# Empirical Studies on the Limitations of Direct Preference Optimization, and a Possible Quick Fix

**Jiarui Yao** [*1] **Yong Lin** [*2] **Tong Zhang** [1]

## Abstract

Reward models play a crucial role in the post-training of large language models (LLMs). While explicit reward models are widely used, implicit approaches like Direct Preference Optimization (DPO (Rafailov et al., 2023)) offer an alternative. However, implicit models often exhibit weaker generalization and performance (Lin et al., 2024). In this work, we investigate why DPO underperforms compared to explicit reward models. We first demonstrate that generating high-quality answers is generally more difficult than discriminating between good and bad answers, providing an intuitive explanation for DPO's weaker generalization, since it directly learns generation rather than discrimination. Further, we show that the DPO objective requires greater model capacity to fit effectively, suggesting that the learning task itself is more challenging. Crucially, because DPO operates by directly optimizing token-level probabilities, the combination of large vocabulary sizes and long-tail token distributions leads to inefficient learning dynamics that ultimately degrade model performance. To address this, we propose a simple modification to DPO's formulation: removing the (log)-softmax function, which improves the implicit reward model's effectiveness. This adjustment can enhance algorithms like PRIME, which rely on implicit reward modeling.

## 1. Introduction and Background

Reward models play a crucial role in the post-training stage of Large Language Models, especially in Reinforcement Learning from Human Feedback (RLHF) (Bai et al., 2022; Ouyang et al., 2022), etc. To obtain an accurate approximation of the underlying rewards, the technique of using another LLM to serve as the reward model has been investigated extensively during the research topic in recent days. Implicit reward models like Rafailov et al. (2023); Ethayarajh et al. (2024); Meng et al. (2024) provide a convenient manner to directly optimize the LLMs instead of introducing an extra projection between hidden embedding and the actual rewards, further eliminating the need to optimize another reward model in parallel. However, such implicit reward models usually introduce extra performance overheads compared to directly utilizing explicit reward models. Xu et al. (2024); Ivison et al. (2024); Wang et al. (2024) have studied the performance comparison between DPO and other explicit reward models based RLHF methods, which also point out the potential challenges of DPO to some extent. In this work, we systematically compare DPO and explicit reward models, analyze the reasons behind the poor performance of DPO compared to the explicit reward models, and propose a simple yet effective "quick fix" to further improve the performance of DPO by removing the "harmful" log-softmax function.

### Related Work

Reward models play a crucial role in the post-training stage of LLMs, especially for reinforcement learning, which usually involves the critic models (Konda & Tsitsiklis, 1999; Schulman et al., 2017). This could be deployed in various downstream tasks, particularly for reasoning (Guo et al., 2025; Yao et al., 2025; Xiong et al., 2025) and function calling (Jin et al., 2025), etc. Moreover, a wide range of works (Luo et al., 2025) have revealed the different properties of reward models and how to construct better ones from different perspectives.

From the way reward models assign the rewards, we could categorize them into the following several classes (Zhong et al., 2025). First is the most commonly used one, explicit reward models, which directly output a real number (usually through appending a projection layer after the hidden states and normalized to the range [0,1]), and in some literature, they are also known as discriminative reward models. Generative reward models (Ye et al., 2024; Zheng et al., 2023;

[*]Equal contribution [1]University of Illinois Urbana-Champaign [2]Princeton Language and Intelligence, Princeton University. Correspondence to: Jiarui Yao <jiarui14@illinois.edu>, Yong Lin <yong.lin@princeton.edu>.

ICML 2025 Workshop on Models of Human Feedback for AI Alignment, Vancouver, Canada. PMLR 267, 2025. 

Li et al., 2023) output a paragraph of text, and they usually provide a more detailed explanation for the criteria adopted in the reward assignment. In comparison, implicit reward models, including Rafailov et al. (2023), Ethayarajh et al. (2024), Meng et al. (2024), Zhao et al. (2023), Rafailov et al. (2024), directly integrate the rewards into the "generation" process of a given prompt, and do not introduce any additional components like explicit reward models, nor need to generate extra output like generative reward models.

