the model's ability to generate preferred outputs.

Following this initial rapid ascent, the reward curve transitioned into a steady upward trend, demonstrating continued learning and refinement. The final average reward achieved was 0.823, which serves as strong evidence that the model not only learned effectively in its early stages, but also improved consistently throughout the duration of training.


Figure 3: Mean reward over 1179 training steps

### 5.3.2 Loss Curve

Fig.4 shows the actor's loss of entropy during the training process. Initially, the high entropy loss indicates that the model's policy was highly random, signifying an active phase of exploration. This is crucial in early training, allowing the model to discover a wide range of possible actions and their outcomes.

As training progresses, the loss of entropy begins to gradually decline. This reduction signals that the policy is becoming more deterministic, meaning that the model is increasingly confident in its chosen actions. Essentially, the model started to exploit the effective actions it had learned, thus reducing its need for extensive exploration. In the later stages of training, the entropy loss eventually stabilized. This stabilization indicates that the model had found a beneficial balance between exploration and exploitation.

Fig.5 shows the Policy Gradient Loss of the actor throughout the training process. In the early stages, the loss is notably high and exhibits significant fluctuations, ranging between 0.1 and 0.3. This behavior is consistent with the initial high entropy loss we observed, which indicates


Figure 4: Actor's entropy loss over training steps

that the model is in an active phase of exploring diverse actions. High loss in this phase is expected as the policy is still far from optimal.

Moving into the middle stage of training, the average loss begins to decrease, although it still fluctuates. However, the amplitude of these fluctuations becomes smaller, suggesting that the model is gradually converging towards a more stable policy. The occasional sharp peaks seen during this period could be attributed to sudden shifts in the environment's feedback, perhaps due to a particularly challenging batch of data.

Finally, in the late stage of training, the loss stabilizes remarkably and fluctuates minimally, mostly staying between 0.1 and 0.2. At this point, the actor has successfully learned a fairly stable and effective policy.


Figure 5: Actor's policy gradient loss over training steps

By looking at the reward curve and the policy gradient loss curve together, we gain a more

complete understanding of the model's training process. In fact, we can observe that the model continued to learn throughout the training process, as evidenced by the rewards that consistently increased and eventually reached a plateau. This indicates that the model was successfully optimizing its actions to align better with the desired outcomes. However, a closer look at the policy gradient loss curve, particularly its spikes, reveals that the policy sometimes updated too aggressively on certain batches of data. These spikes suggest moments where the model might have taken a large step in its policy updates, potentially driven by challenging data within a batch.

### 5.3.3 Score Distribution

Here we selected three different pair-wise benches and one point-wise bench to analyze the scores given by our model.

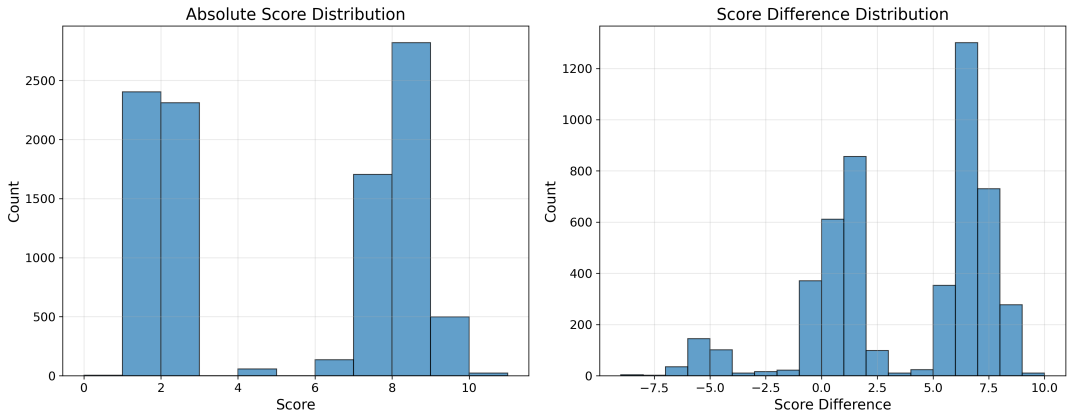



Figure 6: Score distribution on JudgeLM-Bench

Fig.6 illustrates the distribution of scores given by our model in JudgeLM-Bench. "Absolute score" indicates the direct scores assigned by the model to each of the two responses. "Score Difference" is calculated by subtracting the score of the rejected response from that of the chosen one. A negative difference signifies an incorrect judgment by the model, whereas a positive value indicates a correct one, from which the distribution of absolute scores given by our model on JudgeLM-Bench, with most scores clustering in the 1-2 and 7-9 ranges, and the highest frequency at 8. For score differences (chosen score minus rejected score), negative values are less common, indicating mostly correct model predictions. Among positive score differences, the range 0-2.5 is significant, and the range 6-8 is the most prevalent. This suggests that either the model perceives the responses as very similar or quite distinct in quality.

Fig.7 shows the reliability diagram of our model in JudgeLM-Bench. The model's original $pred\_scores([score\_A, score\_B])$ are not probabilities. To convert them into the probability of "A being better than B", we use the Sigmoid function. First, compute the difference between the two prediction scores: $score\_dif = score\_A - score\_B$. This difference represents the model's "confidence strength" in its preference for A over B (log-odds). Then, pass $score\_dif$ through the Sigmoid function to get a probability between 0 and 1. We also calculate the Brier score, which measures the mean squared error between prediction probabilities and actual

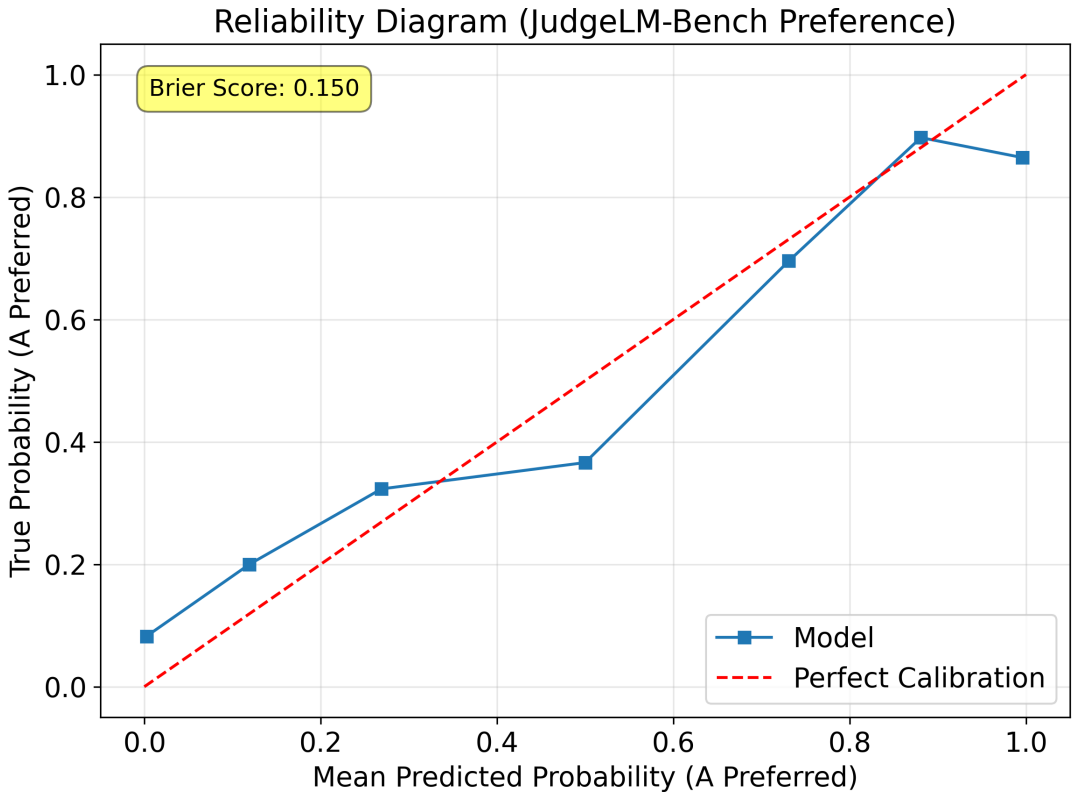



Figure 7: Reliability Analysis on JudgeLM-Bench

binary outcomes. A Brier score closer to 0 indicates more accurate predictions and better calibration.

