MPPO: Multi Pair-wise Preference Optimization for LLMs with Arbitrary Negative Samples

Shuo Xie, Fangzhi Zhu*, Jiahui Wang, Lulu Wen, Wei Dai,
Xiaowei Chen, Junxiong Zhu, Kai Zhou, Bo Zheng
Taobao & Tmall Group of Alibaba
shuoxie090@gmail.com;
{fangzhi.zfz,wjh439451,wenlulu.wll,yiwei.dw}@taobao.com
{qisheng.cxw,xike.zjx,kevin.zk}@taobao.com; bozheng@alibaba-inc.com

Abstract

Aligning Large Language Models (LLMs) with human feedback is crucial for their development. Existing preference optimization methods such as DPO and KTO, while improved based on Reinforcement Learning from Human Feedback (RLHF), are inherently derived from PPO, requiring a reference model that adds GPU memory resources and relies heavily on abundant preference data. Meanwhile, current preference optimization research mainly targets single-question scenarios with two replies, neglecting optimization with multiple replies, which leads to a waste of data in the application. This study introduces the MPPO algorithm, which leverages the average likelihood of model responses to fit the reward function and maximizes the utilization of preference data. Through a comparison of Point-wise, Pair-wise, and List-wise implementations, we found that the Pair-wise approach achieves the best performance, significantly enhancing the quality of model responses. Experimental results demonstrate MPPO's outstanding performance across various benchmarks. On MT-Bench, MPPO outperforms DPO, ORPO, and SimPO. Notably, on Arena-Hard, MPPO surpasses DPO and ORPO by substantial margins. These achievements underscore the remarkable advantages of MPPO in preference optimization tasks.

1 Introduction

As large language models (LLMs) advance at an impressive pace, their performance on various tasks approaches and even exceeds that of humans. Generally, the complete training process for a LLM entails three main stages: pre-training (Brown et al., 2020), task-specific fine-tuning (Supervised Fine-Tuning, SFT) (Wei et al., 2021; Wang et al., 2022) and preference optimization.

The pre-training phase involves unsupervised learning on large text datasets, which provides LLMs with a broad foundation of language knowledge (Hu et al., 2022; Almazrouei et al., 2023). However, since pre-trained models typically learn general language patterns, their performance on specific tasks may be insufficient. Therefore, pre-trained LLMs often require further SFT to excel in practical applications.

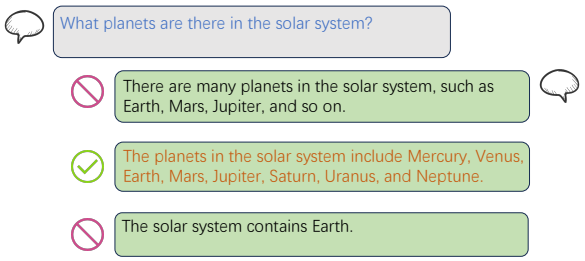


Figure 1: A simple example: When answering questions, the LLMs may generate multiple responses, but the quality of different responses varies.

The SFT process typically involves supervised learning, where the model is trained on a labeled dataset tailored to the target task (Gudibande et al., 2024). SFT enhances LLMs' performance on task-specific metrics, such as accuracy and relevance. However, as depicted in Figure 1, SFT models may produce responses that diverge from human preferences when responding to queries (Carlini et al., 2020; Pryzant et al., 2023). Thus, an efficient preference optimization strategy is crucial for aligning their responses with human values and preferences.

Recent studies on preference optimization, including reinforcement learning with human feedback and direct preference optimization (DPO) (Rafailov et al., 2023), have established effective methods for aligning language models. These approaches have proven successful, as exemplified by models like GPT-4 (Josh et al., 2023) (Ouyang et al., 2022) and Llama-3 (Abhimanyu et al., 2024).

Preference alignment methods have proven successful not only in aligning with human values but

*Corresponding author.

Proceedings of the 31st International Conference on Computational Linguistics, pages 1545–1554
January 19–24, 2025.