## 2. Preliminary Observations and Analyses

### 2.1. Intuition and Preliminary Validation

Recall that DPO uses $r_{\text{dpo}}(x, y) = \beta \log \frac{\pi_{\text{dpo}}(y|x)}{\pi_{\text{ref}}(y|x)}$ to parameterize the reward model implicitly. Hopefully, we can train a good policy model $\pi_{\text{dpo}}$ which also elicits a good implicit reward model. However, we argue that generation is typically harder than discrimination. Specifically, let us take a math problem from the MATH (Hendrycks et al., 2021) bench as an example. For math problems, the input prompt is just the problem itself.

**Math Example**

When rolling a certain unfair six-sided die with faces numbered 1, 2, 3, 4, 5, and 6, the probability of obtaining face $F$ is greater than $1/6$, the probability of obtaining the face opposite face $F$ is less than $1/6$, the probability of obtaining each of the other faces is $1/6$, and the sum of the numbers on each pair of opposite faces is 7. When two such dice are rolled, the probability of obtaining a sum of 7 is $\frac{47}{288}$. Given that the probability of obtaining face $F$ is $m/n$, where $m$ and $n$ are relatively prime positive integers, find $m + n$.

The generation and discrimination tasks are distinct in the following sense.

- Generation task: solve the problem and generate a potential answer to the problem. The accuracy of generation is defined as whether the problem is solved in the math task, or whether the instruction is strictly followed in the instruction following evaluation task.
- Discrimination task: given the description of the math problem and the generated answer from GPT, we put them together and again request GPT to decide whether the generated answer indeed solves the problem or not. The accuracy of the discrimination task is based on whether the model's decision about whether the given answer solves the problem, or whether the model's response satisfies the requirements mentioned in the instructions.

To verify this argument, we conduct a quantitative experiment with the following two tasks. Specifically, we used the datasets IFEval (Zhou et al., 2023) and lighteval/MATH (Huggingface) with two sub-classes from each dataset. For each sub-class, we sample twenty prompts and use the proprietary model GPT as the evaluation language model. For the IFEval dataset, we used "keywords:letter_frequency" and "detectable_content:number_placeholders" as the sub-classes where the accuracy increases from 45% to 80% and 65% to 85% respectively. For the lighteval/MATH dataset, we used "Geometry, Level 5" and "Counting & Probability, Level 5" as the sub-classes, where the accuracy increases from 25% to 85% and 50% to 75% respectively. These results demonstrate that generation tasks are more challenging than discrimination tasks.

| | Instruction Following | Math |
|---|---|---|
| Generation | 55% | 37.5% |
| Discrimination | 82.5% | 80.0% |

*Table 1.* A case study on the difficulty of generation and discrimination.

### 2.2. Model Capacity

To further investigate the underlying reason from the perspective of training dynamics, we try to investigate the size of model capacity that is needed to fit the data for both DPO and explicit reward models.

Recall that we use the same base model for both DPO and the explicit reward model. Ignoring the linear head of the explicit reward model, we approximately regard the number of parameters of it to be close to that of DPO. Let's use $\theta_0$ to denote the initial parameter of the base model and use $\theta$ to denote the model parameter after fine-tuning, e.g., $\theta$ could be either the DPO parameter $\theta_{\text{dpo}}$ or the reward model's parameter $\theta_{\text{rm}}$. So intuitively, if the task is harder for the $\theta_0$, it may need more effort to tune $\theta_0$ to fit such a task. Then, it would result in a larger distance between as $|\theta - \theta_0|$. We use the rank of LoRA (Hu et al., 2022) adapters as a proxy for the distance of model parameters before and after finetuning. Detailed results could be found in Section 3.2.

### 2.3. Explanation

The explicit reward model could be regarded as a projector between the embedding space and the real number, i.e., a mapping $f : \mathbb{R}^d \to \mathbb{R}$, where $d$ is the hidden dimension of the embedding layer of an LLM. In contrast, the direct preference optimization (DPO) (Rafailov et al., 2023) manner could be regarded as an implicit reward, where we directly optimize the policy model itself. The training objective of

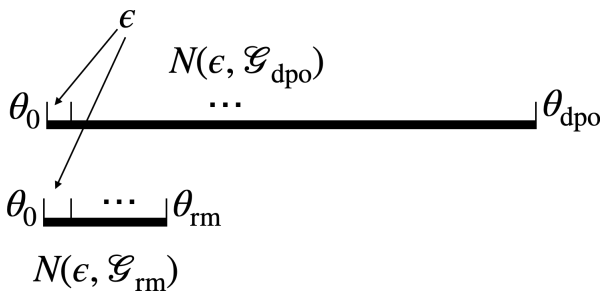


*Figure 1.* Illustration of model space of DPO and RM.