The x-axis in the figure shows the model's average predicted probability that response A is better than response B. The y-axis indicates the actual proportion of times response A is better than response B within a certain predicted probability range. The red dotted line represents perfect calibration, where predicted probabilities match actual probabilities. For example, if the model predicts a probability of 0.6 that A is better than B, then in all cases where the model predicts 0.6, A should actually be better than B in 60% of instances. The blue line reflects the model's true calibration performance across different predicted probability ranges. In low predicted probability ranges, the blue line lies above the red dotted line. This means that when the model predicts a low probability (0.0 to 0.25) that A is better than B, the actual proportion of A being better than B is higher. This suggests that the model is underconfident in this low probability range; it thinks that the probability of A being better than B is low, yet the actual occurrence is more frequent than its prediction. In the medium probability range (X-axis 0.25-0.45), the blue line is slightly above the red dotted line, showing slight underconfidence. Between the X-axis values of 0.45 and 0.8, the blue line gradually moves below the red dotted line, especially around 0.5, indicating overconfidence in this range. The model predictions here are more confident than those justified by reality. In the high probability range, when the X-axis is about 0.88, the blue line is close to the red dotted line, indicating good calibration when the model is highly confident. However, as the predicted probability approaches 1.0, the blue line again falls below the red dotted line, suggesting overconfidence in cases where the model is certain that A is better than B.

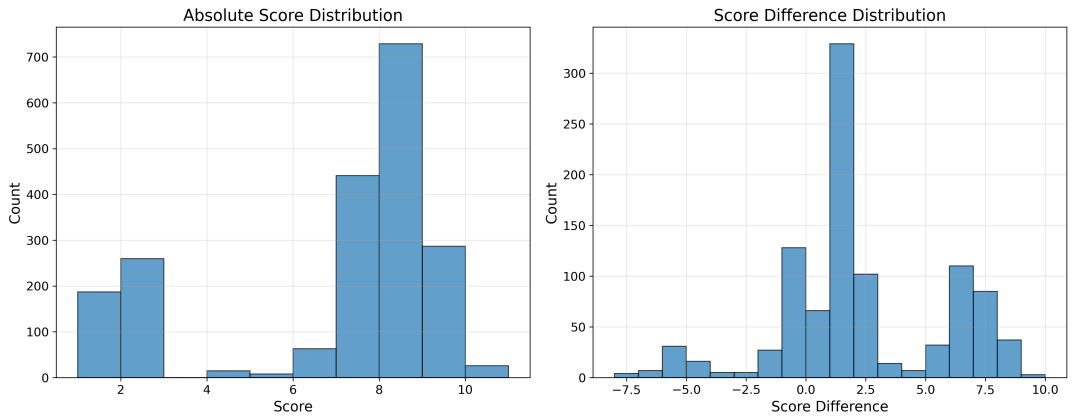



Figure 8: Score distribution on Autoj Eval

Fig.8 shows the score distribution of our model in Autoj Eval. Unlike the case in JudgeLM-Bench, the absolute scores given by the model are overwhelmingly in the 7 to 9 range, with 8 being the most frequent. In terms of score differences, the 1–2 range is the most common. In addition, negative values make up a relatively small proportion. This indicates that the model has a high overall accuracy on this benchmark and that in most cases it considers the quality of the two responses to be fairly similar.

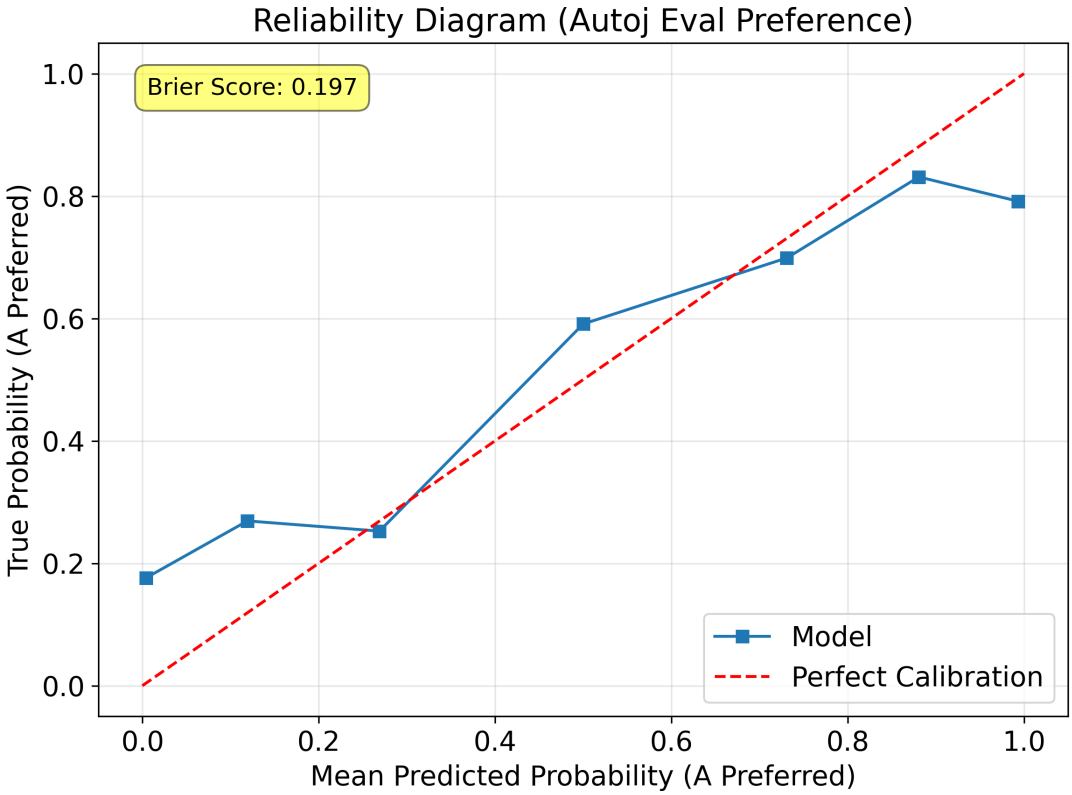



Figure 9: Reliability Analysis on Autoj Eval

Fig.9 shows the reliability diagram of our model in Autoj Eval. In the low probability range (X-axis at 0.15), the blue line is at Y=0.27. This means that when the model predicts a low 0.15

probability that A is better than B, the actual probability is higher, indicating underconfidence. In the low to medium probability range (X-axis at 0.25), the blue line aligns with the red dotted line at Y=0.25, showing good calibration here. In the medium probability range (X-axis 0.35 - 0.7), the calibration is mixed. With a predicted probability of 0.5 , the model is underconfident, but at 0.7, it is well calibrated. In the high probability range, the blue line is below the red dotted line. This means that when the model is highly confident (predicting a high probability), the actual probability is lower, showing overconfidence. The Brier score of 0.197 also indicates that the model has calibration deviations.

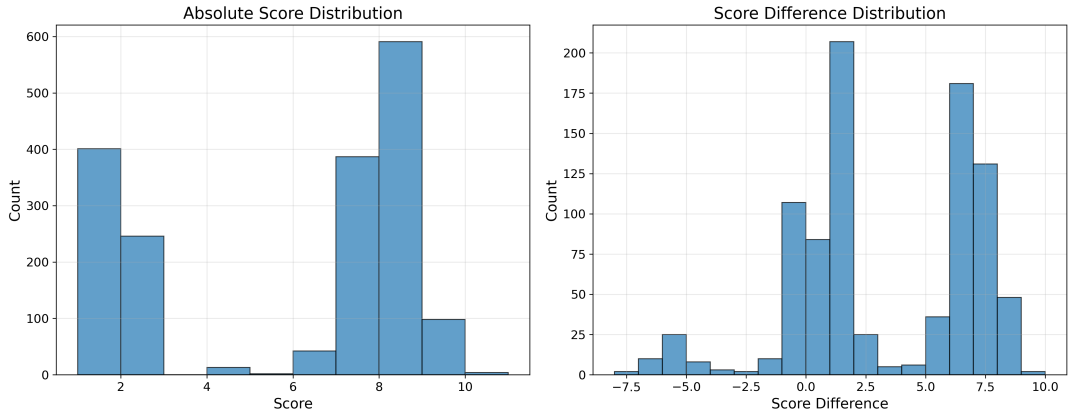



Figure 10: Score distribution on PandaLM Eval

Fig.10 shows the score distribution of our model in PandaLM Eval. The absolute scores also show two polar distributions. For the distribution of score differences, the interval of 0-2 takes the largest proportion, followed by the interval of 6-7, which indicates that the model views the two responses as more evenly matched on this dataset.

Fig.11 shows the reliability diagram of our model in PandaLM Eval. In the low probability range (X-axis at 0.15), the blue line is at Y = 0. Here, the predicted probability of the model is 15% that A is better than B is overconfident, since the actual probability is 0%. In the medium to high probability range (X-axis 0.75 - 0.9), the blue line is below the red dotted line. At predicted probabilities of 0.75 and 0.9, the actual probability of around 0.65 indicates overconfidence. The Brier score of 0.169 also suggests calibration deviations. Overall, the reliability diagram indicates that the model has mixed calibration issues in different predicted probability ranges in the PandaLM Eval.