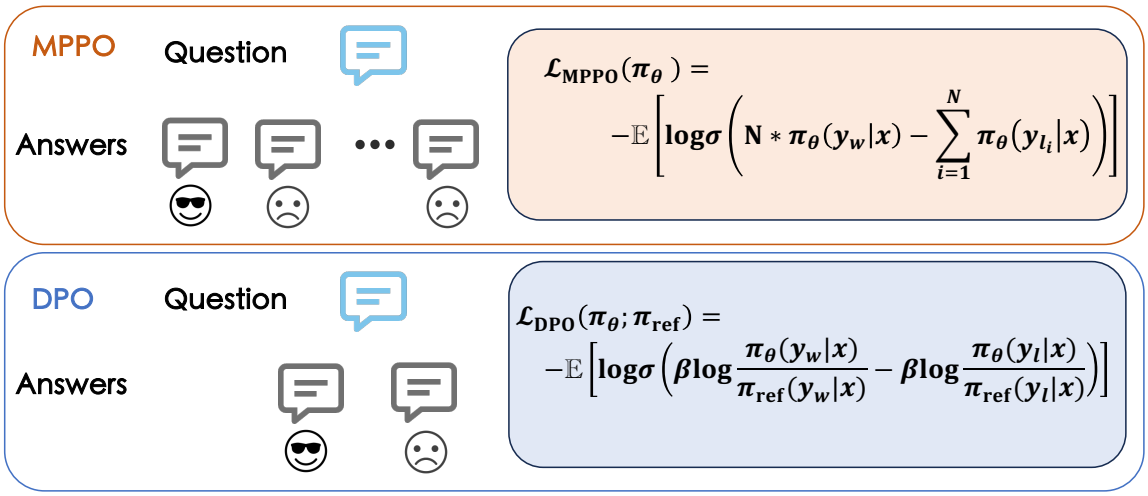


Figure 2: The primary differences between MPPO and DPO are three-fold: (1) MPPO directly models the reward function using the average likelihood of the responses. (2) MPPO can utilize any number of negative samples for training in sparse scenarios. (3) The optimization objective of MPPO does not require a reference model or any hyperparameters, making the model more stable.

also across various downstream tasks such as enhancing factual accuracy (Tian et al., 2023; Cheng et al., 2024) and code-based question answering (Gorbatovski and Kovalchuk, 2024). However, as shown in Figure 2, existing preference alignment methods often require the use of reference models, and the optimization goals are not directly related to the generation of responses.

In this work, we innovatively introduce the use of the average likelihood value of model responses as an approximation for the reward function, named Multi Pair-wise Preference Optimization (MPPO). In the real world, we can generate multiple responses for the same query using different models, although these responses may not be dense. We refer to this as a sparse data scenario, which is more reflective of practical situations. We further explore how to better leverage the preference information from multiple preference samples. In summary, our work makes the following contributions:

- We propose a new algorithm that directly optimizes the policy model without training a reward model or relying on a reference model.
- Our discussion on preference optimization for multiple responses to a single query in sparse data scenarios has led to conclusions that are more suitable for practical applications.
- We thoroughly analyze three primary implementations of MPPO: Point-wise, Pair-wise, and List-wise. Among them, the Pair-wise approach achieves optimal performance, offering insights that can inspire the development of other preference optimization algorithms.
- Our experiments on the UltraFeedback (Cui et al., 2023) dataset demonstrate that the proposed algorithm outperforms previous methods on the MT-Bench and also surpasses algorithms like DPO and ORPO on Arena-Hard.

2 Related Works

In this section, we review the existing research on preference optimization for LLMs. Reinforcement learning from human feedback (RLHF) has been widely applied in the methods of preference optimization. In these methods, people first construct a reward model on preference dataset (Casper et al., 2023), and then fine-tune the LLMs with reinforcement learning algorithms such as proximal policy optimization (PPO) (Schulman et al., 2017) or its variants to maximize the estimated rewards. To enhance the stability of RLHF, Christiano et al. (Christiano et al., 2017) and Ouyang et al. (Ouyang et al., 2022) proposed incorporating KL regularization based on the SFT model into preference optimization.