DPO could be formulated as the following equation,

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\log\sigma\Big(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_l|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\Big)\Bigg]. \quad (2.1)$$

From the above formulation, we could regard the optimization objective of DPO as a much larger network, $[T] \times (\mathbb{R}^d \times \mathbb{R})$, standing for $T$ projection layers from the hidden embedding $\mathbb{R}^d$ to a real number $\mathbb{R}$, where $T$ is the total vocabulary size. Therefore, if the backbone model for the explicit reward model is frozen during the training stage, only the projection layer, with a size of $T \times 1$ parameters, will be optimized, which is much smaller than the effective parameters during DPO training. We observe that only a small fraction of tokens indeed appear in the training corpus, and if we increase the threshold for counting the frequency of different tokens, even a smaller portion of tokens appear in the training dataset, compared to the hundreds of thousands of tokens utilized by the model in total.

For Ultrafeedback (Cui et al., 2023), we downsample twenty thousand samples from the training partition, and compute the frequency distribution of different tokens based on the subset, for which the results are shown in Figure 2. We adopt gemma-2b-it (Team et al., 2024) as the base model, of which the vocabulary size is 256000. For the twenty thousand subset, including both the prompts and responses, the total number of training tokens is 3328491, spanning 49373 distinct tokens. If we only consider the responses where the losses could be calculated on, the total number of training tokens becomes 2977762, spanning 46509 distinct tokens.

## 3. Experiments and Results

In this section, we first give some examples that demonstrate the different levels of difficulties of generation and discrimination tasks from the math reasoning area, which have verifiable answers. Then we display the empirical results in several experimental settings and provide relevant analyses according to the results.

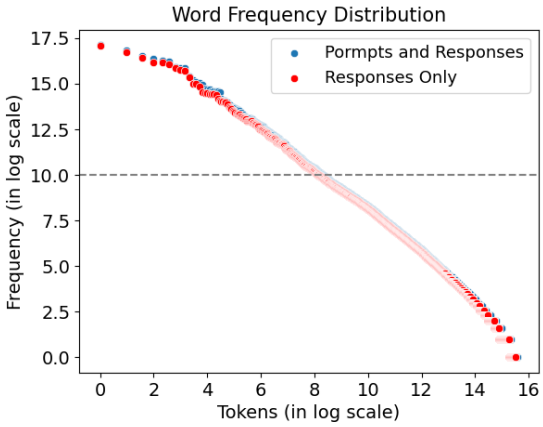


*Figure 2.* Frequency of different tokens in the Ultrafeedback (Cui et al., 2023) training subset. The x and y axis are in log 2 scale. From the figure, we could see that only about $2^8 = 256$ distinct tokens have appeared more than $2^{10} = 1024$ times in the training subset, which means that a huge portion of training tokens remain a very low frequency of appearance.

### 3.1. Training Environments Setup

For the supervised fine-tuning (SFT) conducted on the Ultrafeedback (Cui et al., 2023) subset, we train the model with 4 Nvidia A6000 GPUs, with a batch size of 128 and a learning rate of 1e-6. For the RL training under PRIME (Cui et al., 2025) setting, we use 4 Nvidia H100 GPUs in total, and use the original hyperparameters.

### 3.2. Comparison between Explicit Reward Models and DPO Using LoRA

LoRA (Hu et al., 2022) is a parameter-efficient fine-tuning (PEFT) technique that decomposes the parameters that need to be optimized into the product of two low-rank matrices. This helps decrease the GPU memory and compute resources requirements significantly when the LoRA rank is small enough.