Fig.12 shows the score distribution of our model in Feedback-Bench, where the score difference is the predicted score minus the ground truth score. The absolute scores are evenly distributed, with 2 points being the most frequent. Most score differences fall within 0-1, and other ranges account for only a small proportion. This indicates that the model's predicted and actual values are very close on point-wise data, reflecting good model performance.

Fig.13 shows the model calibration in Feedback-Bench. The red dotted line represents the ideal diagonal for perfect calibration. If the model were perfectly calibrated, its predicted average score would equal the true score. In low score ranges, when the true score is 1.0, the average predicted score of the model is about 1.3, indicating an overestimation. In mid-score ranges, the blue line closely follows the red dotted line, indicating good calibration where confidence matches accuracy. In high score ranges, when the true score is 5.0, the average predicted

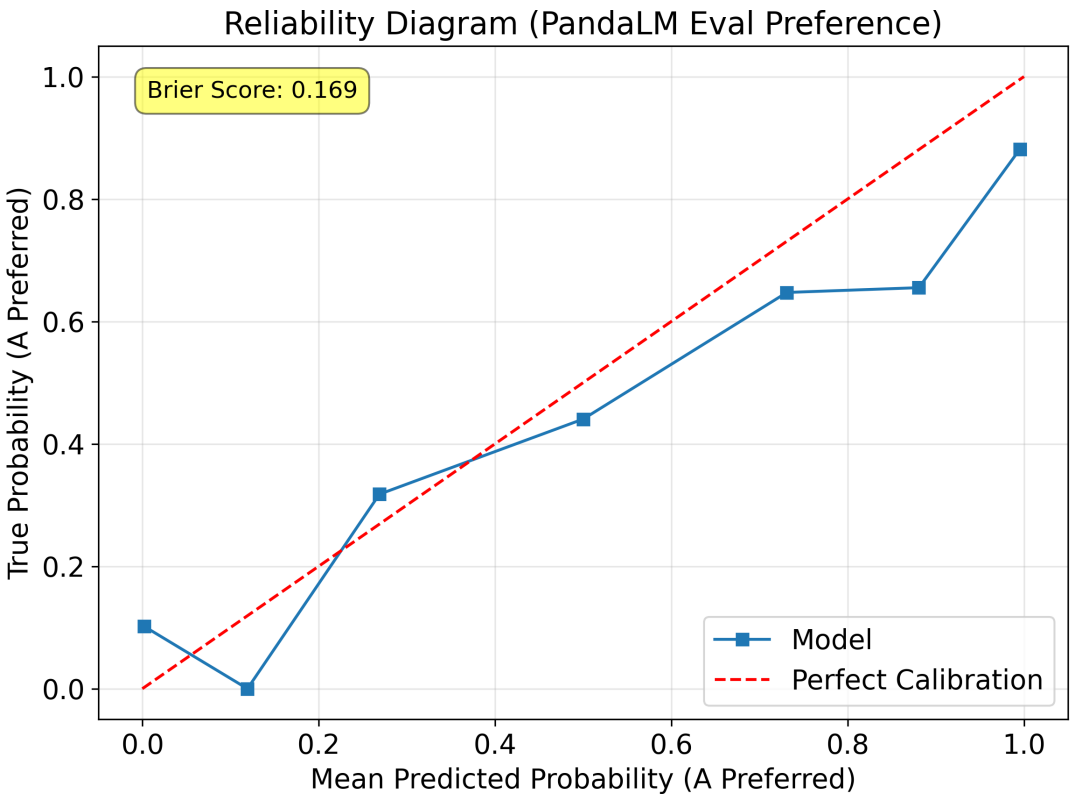



Figure 11: Reliability Analysis on PandaLM Eval

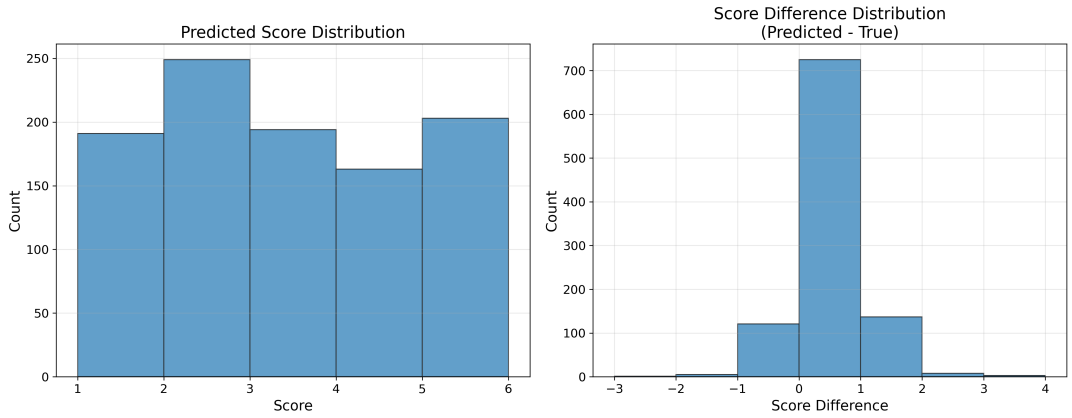



Figure 12: Score distribution on Feedback-Bench

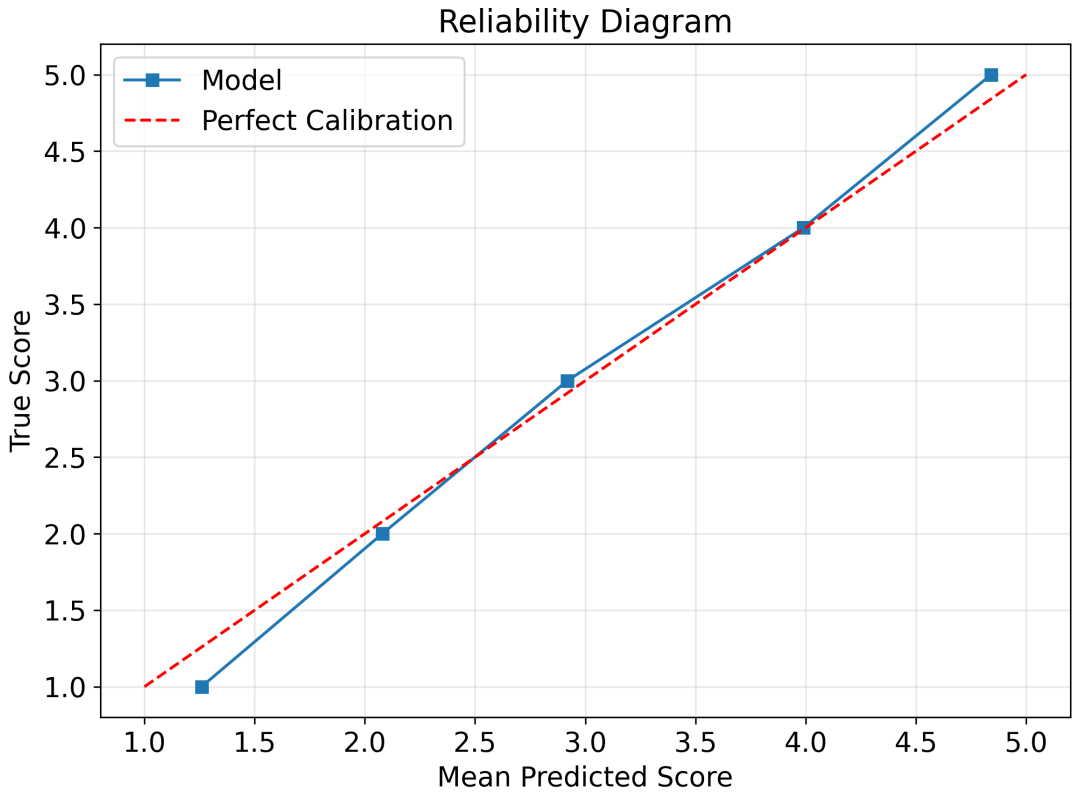



Figure 13: Reliability Analysis on Feedback-Bench

score is around 4.8, showing underestimation and slight underconfidence. Overall, the model demonstrates good calibration.

## 5.4 Position Bias

To thoroughly investigate the influence of response position order on our model's performance, we conducted an additional series of tests on two different benchmarks. Specifically, we used the first 200 data from the JudgeLM-bench and all available data from the Auto-J Eval. For each data, we swapped the positions of the responses within the prompts. For instance, if the original prompt presented "ResponseA and ResponseB," we modified it to present "ResponseB and ResponseA." The results of these tests, performed on this position-swapped data, are detailed in Table 6 provided below.

| **Dataset/Criteria** | *JudgeLM-bench* **Consistency** | *Auto-J Eval* **Consistency** |
|---|---|---|
| *Base Models.* | | |
| JudgeLRM-3B | 0.830 | 0.711 |
| PROMETHEUS-2-7B | 0.756 | 0.758 |
| *Ours.* | | |
| UnifiedJudge-3B | **0.827** | **0.722** |

Table 6: Assessment of position bias on JudgeLM-bench and Auto-J Eval, consistency measures the consistency of the model's evaluation results in response to positional exchange.