While PPO has achieved significant success in training high-performance prognostic models, this method requires prior training of a reward model (Gao et al., 2022; Wang et al., 2024). The reward model demands a large amount of dense data to ensure modeling accuracy, and the instability in

training the reward model causes high sensitivity to hyperparameters (Wang et al., 2024). Completing the full training also necessitates loading multiple models, which consumes extensive GPU memory resources, thereby presenting a series of practical challenges.

To address the stringent need for a reward model, numerous scholars have pursued innovation (Song et al., 2023). Recent studies, such as DPO (Rafailov et al., 2023), have attempted to reparameterize the reward function by integrating reward modeling with the preference learning stage, thereby reducing the training costs of the preference optimization phase. Further research building on DPO, such as KTO (Ethayarajh et al., 2024), circumvents the need for paired preference samples and allows for effective preference optimization even in the presence of imbalanced positive and negative samples. DPOP (Pal et al., 2024) specifically optimizes against a potential issue encountered by DPO, that is, the diminishing likelihood of the model's assessment of preference examples during training, to prevent this occurrence. However, methods such as DPO, KTO (Ethayarajh et al., 2024), and DPOP (Pal et al., 2024) still rely on a KL regularizer centered around SFT, and complete training still demands the loading of both the policy model and reference model.

In response to this challenge, odds ratio preference optimization (ORPO) (Hong et al., 2024) investigates the role and impact of SFT in model optimization with pairwise preference datasets, seamlessly integrating the SFT phase with the preference optimization stage and incorporating probability ratios into the optimization objective, thus eliminating the mandatory use of the reference model during the preference optimization phase. Simple preference optimization (SimPO) (Meng et al., 2024) introduces length-normalized rewards and marginal target rewards, which similarly omit the reference model, rendering the preference optimization process both efficient and concise.

Our subsequent research will focus on how to effectively and concisely achieve preference optimization, as well as how to better model rewards in real-world scenarios with preference samples that include multiple responses.

3 Methodology

In this section, we first introduce the background of DPO. Then, we present the fundamental principle of our approach: fitting the reward function with the responses' average likelihood of the model. Extending from this principle, we have outlined three primary methods of implementation.

3.1 Direct Preference Optimization

DPO is a highly successful method for offline preference optimization. Compared to the RLHF training approach, DPO reparameterizes the reward function $r(x, y)$ using a closed-form expression with the optimal policy:

$$r(x, y) = \beta \log \left(\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \right) + \beta \log Z(x), \quad (1)$$

where π_θ is the policy model, π_{ref} is the reference policy, typically the supervised SFT model, and $Z(x)$ is the partition function. Integrating the reward model into the Bradley-Terry objective:

$$p(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l)). \quad (2)$$

DPO expresses the probability of preference data through a policy model rather than a reward model, thus generating the following objective:

$$\mathcal{L}_{\text{DPO}(\pi_\theta;\pi_{\text{ref}})} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log\sigma \left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right], \quad (3)$$

where (x, y_w, y_l) are preference pairs consisting of the query, the winning response, and the losing response from the preference dataset $\mathcal{D}$. DPO integrates reward modeling with the preference learning stage, thus reducing the training costs of the preference optimization phase.

3.2 MPPO: Fitting the Reward Function to the Average Likelihood of the Model Responses

Using Eq. (3) as the optimization objective in DPO has three drawbacks: (1). It necessitates loading a reference model during training, which results in additional GPU memory requirements and computational costs. (2). Training data requires strictly positive and negative preference pairs. When dealing with multiple responses to a single query, this requirement leads to significant data redundancy and inefficiency. (3). DPO uses the policy model to represent the probability of preference data, but this does not fully correspond to the actual probability of the preference data.

To address the aforementioned issue, we proposed MPPO, which uses the average likelihood fitting of the model's responses as the reward function:

$$r_{\text{MPPO}}(x, y) = \prod_{i=1}^{|y|} \pi_\theta(y_i|x, y_{<i})^{\frac{1}{|y|}}, \quad (4)$$

where $\pi_\theta(y_i|x, y_{<i})$ represents the probability of generating the i-th token y_i given the input x and the preceding tokens $y_{<i}$. Here, $|y|$ denotes the length of the response y, which is the total number of tokens in the response. This function calculates the geometric mean of the likelihoods for all tokens in the response. When the model is well-constructed, this approach increases the likelihood of generating better responses, guiding the model to favor producing superior responses.