Table 2 summarizes the results of applying LoRA to both explicit reward models and DPO in the supervised fine-tuning setting, from which we could conclude that explicit reward models always outperform DPO across different ranks for LoRA. This indicates that DPO suffer from performance drop not only on out-of-distribution domains as shown in the previous work Lin et al. (2024), but also on in-distribution domains. Therefore, to understand the causes of such a gap between these two kinds of reward models, we try to peel each affecting factor that distinguishes DPO from explicit reward models, as formulated in Equation 2.1. Actually we could regard the logits from LLMs $o_w^i$ or $o_l^i$, where $w$ and $l$ stand for the winning and losing sample in the preference dataset, and $i$ stands for the $i$-th token in the sequence, as an

| Model | Split | $r = 4$ | $r = 8$ | $r = 16$ | $r = 32$ | $r = 64$ | $r = 128$ |
|---|---|---|---|---|---|---|---|
| Explicit Reward Model | Train | 0.7393 | 0.7585 | 0.7702 | 0.7741 | 0.7754 | 0.7836 |
| | Test | **0.7387** | **0.7511** | **0.7628** | **0.7678** | **0.7683** | **0.7787** |
| DPO | Train | 0.7134 | 0.6645 | 0.7230 | 0.7398 | 0.7794 | 0.7900 |
| | Test | 0.7080 | 0.7074 | 0.7440 | 0.7467 | 0.7567 | 0.7707 |

*Table 2.* Empirical results from using LoRA (Hu et al., 2022) as the trainable adaptor to investigate how the number of trainable parameters would impact the performance of explicit reward models and DPO models.

| Heads Number | 1 | 2 | 4 | 8 | 16 | 128 | 1024 | 4096 | 16384 | 65536 |
|---|---|---|---|---|---|---|---|---|---|---|
| Multi-Head | **0.7575** | **0.7440** | **0.7535** | **0.7410** | **0.7555** | **0.7756** | **0.7796** | **0.7676** | **0.7605** | **0.7495** |
| Softmax | 0.4076 | 0.6596 | 0.6918 | 0.6998 | 0.7053 | 0.6998 | 0.6928 | 0.6923 | 0.6682 | 0.6511 |
| Log Softmax | 0.5000 | 0.7380 | 0.7334 | 0.7354 | 0.7289 | 0.7314 | 0.7505 | 0.7299 | 0.7118 | 0.6933 |

*Table 3.* The performance trend with respect to the number of heads for the multi-head only model, multi-head plus softmax model, and the multi-head plus log softmax model. From the results, we could see that softmax itself does great harm to the model's performance, while log softmax could mitigate the bad effects, which we suspect is due to the log function rescaling the range of rewards from $[0, 1]$ to $\mathbb{R}$ again, like without using softmax.

unnormalized explicit scalar reward. Then, $\pi_\theta$ just applies a softmax to all logits in the vocabulary to form a probability distribution. If we omit the pairwise characteristic of the objective of DPO, then the essential difference of the optimization goals between explicit reward models and DPO lies in the following points. DPO first takes a softmax on the "logits rewards", then passes it to a log function, and finally conducts a reduction along the sequence on all tokens by taking a summation. We then analyze which transformation leads to the performance drop in the reward shaping of DPO in the succeeding section.

### 3.3. Exploration of the Harmful Effects of Log Softmax

Here we show the results of training different models with or without log softmax. We based the experiments on the vanilla explicit reward models, which add an extra projection layer $f : \mathbb{R}^d \to \mathbb{R}$ following the final hidden embedding layer to the backbone model. First, starting from the vanilla explicit reward models, we change the input to the projection layer from using only the last hidden embedding to averaging each of the hidden embeddings corresponding to each token position in the sequence. Next, we increase the number of projection heads, with each corresponding to a subset of the total vocabulary. Then, we could add the (log) softmax function among the outputs of different projection heads.

Table 3 demonstrates that simply adding more projection heads and taking the summation along the sequence dimension will not degrade the performance, and only after integrating the (log) softmax functions, a considerable gap has appeared.

### 3.4. Refinement of PRIME

PRIME (Cui et al., 2025) proposes to plug in the implicit rewards defined similarly to the DPO formulation, with further updating through a Proximal Policy Optimization (PPO) (Schulman et al., 2017) way. Based on our observations from the preliminary experiments conducted on the SFT stage, we try to explore whether removing the log softmax function and reference model could improve the performance, as this will mimic the formulation of explicit reward models, which usually have a better generalization ability than implicit reward models.

| Method | MATH500 | Minerva Math | Olympiad Bench | Average |
|---|---|---|---|---|
| Base | 0.3948 | 0.1153 | 0.1941 | 0.2347 |
| PRIME | 0.4450 | 0.1544 | 0.2067 | 0.2687 |
| Ours | **0.4482** | **0.1673** | **0.2122** | **0.2759** |

*Table 4.* The performance evaluated on three math benchmarks: MATH500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), and Olympiad Bench (He et al., 2024), with metric average @ 8, which means take the average accuracy of eight samples per prompt with temperature 1.0.