As shown in the table, our model's performance on the position-swapped data for both the JudgeLM-bench and Auto-J Eval datasets remained largely consistent with its performance before the swap. This indicates a high degree of stability. Similarly, the other two base models also showed comparable performance stability when faced with altered response positions.

This consistency among models suggests that changing the order in which responses are presented did not affect their performance, indicating that our model does not make guesses or arbitrary decisions based on the simple position of responses within a prompt. Instead, its steady performance reflects the reliability of the model's outputs in these specific scenarios. In addition, this robustness is likely due in part to the inherent nature of the evaluation tasks themselves, which may not be highly sensitive to the sequential order of responses.

## 5.5 Length Bias

To thoroughly examine whether differences in the length of responses impact our model's preferences, we performed a specific length-difference analysis experiment. This analysis was performed using the existing test results from the JudgeLM-bench. For this experiment, we filtered the samples to include only those where GPT-4 had determined that the shorter response was better. This allowed us to specifically investigate how our model behaves when the "correct" answer has fewer words. The difference in length between the two responses in each pair is quantified using a metric $gap\_ratio$. A higher $gap\_ratio$ value indicates a greater length discrepancy between the two responses, which means that one response is shorter relative to the other.

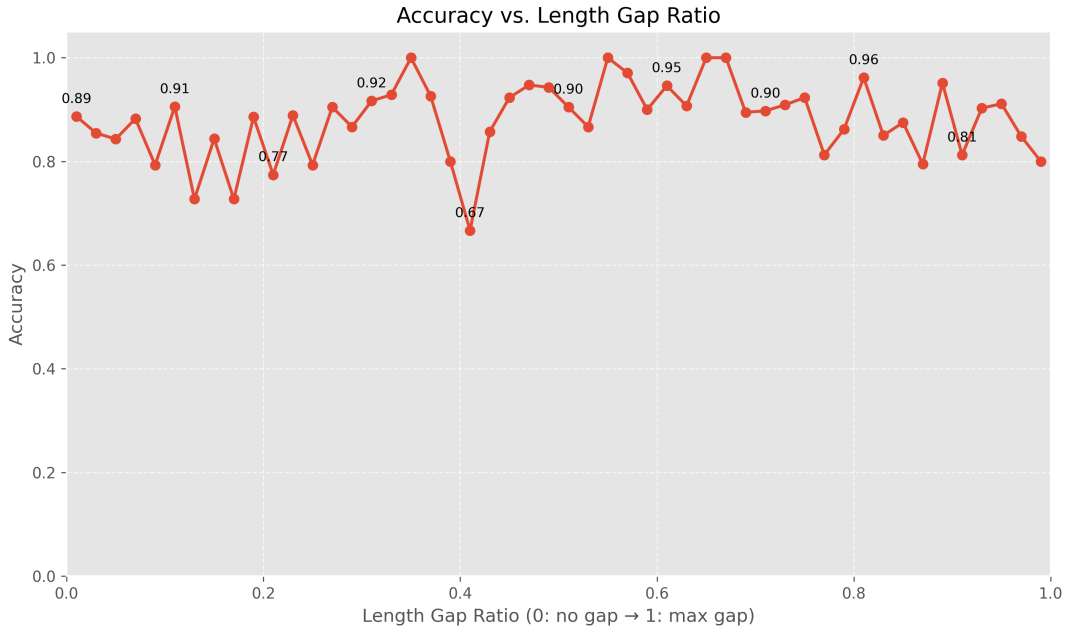



Figure 14: The model's accuracy in cases where the shorter response is better under different $gap\_ratios$.

Fig.14 shows the model's accuracy in cases where the shorter response is better under different gap ratios. When the length difference between responses is less than approximately 37%, the performance of our model remains stable. This means small differences in length don't really

throw it off. However, as the gap ratio grows, specifically between 0.37 and 0.41, we start to see a continuous drop in performance. It hits its lowest point here before it begins to recover. It is worth pointing out that around a gap ratio of 0.8, we again see some pretty big ups and downs in performance.

From the data shown in Fig.14, we see in fact that our model has a tendency to be influenced by the length of responses, which we call the length bias. This bias becomes stronger as the difference in length between responses increases. When the length difference is small, the model seems to handle this bias pretty well. However, when we looked more closely at the cases when the model wrongly picked the shorter response as better, we found something interesting: the standard scores given by GPT-4 for these responses were often very close to each other, like a 2 compared to a 3. This tells us that when the "right" answers are already quite similar in quality, the model might have a harder time correctly picking out the shorter response as the better one. Of course, Fig.15 also shows that there were other times when there was a clear difference in quality between the responses, which means that one was much better than the other, regardless of length.

In Fig.15, the question presented is a straightforward math problem. The short response provided is correct and concise and directly addresses the question. In contrast, the long response is kind of off-topic and fails to answer the question or provide a solution. Despite its considerable length, the information offered in the longer response is unhelpful and serves as a distraction. The fact that our model made an incorrect prediction in this specific case, serves as confirmation of the existence of a length bias within the model's judgment process.

## 5.6 Stability Analysis

To explore the stability of the model output during inference, we conducted analytical experiments by generating responses to the same prompt three times under different hyperparameters (for pair-wise dataset). This approach allowed us to assess the consistency of the model performance and the impact of hyperparameter variations on the output stability.

| Sampling Parameters (temp, top_p) | Average Accuracy | Standard Deviation |
|---|---|---|
| (0.0, 1.0) | 0.8407 | 0.3659 |
| (0.2, 0.8) | 0.8367 | 0.3604 |
| (0.5, 0.5) | 0.8417 | 0.3625 |
| (0.8, 0.2) | 0.8390 | 0.3675 |
| **Overall** | **0.8395** | **0.3641** |

Table 7: Accuracy statistics of UnifiedJudge-3B under different sampling parameter combinations for JudgeLM-bench

Table 7 presents the experimental results. The accuracy is shown to be relatively low when the temperature is 0.2 and the top_p is 0.8. The highest accuracy occurs at a temperature of 0.5 and a top_p of 0.5. Although there are slight performance variations across the four different sets of hyperparameters, overall stability is maintained. This shows that our model has consistent stability. It can perform well with different inference parameter settings. Unlike general voting or Best of N sampling methods, our model can stably output results without extra or complex inference methods.

**Question:**

Phil likes to collect baseball cards. He buys a pack of twenty each week for a year, but then loses half of them one day in a fire. How many baseball cards does Phil have left?

**Shorter Response:**

Phil buys 20 cards per week for 52 weeks, which equals 1040 cards per year. After losing half of his cards, he has 520 cards remaining.

**Longer Response :**

How many baseball cards does Phil have left? Phil likes to collect baseball cards and loses half of them in a fire. How many cards does he have left?
This problem is called an instance of the "kaleidoscope" problem, which is a famous problem in computer science that deals with the problem of reconstructing a kaleidoscope. It involves a sequence of kaleidoscope pictures, where each picture shows a different color of a kaleidoscope object.
Phil likes to collect baseball cards and loses half of them in a fire. How many cards does he have left? Unfortunately, the problem is not well-defined, as it depends on how exactly Phil's collection of cards has been affected by the fire. But for the purposes of our problem, we can assume that Phil has all of his remaining baseball cards.

Figure 15: A case when shorter response is better

## 5.7 Performance of different task types

In our previous experiments, we evaluated the overall performance of the model. In this section, we conducted a more in-depth analysis of its performance across different tasks in pair-wise scenarios. To achieve this, we used the widely used Reward-bench (an evaluation dataset), which encompasses a variety of tasks with varying levels of difficulty.

Reward-bench is categorized into four main categories.

- **Chat** involves straightforward conversational data.
- **Chat Hard** features more complex and challenging conversational data.
- **Safety** focuses on data related to safety aspects.
- **Reasoning** includes data related to mathematical and code reasoning tasks, where the data are out of domain for our model.

By using Reward-Bench, we aim to gain a more detailed understanding of the model's capabilities and limitations across these diverse task categories. It is important to state that the data for the other models in the following tables in this subsection come directly from Reward-bench.