3.3 Three Primary Methods of Implementation

The MPPO algorithm can be implemented in various ways, making it crucial to evaluate these methods to determine the most effective approach for aligning LLMs.

To more comprehensively study which aspects of preference data and implementation methods are most critical, we assume access to complete preference information and a diverse dataset that includes annotations for both high-quality and low-quality responses, as well as specific scores for each response (i.e., one query with n responses and corresponding scores for those responses), we examine the effectiveness of different implementation strategies based on this premise.

There are three primary implementation approaches: Point-wise, Pair-wise, and List-wise. In the following discussion, we denote p as:

$$p = \prod_{i=1}^{|y|} \pi_\theta(y_i|x, y_{<i})^{\frac{1}{|y|}}. \quad (5)$$

3.3.1 Implementation Approach Based on Point-Wise

The idea behind Point-wise approaches is that each query, response, and score is considered individually during training, rather than as pairs. This allows the score to be aligned with Point-wise predictions. Since the score takes on discrete values, this process can be treated as a multi-class classification problem. Optimization objectives can include cross-entropy loss and mean squared error, as described in Eq. (6) and Eq. (7), respectively.

$$\mathcal{L}_{\text{Point-CE}}(\pi_\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\text{score}\cdot\log(p) + (1-\text{score})\cdot\log(1-p)\Big], \quad (6)$$

$$\mathcal{L}_{\text{Point-MSE}}(\pi_\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[(\text{score}-p)^2\right], \quad (7)$$

where score represents the reward value assigned to each response, and y can be both y_w and y_l.

3.3.2 Implementation Approach Based on Pair-Wise

Pair-wise approaches focus on handling pairs of data or binary relations within datasets. At each instance, two response samples are selected—one designated as the positive sample and the other as the negative sample, based on their respective scores. The objective is to increase the average likelihood of selecting the positive sample while decreasing the likelihood of selecting the negative sample.

In the special case where each piece of data only contains one positive sample and one negative sample, the reward formula $r(x, y) = p$ can be substituted into the Bradley-Terry ranking objective $p(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l))$, resulting in Pair-Single optimization function (8):

$$\mathcal{L}_{\text{Pair-Single}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\sigma(p_w - p_l)\right]. \quad (8)$$

However, in practical settings, obtaining large volumes of human preference data is comparatively easy and efficient. For example, a series of responses can be quickly generated from an identical query. Consequently, we consider the implementation of a Pair-wise approach when there are $N+1$ responses to the same prompt. A straightforward approach is to mark the response with the highest score value among the $N+1$ answers as the positive sample, and all the others as negative samples. This extends the Pair-Single method to two new variants: Pair-Multi-N-Separate (Pair-MNS) and Pair-Multi-N-Merge (Pair-MNM), as shown in Eq. (9) and Eq. (10).

$$\mathcal{L}_{\text{Pair-MNS}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_{l_i})\sim\mathcal{D}}\left[\sum_{i=1}^{N}\underbrace{\log\sigma(p_w - p_{l_i})}_{\text{total of } N \text{ items}}\right], \quad (9)$$

$$\mathcal{L}_{\text{Pair-MNM}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_{l_i})\sim\mathcal{D}} \left[\log\sigma\left(N\cdot p_w - \sum_{i=1}^{N} p_{l_i}\right)\right]. \quad (10)$$