Table 4 displays the accuracies on several math benchmarks before and after removing the log softmax function during the training of PRIME. Though the log softmax function is directly adopted for training the critic model, while the accuracies are evaluated on the actor model, it is intuitive that the performances of the two are highly correlated. The

performance gain after removing the log softmax indicates the effectiveness of our quick fix.

## 4. Conclusion

From the results of our experiments, we could conclude that DPO has worse performance on preference learning than explicit reward models, especially when generalizing to OOD domains. We analyzed the differences between explicit reward models and DPO, and located the log softmax function as the culprit. Tentative experiments show that after removing the log softmax function, DPO could achieve a performance gain, even matching explicit reward models.

## Limitations

Though we have verified our hypothesis on several tasks, from SFT to actor-critic RL, the experiments could be conducted in more settings. The performance drop caused by DPO via log softmax is potentially related to the fact that DPO entangles all the model parameters as the optimization objective through the softmax function, however, there might exist other underlying possible explanations for the worse performance of implicit reward models compared to explicit ones. Other hypotheses may be proposed and verified beyond ours. In addition, the quick fix mentioned before could serve as a way to match the performance of implicit reward models with explicit ones, but it does not provide a means to surpass existing explicit reward models. Further efforts need to be devoted to this direction to discover a new way that may outperform the commonly adopted explicit reward models.

## References


Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Cui, G., Yuan, L., Wang, Z., Wang, H., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., Liu, J., Qi, L., Liu, Z., and Sun, M. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Huggingface. lighteval/math. URL https://huggingface.co/datasets/lighteval/MATH.

Ivison, H., Wang, Y., Liu, J., Wu, Z., Pyatkin, V., Lambert, N., Smith, N. A., Choi, Y., and Hajishirzi, H. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Advances in neural information processing systems*, 37:36602–36633, 2024.

Jin, B., Zeng, H., Yue, Z., Yoon, J., Arik, S., Wang, D., Zamani, H., and Han, J. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Li, J., Sun, S., Yuan, W., Fan, R.-Z., Zhao, H., and Liu, P. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*, 2023.

Lin, Y., Seto, S., Ter Hoeve, M., Metcalf, K., Theobald, B.-J., Wang, X., Zhang, Y., Huang, C., and Zhang, T. On the limited generalization capability of the implicit reward model induced by direct preference optimization. *arXiv preprint arXiv:2409.03650*, 2024.

Luo, F., Yang, R., Sun, H., Deng, C., Yao, J., Shen, J., Zhang, H., and Chen, H. Rethinking diverse human preference learning through principal component analysis. *arXiv preprint arXiv:2502.13131*, 2025.

Meng, Y., Xia, M., and Chen, D. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741, 2023.

Rafailov, R., Hejna, J., Park, R., and Finn, C. From $r$ to $q^*$: Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Wang, H., Xiong, W., Xie, T., Zhao, H., and Zhang, T. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024.

Xiong, W., Yao, J., Xu, Y., Pang, B., Wang, L., Sahoo, D., Li, J., Jiang, N., Zhang, T., Xiong, C., et al. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

Xu, S., Fu, W., Gao, J., Ye, W., Liu, W., Mei, Z., Wang, G., Yu, C., and Wu, Y. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024.

Yao, J., Hao, Y., Zhang, H., Dong, H., Xiong, W., Jiang, N., and Zhang, T. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *arXiv preprint arXiv:2505.02391*, 2025.

Ye, Z., Li, X., Li, Q., Ai, Q., Zhou, Y., Shen, W., Yan, D., and Liu, Y. Beyond scalar reward model: Learning generative judge from preference data. *arXiv preprint arXiv:2410.03742*, 2024.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623, 2023.

Zhong, J., Shen, W., Li, Y., Gao, S., Lu, H., Chen, Y., Zhang, Y., Zhou, W., Gu, J., and Zou, L. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*, 2025.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.