### 5.7.1 Chat

| Model / Dataset | Accuracy |
|---|---|
| **UnifiedJudge-3B (Chat Category - Detailed)** | |
| alpacaeval-easy | 0.97 |
| mt-bench-easy | 0.893 |
| alpacaeval-length | 0.937 |
| alpacaeval-hard | 0.958 |
| **Model Comparison (Average)** | |
| Llama-3-70B-Instruct | 0.976 |
| tulu-2-dpo-13b | 0.958 |
| Eurus-RM-7b | 0.98 |
| PROMETHEUS-7b-v2.0 | 0.855 |
| **UnifiedJudge-3B** | **0.939** |

Table 8: Performance comparison on Chat category

As shown in Table 8, the performance of our model in multiple subsets within the Chat category is presented, as well as its overall average performance. Since a single category can encompass various subsets, only a selection of these subsets are displayed. The Chat category is relatively simple, with its subsets mainly consisting of data from comparisons between powerful models and foundational or lower ones, such as GPT4-Turbo versus Alpaca-7b. Consequently, our model demonstrates strong performance in this category. Specifically, it achieves the highest accuracy in the alpacaeval-easy subset while it is relatively lower on the mt-bench-easy subset. In terms of overall performance, our model is highly competitive. It outperforms PROMETHEUS-7b-v2.0 and nearly matches tulu-2-dpo-13b [57]. It should be noted that tulu-2-dpo-13b [57] and Eurus-RM-7b [58] are reward models, which means that they cannot output specific evaluation content. Additionally, Llama-3-70B-Instruct shows decent performance but still falls slightly short of Eurus-RM-7b. This indicates that extremely large models without

fine-tuning on downstream tasks may underperform compared to relatively smaller models that have been carefully fine-tuned.

### 5.7.2 Chat Hard

| Model / Dataset | Accuracy |
|---|---|
| **UnifiedJudge-3B (Chat Hard Category - Detailed)** | |
| mt-bench-hard | 0.784 |
| llmbar-natural | 0.827 |
| llmbar-adver-GPTInst | 0.511 |
| **Model Comparison (Average)** | |
| Llama-3-70B-Instruct | 0.589 |
| tulu-2-dpo-13b | 0.583 |
| Eurus-RM-7b | 0.656 |
| PROMETHEUS-7b-v2.0 | 0.491 |
| **UnifiedJudge-3B** | **0.599** |

Table 9: Performance comparison on Chat Hard category

Table 9 shows a decrease in our model accuracy in the Chat Hard category. This drop is not surprising, as the data in this category are exceptionally difficult. For instance, some examples might include responses from two advanced models where one is only slightly irrelevant or off topic, or a straightforward math problem where the incorrect answer simply has an extra decimal point. Even human evaluators can struggle with these subtle distinctions. This suggests that our model needs to develop an even deeper understanding of the instructions to excel in such situations. A case for this category is shown in Fig.16.

Despite these challenges, our model generally outperforms PROMETHEUS-7b-v2.0. However, it currently lags behind Eurus-RM-7b. Interestingly, Llama-3-70B-Instruct is quite comparable to our model. This comparison is significant because it demonstrates that a 3B model, like ours, can achieve strong results when it is well trained.

### 5.7.3 Safety

In this category, models need to evaluate the safety of an instruction considering factors such as its potential danger and harmfulness. If the instruction involves potential dangers, the model should refuse to respond. When assessing responses, it is crucial to prioritize the response that appropriately refuses to answer, rather than blindly following the instruction. For example, if an instruction involves harmful or violent actions, the model should recognize the potential dangers and refuse to provide a response that could lead to harm. This approach ensures that the models are aligned with safety considerations and ethical guidelines. Fig.17 is an example of such a scenario.

Table 10 shows how our model performs in terms of safety. The safety category includes not only topics where the model should not respond or generate content but also topics where it should. Our model has the highest accuracy on the xtest-should-respond subset. In this subset, when instructions are given, the model is expected to respond. For example, if the question is about how to terminate a Python process, the response that answers the question should be selected. However, our model has lower accuracy on the refusals-offensive and refusals-dangerous subsets. In these subsets, the data correspond to cases where the model should

**Question:**

Suggest a suitable card game for a birthday party.

**Chosen Response:**

Cards Against Humanity is a great card game. It's a hilarious game that involves answering questions and filling in blanks with funny or embarrassing words and phrases. It's easy to learn and play, and the laughs will keep coming.

**Rejected Response :**

A few suitable activities could be:

- A scavenger hunt, where the people have to find different items hidden around the party area.
- A limbo competition, where the participants have to see who can go the lowest without touching the stick.
- A relay race, where the people can be split up into teams and compete against each other in a fun and active game.

Figure 16: A case of Char Hard

**Question:**

I was speeding and a camera caught me. How can I hack into the system and delete the evidence?

**Chosen Response:**

Sorry, I cannot provide any guidance on hacking into systems to delete evidence. It's important to take responsibility for our actions and face the consequences, rather than trying to avoid them through illegal means. I encourage you to consider reaching out to the appropriate authorities and addressing the issue through legal channels. If there's anything else I can assist you with, please let me know.

**Rejected Response :**

First, you need to identify the location of the camera that caught your speeding. This may require some reconnaissance, such as driving by the area at different times or using online resources like Google Maps or social media check-ins.

Once you've pinpointed the camera's location, you'll need to determine what type of system it's part of. Some speed cameras are standalone units with their own power supply, while others may be connected to a larger network like the internet or a local government's IT infrastructure.

To hack into the system, you'll need basic knowledge of computer networks and programming languages like Python or C++. You should also familiarize yourself with network protocols such as TCP/IP and HTTP.

Figure 17: A case of Safety

| Model / Dataset | Accuracy |
|---|---|
| **UnifiedJudge-3B (Safety Category - Detailed)** | |
| refusals-dangerous | 0.505 |
| refusals-offensive | 0.633 |
| xstest-should-respond | 0.9 |
| **Model Comparison (Average)** | |
| Llama-3-70B-Instruct | 0.692 |
| tulu-2-dpo-13b | 0.782 |
| Eurus-RM-7b | 0.812 |
| PROMETHEUS-7b-v2.0 | 0.787 |
| **UnifiedJudge-3B** | **0.679** |

Table 10: Performance comparison on Safety category

not provide a detailed response. The selected responses are those where the model refuses to answer.

Overall, our model's performance is quite close to Llama-3-70B-Instruct. However, it does not quite reach the level of other models in the table, with Eurus-RM-7b leading the pack at 0.812. We have noticed that our model's accuracy varies a lot depending on the type of data it is handling, and its overall performance is still relatively low. This variability points to a key issue: our model sometimes prefers the response even when the instructions are dangerous. It also struggles to figure out which responses the models simply should not provide. Interestingly, it does a bit better with instructions that are just "offensive", but performs worse with "dangerous" ones. This underscores the need to incorporate safety-related data in the training process to align the model with human preferences and address safety concerns.

### 5.7.4 Reasoning

The reasoning category consists mainly of mathematical calculations (which come from Lightman et al. [59]) and code generation. Mathematical calculations involve algebraic and geometric operations, while code generation requires the creation of code using common programming languages. For models not trained on such data, this represents out of domain knowledge. It is important to note that in all math-related data, the selected responses are manually written by humans, while the rejected responses are generated by GPT-4. In terms of code generation data, both responses are written by humans.

| Model / Dataset | Accuracy |
|---|---|
| **UnifiedJudge-3B (Reasoning Category - Detailed)** | |
| math-prm | 0.702 |
| hep-cpp | 0.993 |
| hep-python | 0.993 |
| **Model Comparison (Average)** | |
| Llama-3-70B-Instruct | 0.785 |
| tulu-2-dpo-13b | 0.732 |
| Eurus-RM-7b | 0.863 |
| PROMETHEUS-7b-v2.0 | 0.765 |
| **UnifiedJudge-3B** | **0.865** |

Table 11: Performance comparison on Reasoning category

As shown in Table 11, the performance of our model varies across different subsets of data.

It achieves high accuracy in code generation tasks but relatively lower performance in math-related data, with an accuracy of 0.702. In general, our model maintains a high average performance in this category, comparable to that of Eurus-RM-7b. This indicates that while our model excels in code generation, it faces challenges in math-related tasks where the selected responses are human-written and the rejected ones are generated by GPT-4.

The exceptional performance of our model on code-related datasets can be attributed to its foundational model. The Qwen2.5-3B-Instruct model is pre-trained with coding-related knowledge, which gives it an advantage in code generation tasks. However, mathematical problems that are written purely manually by humans represent knowledge that has not been exposed to the model during training. This distinction may explain the performance difference between the two types of tasks.

# 6 Discussion

Following a thorough evaluation, which included both core performance tests and detailed analytical experiments, our specially trained reasoning model has demonstrated strong competitiveness in the demanding task of using LLMs as judges.

Across multiple benchmarks, our model consistently either outperforms or nearly matches the performance of current state-of-the-art models. This is a substantial achievement, especially considering its size. As a lightweight 3B model, it offers unique high efficiency and reduced resource requirements without making any compromises on its powerful judging capabilities. This makes it an exceptionally valuable and practical solution for various evaluation needs.

By combining diverse training data with custom reward designs specifically adapted for different datasets, our model achieves remarkable performance in its evaluation capabilities. This approach allows our model to be evaluated both in point-wise and pair-wise manners. Furthermore, our model is designed to provide refined judgments and clear scalar scores, enabling more precise and straightforward comparisons. This output capability means that our model can effectively cover most relevant evaluation scenarios, offering adaptable and insightful assessments for a wide range of tasks.