Subsequently, we adopted OpenAI's strategy for selecting data when training reward models, which involves randomly choosing a pair from K data points for comparative training (Ouyang et al., 2022). In our method, we randomly select two data points from the $N+1$ responses and compare them based on their scores. The sample with the higher score is treated as the positive instance, while the one with the lower score is considered the negative instance. Based on this, we extend Equations (9) and (10) to incorporate this training framework as Pair-Multi-Combination-Separate (Pair-MCS) and Pair-Multi-Combination-Merge (Pair-MCM) in Eq. (11) and Eq (12):

$$\mathcal{L}_{\text{Pair-MCS}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_{l_i})\sim\mathcal{D}} \left[\sum_{i=1}^{N} \underbrace{\log\sigma(p_w - p_{l_i})}_{\text{total of } C_{N+1}^2 \text{ items}}\right], \quad (11)$$

$$\mathcal{L}_{\text{Pair-MCM}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_{l_i})\sim\mathcal{D}} \left[\log\sigma\left(\sum_{i=1}^{N} \underbrace{(p_w - p_{l_i})}_{\text{total of } C_{N+1}^2 \text{ items}}\right)\right]. \quad (12)$$

3.3.3 Implementation Approach Based on List-Wise

Unlike Pair-wise methods, which focus on pairwise preferences between items, the List-wise approach considers the ordering of items across the entire list. This method aims to directly optimize ranking models based on a list's overall ranking quality. In this context, the list of responses contains $N+1$ items. Notably, when $N=1$, the List-wise and Pair-wise approaches are equivalent. A specific implementation of the List-wise method is List-MLE (Lan et al., 2014). List-MLE applies Maximum Likelihood Estimation (MLE) directly to ranking list data, integrating Eq. (4) into the List-wise ranking loss to derive the List-MLE optimization function:

$$\mathcal{L}_{\text{List-MLE}}(\pi_\theta) = -\mathbb{E}_{(x,y_1,\ldots,y_{N+1})\sim\mathcal{D}} \left[\log\prod_{i=1}^{N+1}\frac{\exp(p_i)}{\sum_{j=i}^{N+1}\exp(p_j)}\right], \quad (13)$$

where p_i represents the reward value for the i-th item, and the items are arranged in descending order of their reward values.

4 Experimental Settings

4.1 Training Configurations

4.1.1 Model

For training, we utilized the Llama3-8B model (Abhimanyu et al., 2024), following the setups of Zephyr (Tunstall et al., 2023) and SimPO. The training process began by fine-tuning a foundational model on the UltraChat-200k dataset (Ding et al., 2023) to obtain a supervised fine-tuned (SFT) model. This SFT model then served as the initial model for preference optimization on the UltraFeedback dataset (Cui et al., 2023).

To ensure transparency, the SFT model was trained on open-source data, and we used the publicly available Llama3-8B-SFT weights from SimPO as our baseline. Preference optimization was conducted using three different implementations of MPPO: Point-wise, Pair-wise, and List-wise. Within each implementation, we experimented with different variations to determine the most effective alignment method for LLMs.

Unlike SimPO, which required extensive hyperparameter tuning, our approach only involved adjusting the learning rate, significantly reducing training costs while maintaining consistency and reliability.

4.1.2 Datasets

We conducted preference optimization training on the UltraFeedback dataset, which includes 64k instructions. For each instruction, four models generated responses, and GPT-4 rated each response from 1 to 10 based on instruction adherence, authenticity, honesty, and helpfulness, with higher scores indicating better responses.

In the Point-wise implementation, each instruction with its four responses and reward values was split into four separate samples, totaling 256k data points (64k*4). Reward values were normalized to a range of 0.1 to 1 by dividing by 10.

In the Pair-Single implementation, the highest-scoring response was labeled positive, and one of the remaining three responses was randomly chosen as negative.

In the List-wise implementation, all four responses were sorted by their scores and trained together as a list.

4.2 Leaderboard Evaluation

To evaluate our model, we use two of the most popular open-ended instruction-following benchmarks: MT-Bench and Arena-Hard (details in Table 1). MT-Bench encompasses 80 tasks across 8 categories, with each task consisting of two rounds of question-and-answer phases. The newly released Arena-Hard is an enhanced version of MT-Bench that includes 500 clearly defined technical problem-solving queries. We report scores in accordance with the evaluation protocols of each benchmark. For MT-Bench, we provide the average score using GPT-4-Turbo-0409 as the evaluating model, which implements stricter grading criteria. For Arena-Hard, we report the win rate in comparison to the baseline model (GPT-4-0314).

5 Results and Analysis

In this section, we present the results of our experiments in Section 5.1, which include the outcomes for various implementations of MPPO and comparisons with SOTA preference optimization algorithms such as DPO, KTO, ORPO, and SimPO. In the subsequent subsections, 5.2 and 5.3, we draw several conclusions regarding preference optimization by analyzing and comparing these experimental results.

We have primarily explored four Research Questions (RQs) regarding the MPPO method:

- **RQ1:** Are all implementations of MPPO: Point-wise, Pair-wise, and List-wise effective? Which method is the most effective?
- **RQ2:** Is the use of multiple samples ($N + 1$) preferable to optimizing with just a single positive and a single negative sample?
- **RQ3:** In sparse data scenarios, is the optimization goal of collaboratively leveraging multiple samples for preference optimization effective?
- **RQ4:** In sparse data scenarios with multiple responses, is it necessary to consider multiple samples for reward fitting (MCM), or is it sufficient to focus on just one optimal response (MNM)?

5.1 Main Results

In Table 2, we present the results of various implementations of MPPO and several preference optimization algorithms on MT-bench and Arena-hard benchmarks. It is observable that while all algorithms yield certain performance gains in the preference optimization for the SFT model, the magnitude of these improvements varies. The Pair-MNM implementation of MPPO achieved the highest scores on the MT-Bench leaderboard, surpassing the SFT model and the DPO, SimPO by 1.54 points, 0.23 points, and 0.19 points, respectively, demonstrating a significant enhancement and establishing itself as the latest SOTA algorithm.

In the Arena-Hard evaluation, the Pair-MNM implementation of MPPO placed second with a win rate of 21.6, only trailing behind SimPO, which had a win rate of 23.4. Pair-MNM outperformed DPO (win rate of 15.9), KTO (12.8), and ORPO (10.7).

5.2 Comparison of Three Implementation Approaches: Point-wise, Pair-wise, and List-wise (RQ1)

Based on the analysis results in Table 2, while MPPO algorithm has various implementation strategies, not all strategies can achieve effective goals. Firstly, among the three MPPO implementation strategies, the Pair-wise method stands out, outperforming the List-wise and Point-wise methods in both MT-Bench and Arena-hard benchmarks.

The Point-wise method, despite high expectations, did not perform as well as anticipated. In fact, it underperformed compared to the original SFT model on the MT-Bench evaluation set. This suggests that relying solely on the magnitude of label information is inadequate and that incorporating preference information is crucial. Figure 3 illustrates the issue with the Point-wise method's Point-CE training: the scores for all answers remain relatively high (between 0.1 and 1) throughout the training period. Consequently, the average likelihood of each answer increases uniformly. However, the score difference between the best response and several poor responses decreases, making it harder to distinguish high-quality response from lower-quality ones. This explains why the Point-CE model performs worse than the SFT model.

Additionally, it is important to note that there is only a positive correlation, not an exact correspondence, between the responses' average likelihood and reward values. Therefore, directly approximating these specific values may not be effective. The inherent randomness of scores generated by GPT-4 also adds complexity to the modeling process.

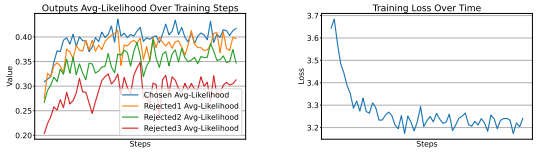
In conclusion, the List-wise approach exhibits

	Exs.	Baseline Model	Judge Model	Scoring Type	Metric
MT-Bench	80	-	GPT-4-Turbo-0409	Single-answer grading	Rating of 1-10
Arena-Hard	500	GPT-4-0314	GPT-4-Turbo-0409	Pairwise comparison	Win rate

Table 1: Comparison of baseline and judge models on MT-Bench and Arena-Hard datasets.