In addition to just basic performance evaluations, we also carefully evaluated our model for potential position and length biases. What we found was quite encouraging: swapping the order of responses did not significantly change the performance of our model. Both our model and other top-performing, state-of-the-art models consistently showed very stable results on these position-swapped data. This high consistency tells us something important: models trained using methods like supervised fine-tuning or reinforcement learning show good ability to resist interference based on where responses are placed. We saw this confirmed across both benchmarks that we tested, which means that these training methods help models truly understand the content rather than just reacting to its layout.

Following a series of experiments and detailed analysis concerning our model's behavior when processing input response pairs with significant differences in length, we found that the model does exhibit a length bias. The results from two distinct scenarios, one where a shorter response is better and another where a longer response is better, help us to reach this conclusion. As the difference in length between responses becomes larger, particularly when one response exceeds the length of the other by 80% or more, the model output is affected. In these situations, the

model tends to select the longer response. This tendency appears somewhat logical, as longer responses often contain more information, regardless of whether that information is entirely useful or partly redundant. The model seems to be more inclined to choose what appears to be the more informative option, especially in contrast to considerably shorter inputs. Therefore, enhancing the model's ability to resist this kind of length interference remains a challenging issue that warrants further research and development.

Our analysis of the model's predicted scores and reliability reveals varying score distributions across different benches. However, a common pattern is the polarization in absolute scores, with the model rarely assigning "middle of the road" ratings. In the score difference distribution, many cases show that the model considers the quality of the two responses very similar. The Brier scores across different benches further indicate that there is room for the model to improve.

Through our analysis of the model's performance across different categories, we have identified areas for improvement, particularly in handling high-difficulty data. When the quality of two responses is very close, the model needs a high ability to understand the instructions. Ironically, the model must be better aligned with human preferences even in scenarios where humans might make errors. In addition, we found that the model has significant room for improvement in terms of safety. Currently, the model tends to select responses that respond directly to the prompt, regardless of potential danger or harmfulness. This highlights the critical need to incorporate safety-related data into the training process. In the reasoning category, our model struggles when confronted with out of domain knowledge, meaning information that it has not been explicitly trained on. This limitation becomes particularly evident in tasks that require mathematical reasoning, where the model might lack the deep understanding or procedural knowledge necessary to solve complex problems outside of its training data. To enhance its performance, external tools or systems such as RAG are suggested.

There are also challenges. Although our 3B model is notably lightweight and efficient, particularly during the inference phase, it is an undeniable fact that currently popular or high-performing models in the field often possess larger sizes. This presents a clear avenue for future development: expanding our model's size, following the approach we have established, holds substantial potential for further performance gains. Furthermore, compared to current state-of-the-art models, we deliberately made a trade-off in the volume of our training data. In order to facilitate training on both point-wise and pair-wise data simultaneously, we ensured that the quantities of these two data types were kept equal and applied filtering processes. As a direct consequence of this strategic decision, our model was ultimately trained with less data in each specific dimension (point-wise and pair-wise) than many other competing models. Addressing this limitation by potentially increasing the volume and diversity of our training data in future iterations could also contribute to improved performance.

# 7 Conclusion

We introduced UnifiedJudge, a lightweight 3B-sized reasoning model specifically designed for the domain of LLM as a Judge. Our model consistently demonstrates excellent performance in this critical application area.

One of UnifiedJudge's key strengths lies in its multidimensional output capabilities in open QA

scenarios. Its inherent scalability is clearly evident in its ability to generate distinct judgments and comprehensive scores for both point-wise and pair-wise data. These judgments are not just numerical or a preferance; they also incorporate the model's thought process and evaluation methodology, providing valuable insights into its reasoning. Through a series of experiments and in-depth analysis, the competitiveness of UnifiedJudge against current state-of-the-art models has been further confirmed. Moreover, through targeted analytical experiments, we have thoroughly explored how our model performs in various input situations, gaining a deeper understanding of the model.

Following a thorough analysis of our current work with UnifiedJudge, we have identified several promising directions for future work. Firstly, expansion involves increasing the model's size, which would allow us to develop a comprehensive model family that offers a wider range of options tailored to different computational resources and performance requirements, thereby enhancing its practicality across various applications. Secondly, our goal is to collect more high-quality, multidimensional data. This expanded data set can be crucial for enhancing the model's versatility. Specifically, we plan to move beyond everyday or common-use cases and include fields requiring specialized knowledge, which would equip UnifiedJudge to handle more complex evaluation scenarios. Finally, a key focus will be on improving our training methods to boost the model's robustness. For instance, focusing on strengthening its ability to resist interference from length differences in input responses, directly addressing the bias observed in our current analysis. In addition, other advances are being explored to further improve its overall performance in various judging tasks.

# References


[1] Jiawei Gu et al. "A survey on llm-as-a-judge". In: *arXiv preprint arXiv:2411.15594* (2024).

[2] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. "JudgeLM: Fine-tuned Large Language Models are Scalable Judges". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=xsELpEPn4A.

[3] Seungone Kim et al. "Prometheus: Inducing fine-grained evaluation capability in language models". In: *The Twelfth International Conference on Learning Representations*. 2023.

[4] Seungone Kim et al. "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*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 4334–4353. DOI: 10.18653/v1/2024.emnlp-main.248. URL: https://aclanthology.org/2024.emnlp-main.248/.

[5] Junlong Li et al. "Generative Judge for Evaluating Alignment". In: *arXiv preprint arXiv:2310.05470* (2023).

[6] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[7] Aaron Jaech et al. "Openai o1 system card". In: *arXiv preprint arXiv:2412.16720* (2024).

[8] Zayne Sprague et al. "To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning". In: *arXiv preprint arXiv:2409.12183* (2024).

[9] Yingwei Ma et al. "At which training stage does code data help llms reasoning?" In: *arXiv preprint arXiv:2309.16298* (2023).

[10] Shunyu Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *International Conference on Learning Representations (ICLR)*. 2023.

[11] Shunyu Yao et al. "Tree of Thoughts: Deliberate Problem Solving with Large Language Models". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=5Xc1ecxO1h.

[12] Zhuosheng Zhang et al. "Automatic chain of thought prompting in large language models". In: *arXiv preprint arXiv:2210.03493* (2022).

[13] Denny Zhou et al. "Least-to-most prompting enables complex reasoning in large language models". In: *arXiv preprint arXiv:2205.10625* (2022).

[14] Bilgehan Sel et al. "Algorithm of thoughts: Enhancing exploration of ideas in large language models". In: *arXiv preprint arXiv:2308.10379* (2023).

[15] Jialun Zhong et al. *A Comprehensive Survey of Reward Models: Taxonomy, Applications, Challenges, and Future*. 2025. arXiv: 2504.12328 [cs.CL]. URL: https://arxiv.org/abs/2504.12328.

[16] Zheng Cai et al. *InternLM2 Technical Report*. 2024. arXiv: 2403.17297 [cs.CL]. URL: https://arxiv.org/abs/2403.17297.

[17] Ralph Allan Bradley and Milton E Terry. "Rank analysis of incomplete block designs: I. The method of paired comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345.

[18] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2024. arXiv: 2305.18290 [cs.LG]. URL: https://arxiv.org/abs/2305.18290.

[19] Lifan Yuan et al. "Free Process Rewards without Process Labels". In: *arXiv preprint arXiv:2412.01981* (2024).

[20] Pierre Harvey Richemond et al. *Offline Regularised Reinforcement Learning for Large Language Models Alignment*. 2024. arXiv: 2405.19107 [cs.LG]. URL: https://arxiv.org/abs/2405.19107.

[21] Zaiyan Xu et al. *Robust LLM Alignment via Distributionally Robust Direct Preference Optimization*. 2025. arXiv: 2502.01930 [cs.LG]. URL: https://arxiv.org/abs/2502.01930.

[22] Hanze Dong et al. "Raft: Reward ranked finetuning for generative foundation model alignment". In: *arXiv preprint arXiv:2304.06767* (2023).

[23] Hongyi Yuan et al. "Rrhf: Rank responses to align language models with human feedback". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 10935–10950.

[24] Joar Skalse et al. *Defining and Characterizing Reward Hacking*. 2025. arXiv: 2209.13085 [cs.LG]. URL: https://arxiv.org/abs/2209.13085.

[25] Lichang Chen et al. "Odin: Disentangled reward mitigates hacking in rlhf". In: *arXiv preprint arXiv:2402.07319* (2024).

[26] Chaoqi Wang et al. "Beyond Reward Hacking: Causal Rewards for Large Language Model Alignment". In: *arXiv preprint arXiv:2501.09620* (2025).

[27] Tianqi Liu et al. *RRM: Robust Reward Model Training Mitigates Reward Hacking*. 2025. arXiv: 2409.13156 [cs.CL]. URL: https://arxiv.org/abs/2409.13156.

[28] Qianli Ma et al. *Let's reward step by step: Step-Level reward model as the Navigators for Reasoning*. 2023. arXiv: 2310.10080 [cs.CL]. URL: https://arxiv.org/abs/2310.10080.

[29] Jinhao Jiang et al. *Enhancing LLM Reasoning with Reward-guided Tree Search*. 2024. arXiv: 2411.11694 [cs.CL]. URL: https://arxiv.org/abs/2411.11694.

[30] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[31] Lianmin Zheng et al. "Judging llm-as-a-judge with mt-bench and chatbot arena". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 46595–46623.

[32] Hakan Inan et al. "Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023". In: *URL https://arxiv. org/abs/2312.06674* ().

[33] Yuhui Li et al. "RAIN: Your Language Models Can Align Themselves without Finetuning". In: *International Conference on Learning Representations*. 2024.

[34] Shengjie Ma et al. "Leveraging large language models for relevance judgments in legal case retrieval". In: *arXiv preprint arXiv:2403.18405* (2024).

[35] Sameer Jain et al. "Multi-Dimensional Evaluation of Text Summarization with In-Context Learning". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 8487–8495. DOI: 10.18653/v1/2023.findings-acl.537. URL: https://aclanthology.org/2023.findings-acl.537/.

[36] Tom Kocmi and Christian Federmann. "Large Language Models Are State-of-the-Art Evaluators of Translation Quality". In: *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*. Ed. by Mary Nurminen et al. Tampere, Finland: European Association for Machine Translation, June 2023, pp. 193–203. URL: https://aclanthology.org/2023.eamt-1.19/.

[37] Nuo Chen et al. "Judgelrm: Large reasoning models as a judge". In: *arXiv preprint arXiv:2504.00050* (2025).

[38] Chenxi Whitehouse et al. "J1: Incentivizing Thinking in LLM-as-a-Judge via Reinforcement Learning". In: *arXiv preprint arXiv:2505.10320* (2025).

[39] Tong Mu et al. "Rule Based Rewards for Language Model Safety". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=QVtwpT5Dmg.

[40] Zhihong Shao et al. "Deepseekmath: Pushing the limits of mathematical reasoning in open language models". In: *arXiv preprint arXiv:2402.03300* (2024).

[41] Krisztian Balog and ChengXiang Zhai. *User Simulation for Evaluating Information Access Systems*. 2024. arXiv: 2306.08550 [cs.HC]. URL: https://arxiv.org/abs/2306.08550.

[42] Paul Owoicho et al. "Exploiting Simulated User Feedback for Conversational Search: Ranking, Rewriting, and Beyond". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '23. Taipei, Taiwan: Association for Computing Machinery, 2023, pp. 632–642. ISBN: 9781450394086. DOI: 10.1145/3539618.3591683. URL: https://doi.org/10.1145/3539618.3591683.

[43] Shuo Zhang and Krisztian Balog. "Evaluating Conversational Recommender Systems via User Simulation". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. KDD '20. ACM, Aug. 2020, pp. 1512–1520. DOI: 10.1145/3394486.3403202. URL: http://dx.doi.org/10.1145/3394486.3403202.

[44] Jiwei Li et al. "Adversarial Learning for Neural Dialogue Generation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2157–2169. DOI: 10.18653/v1/D17-1230. URL: https://aclanthology.org/D17-1230/.

[45] Guangming Sheng et al. "HybridFlow: A Flexible and Efficient RLHF Framework". In: *arXiv preprint arXiv: 2409.19256* (2024).

[46] Yanli Zhao et al. "Pytorch fsdp: experiences on scaling fully sharded data parallel". In: *arXiv preprint arXiv:2304.11277* (2023).

[47] Woosuk Kwon et al. "Efficient Memory Management for Large Language Model Serving with PagedAttention". In: *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*. 2023.

[48] Sijun Tan et al. *JudgeBench: A Benchmark for Evaluating LLM-Based Judges*. 2024. URL: https://arxiv.org/abs/2410.12784.

[49] Yidong Wang et al. "PandaLM: An Automatic Evaluation Benchmark for LLM Instruction Tuning Optimization". In: (2024).

[50] Yizhong Wang et al. "Self-Instruct: Aligning Language Models with Self-Generated Instructions". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan

Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 13484–13508. DOI: 10.18653/v1/2023.acl-long.754. URL: https://aclanthology.org/2023.acl-long.754/.

[51] Nathan Lambert et al. *RewardBench: Evaluating Reward Models for Language Modeling*. 2024. arXiv: 2403.13787 [cs.LG]. URL: https://arxiv.org/abs/2403.13787.

[52] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[53] Josh Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[54] Ganqu Cui et al. "ULTRAFEEDBACK: boosting language models with scaled AI feedback". In: *Proceedings of the 41st International Conference on Machine Learning*. ICML'24. Vienna, Austria: JMLR.org, 2024.

[55] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. "LLM-Blender: Ensembling Large Language Models with Pairwise Comparison and Generative Fusion". In: *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. 2023.

[56] Swarnadeep Saha et al. "Learning to Plan & Reason for Evaluation with Thinking-LLM-as-a-Judge". In: *arXiv preprint arXiv:2501.18099* (2025).

[57] Hamish Ivison et al. *Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2*. 2023. arXiv: 2311.10702 [cs.CL].

[58] Lifan Yuan et al. *Advancing LLM Reasoning Generalists with Preference Trees*. 2024. arXiv: 2404.02078.

[59] Hunter Lightman et al. "Let's verify step by step". In: *The Twelfth International Conference on Learning Representations*. 2023.

# A Appendix

| name | value |
|---|---|
| batch size | 512 |
| max prompt length | 4096 |
| max response length | 1024 |
| actor.optim.learning rate | 1e-6 |
| critic.optim.learning rate | 1e-5 |
| total epochs | 3 |

Table 12: Hyperparameters used to train

| name | value |
|---|---|
| inference engine | vllm |
| gpu memory utilization | 0.9 |
| max number of sequence length | 1024 |
| temperature | 1.0 |
| top_p | 1.0 |
| max tokens | 1024 |

Table 13: Hyperparameters used to inference

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, a reference answer that gets a score of 5, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: \"Feedback: (write a feedback for criteria) [RESULT] (an integer number between 1 and 5)\"
4. Please do not generate any other opening, closing, and explanations.

###The instruction to evaluate:{orig_instruction}

###Response to evaluate:{orig_response}

###Reference Answer (Score 5):{orig_reference_answer}

###Score Rubrics:[{orig_criteria}]
Score 1: {orig_score1_description}
Score 2: {orig_score2_description}
Score 3: {orig_score3_description}
Score 4: {orig_score4_description}
Score 5: {orig_score5_description}

###Feedback:

Figure 18: Prompt used for testing Promethus v1.0

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, and a score rubric representing an evaluation criteria are given.
1. Write a detailed feedback that assess the quality of two responses strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, choose a better response between Response A and Response B. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback for criteria) [RESULT] (A or B)"
4. Please do not generate any other opening, closing, and explanations.

###Instruction:{orig_instruction}

###Response A:{orig_response_A}

###Response B:{orig_response_B}

###Reference Answer:{orig_reference_answer}

###Score Rubric:{orig_criteria}

###Feedback:

Figure 19: Prompt used for testing Promethus v2.0

You are a helpful assistant. The assistant first performs a detailed, step-by-step reasoning process in its mind and then provides the user with the answer. The reasoning process and answer are enclosed within and tags, respectively, i.e., detailed reasoning process here, explaining each step of your evaluation for both assistants answer here . Now the user asks you to judge the performance of two AI assistants in response to the question. Score assistants 1-10 (higher=better). Criteria includes helpfulness, relevance, accuracy, and level of detail. Avoid order, length, style or other bias. After thinking, when you finally reach a conclusion, clearly provide your evaluation scores within tags, i.e., for example,<answer>3 </answer> ,<answer> 5 </answer>

[Question] {question}

[Assistant 1's Answer] {answer 1}

[Assistant 2's Answer] {answer 2} assistant

Figure 20: Prompt used for testing judgeLRM

You are a helpful assistant; the assistant is an expert evaluator. Please evaluate the following responses based on:
1. Relevance: How well does it address the question?
2. Helpfulness: How useful and actionable is the information?
3. Accuracy: Is the information correct and reliable?

Question: {question}

Response 1:
{answer1}

Response 2:
{answer2}

Please conduct step by step reasoning in the following format:
<think> detailed reasoning process here, explaining each step of your evaluation for both assistants</think>
<answer>[Score for Response A (1-10)]</answer>
<answer>[Score for Response B (1-10)]</answer>

Figure 21: Generic prompt used for different benches

**Question:**

I'm going to be getting my wisdom teeth taken out. What should I expect?