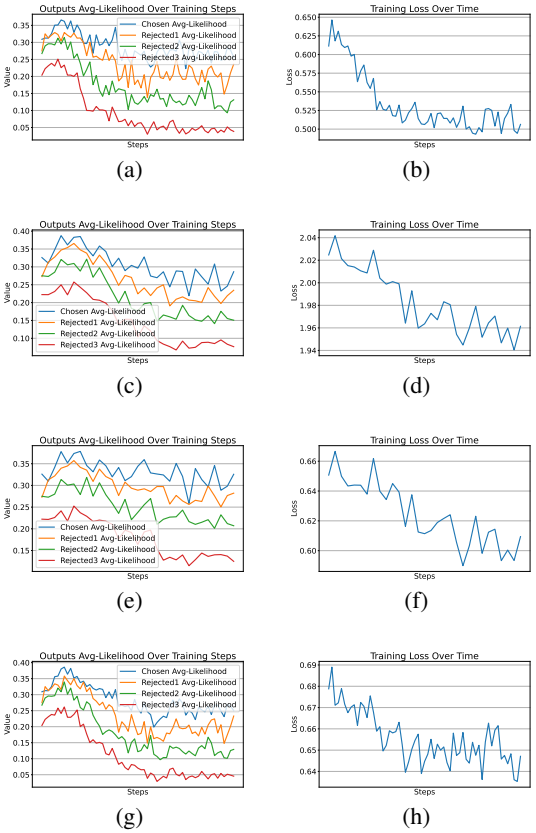
Method	Mt-Bench	Arena-Hard
SFT	4.62	3.3
DPO	5.93	15.9
KTO	5.87	12.8
ORPO	5.49	10.7
SimPO	5.97	**23.4**
Point-CE	4.38	12.8
Point-MSE	4.43	13.1
Pair-Single	5.96	19.1
Pair-MNS	5.84	5.1
Pair-MNM	**6.16**	21.6
Pair-MCS	5.72	14.6
Pair-MCM	5.77	7.8
List-MLE	5.87	5.4

Table 2: The results of MT-Bench and Arena-Hard. The SFT models are trained on the UltraChat dataset, and then preference optimization models are trained from the SFT models using algorithms such as DPO, MPPO, etc.



(a) (b)

Figure 3: Training Logs for Point-CE (N=3). (a) shows the variation in average likelihood for the chosen and three rejected samples; (b) illustrates the overall loss variation during the training period.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4: Based on the Pair-wise implementation approach, the training records (N=3) are organized and listed from top to bottom as follows: (a) and (b). Pair-Single implementation; (c) and (d). Pair-MNS implementation; (e) and (f). Pair-MNM implementation; (g) and (h). Pair-MCM implementation.

certain disadvantages compared to Pair-wise methods. While it performs well on the MT-Bench benchmark, its performance is poor on Arena-Hard. This indicates the instability of the List-wise method, as well as the higher level of challenge presented by the Arena-Hard evaluation set compared to the MT-Bench. Although List-wise approach take preference information into account, it focus on preference ranking and do not strongly reflect preference information. Therefore, the list ranking information can be considered as weak preference information.

5.3 Comparison of Implementation Methodology Based on Pair-wise

In various implementations based on the Pair-wise strategy, the Pair-MNM approach achieved the best evaluation results. Subsequently, we will compare other Pair-Wise-based implementations to analyze the reasons behind the superior performance of Pair-MNM.

Considering all responses will enhance the preference model. (RQ2) Compared to Single-pair optimization, the Pair-MNM approach evaluates all samples collectively. It designates the highest-quality answer as the positive sample and classifies the remaining responses as negative. This

method enhances the incorporation of preference information by focusing on increasing the average likelihood of the chosen answers and decreasing that of the rejected ones. As shown in Figures 4a and 4c, Pair-MNM effectively amplifies the distinction between the chosen answer and each rejected one throughout the training process. This results in improved model performance, as it better learns to differentiate between varying responses.

Incorporating Multiple Rejections Collaboratively for Optimal Objective Achievement in Preference Optimization. (RQ3) Both Pair-MNM and Pair-MNS approaches, while based on the Single-pair method and considering multiple rejections, differ in how they handle these rejections. Pair-MNS calculates the average likelihood difference between the chosen instance and each rejected instance separately, summing the log-sigmoid values of these differences. This means each rejected instance independently influences the optimization objective. In contrast, Pair-MNM considers all rejected instances collectively, aiming to synergistically account for their combined impact to optimize the overall objective.