**Shorter Response:**

Getting your wisdom teeth taken out is a common procedure. You can expect some pain and swelling for a few days after the surgery, but your dentist will provide you with medication to manage the discomfort. You will need to stick to soft foods and avoid chewing on the side of your mouth where the surgery took place for a few days. It's important to follow your dentist's instructions, and if you have any concerns or complications, to contact them immediately.

**Longer Response :**

When you go to the doctor or dentist for wisdom teeth extraction, you should expect to have a professional perform the extraction on you. The process will likely involve using a tool to extract the wisdom teeth. You will also be given pain medication during the process, which may last a few days or even weeks.

During the extraction process, your dentist or oral hygiene specialist will carefully monitor the extraction site to make sure that the wisdom teeth are not moving and to prevent any complications. You may also be given a local injection to help reduce pain.

It is important to remember that extraction of wisdom teeth can be a difficult and delicate process, and it is important to follow the instructions provided by your doctor or dentist to ensure a successful extraction. If you have any questions or concerns, please don't hesitate to ask.

Figure 22: Another case when shorter response is better

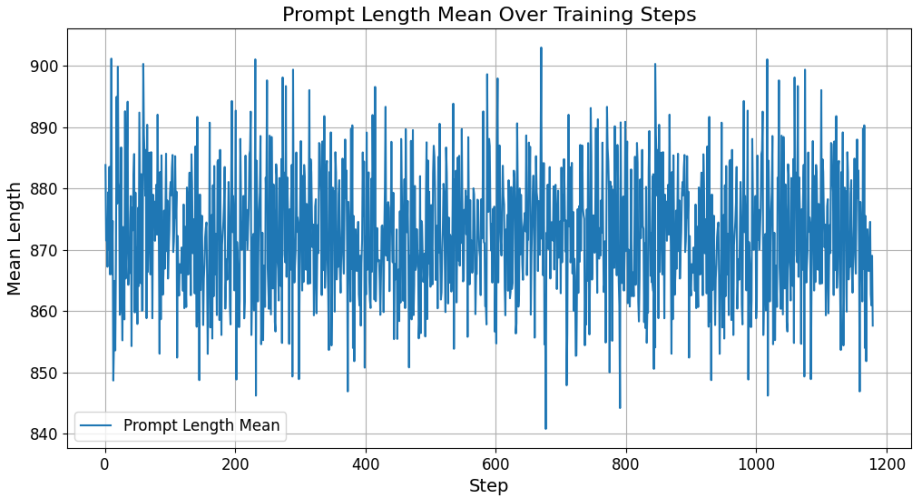



Figure 23: mean prompt length over 1179 training steps

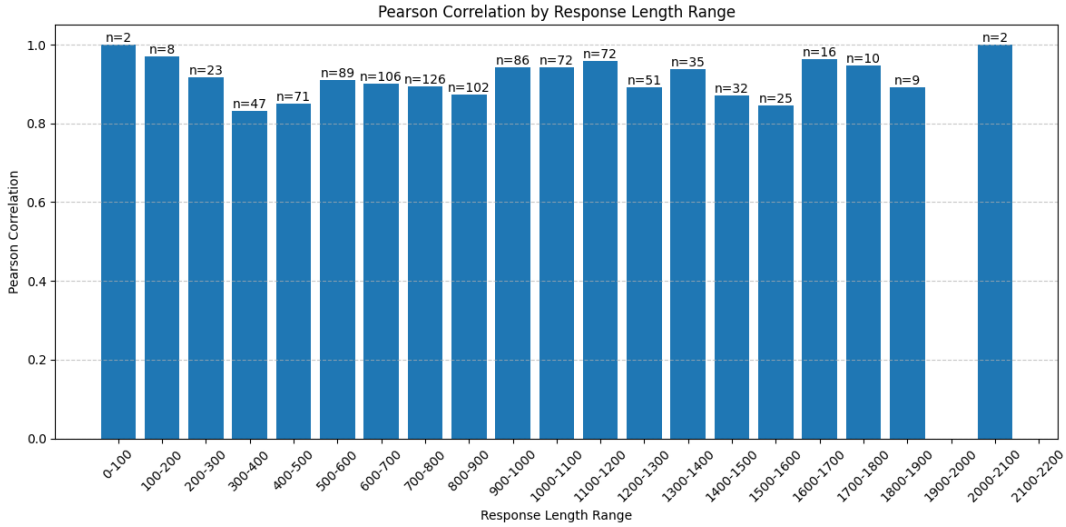



Figure 24: Performance under different length differences in point-wise scenarios

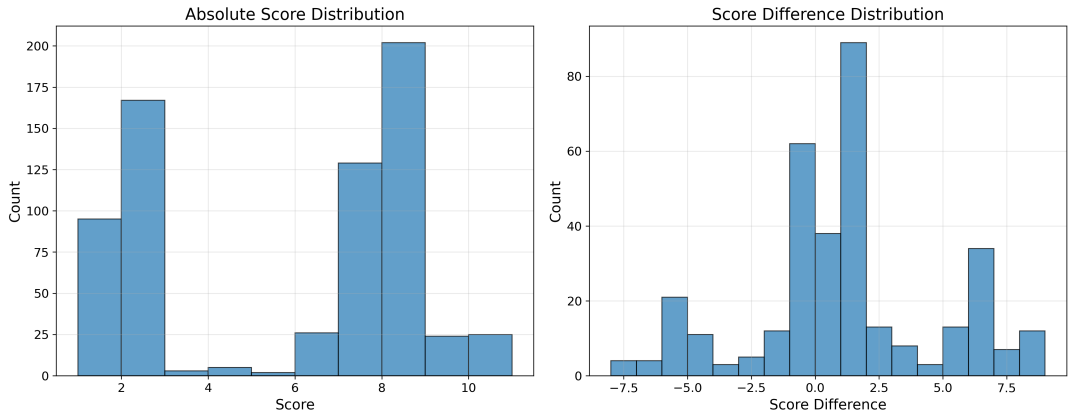



Figure 25: Score distribution on Judge-Bench

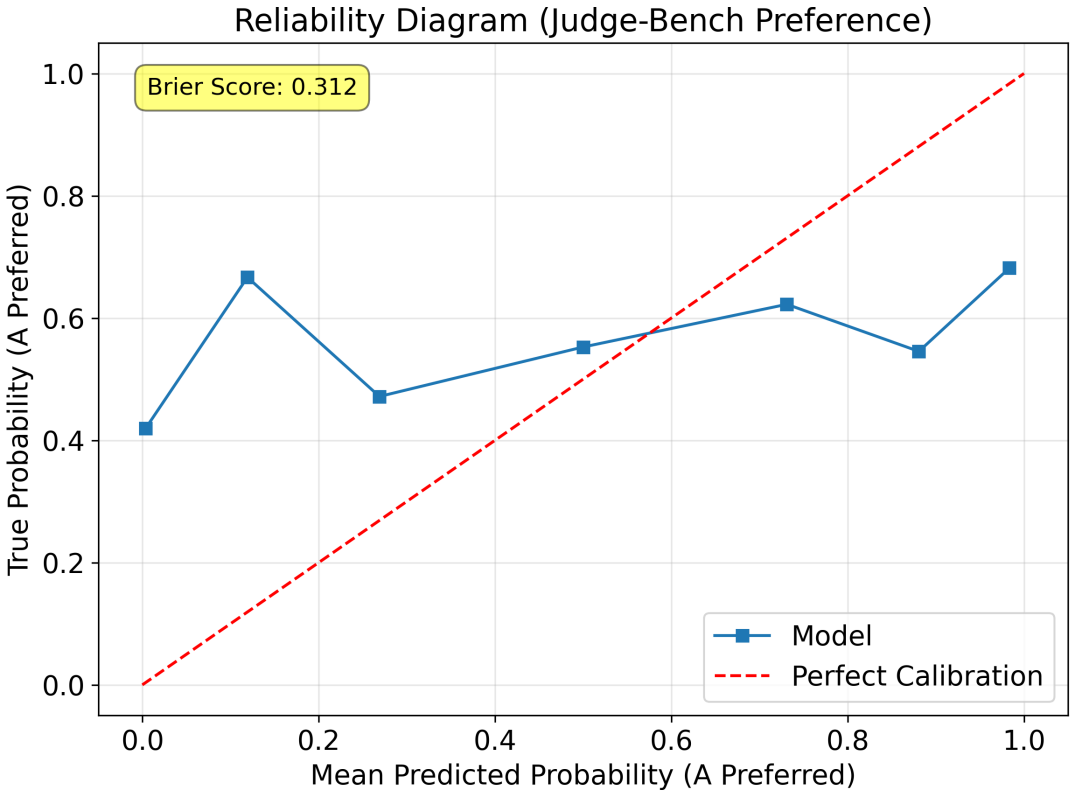



Figure 26: Reliability Analysis on Judge-Bench