The subtle differences in the optimization objectives lead to different outcomes during training, as depicted in Figures 4 c-f. While the loss curves and average likelihood trends are fundamentally similar, Pair-MNM consistently achieves an average likelihood of approximately 0.05 higher than Pair-MNS. This suggests that Pair-MNM, by integrating the discrepancies between multiple rejections relative to the chosen instance, results in a more effective decision boundary.

Just need one Optimal Response. (RQ4) The main difference between Pair-MNM and Pair-MCM lies in how positive and negative samples are selected in sparse data scenarios. Pair-MNM chooses the highest-scoring sample as positive and labels all others as negative. In contrast, Pair-MCM uses a method similar to OpenAI's approach: it draws two samples from N+1 data points, compares them based on their scores, and labels the higher-scoring sample as positive.

Unlike Pair-MNM, which treats all rejected samples equally, Pair-MCM adjusts suppression intensity based on each sample's score, applying greater suppression to lower-scoring samples and less to higher ones. As illustrated in Figure 4g, this approach sometimes results in incorrect promotion of samples relative to the positive sample, as seen in the mean likelihood trends.

Our experiments show that Pair-MCM does not perform better in sparse data scenarios. Thus, for multiple-answer preference optimization in such contexts, focusing on a single optimal response (Pair-MNM) and suppressing other samples is more effective for achieving better model performance and preference optimization.

6 Conclusion

In this article, we present MPPO (Multi Pair-wise Preference Optimization), a preference optimization algorithm designed for directly modeling reward models in sparse data scenarios. Compared to existing methods, MPPO demonstrates superior performance on the Llama3-8B model. Our analysis reveals that Pair-wise implementations outperform Point-wise and List-wise approaches. Additionally, considering all responses enhances the preference model, and collaboratively addressing multiple rejections yields optimal results. Notably, only one optimal response is needed, eliminating the need for multiple sampling of preference pairs. MPPO effectively illustrates how to leverage preference data from multiple responses to a single query and addresses common challenges in real-world sparse data applications.

Limitations and Ethics

Our work primarily proposes a preference optimization algorithm for directly modeling reward models in sparse data scenarios, without the need for a reference model. We acknowledge that the main limitations of this study are as follows:

1. We covered various implementation methods, including Point-wise, Pair-wise, and List-wise, as well as others like logistic ranking loss and ListNet. Future work will explore a broader range of methods and analyze their strengths and weaknesses in more detail.

2. We did not clearly define the boundary between data-rich and data-scarce scenarios in preference optimization. This will be addressed in future work through further discussion and experimental analysis.

All experiments are conducted on publicly available datasets; no scientific ethical violations or privacy infringements occurred.

Acknowledgments

The research work is supported by Alibaba Group through Alibaba Research Intern Program.

References

Abhimanyu, Dubey, and et al. 2024. The llama 3 herd of models.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra-Aimée Cojocaru, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *ArXiv*, abs/2311.16867.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. In *USENIX Security Symposium*.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, J'er'emy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Ségerie, Micah Carroll, Andi Peng, Phillip J. K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco di Langosco, Peter Hase, Erdem Biyik, Anca D. Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *ArXiv*, abs/2307.15217.

Qinyuan Cheng, Tianxiang Sun, Xiangyang Liu, Wenwei Zhang, Zhangyue Yin, Shimin Li, Linyang Li, Kai Chen, and Xipeng Qiu. 2024. Can ai assistants know what they don't know? *ArXiv*, abs/2401.13275.

Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *ArXiv*, abs/1706.03741.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with scaled ai feedback.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Conference on Empirical Methods in Natural Language Processing*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *ArXiv*, abs/2402.01306.

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*.

Alexey Gorbatovski and Sergey Kovalchuk. 2024. Reinforcement learning for question answering in programming domain using public community scoring as a human feedback. In *Adaptive Agents and Multi-Agent Systems*.

Arnav Gudibande, Eric Wallace, Charlie Victor Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2024. The false promise of imitating proprietary language models. In *The Twelfth International Conference on Learning Representations*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *ArXiv*, abs/2403.07691.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2022. A survey of knowledge enhanced pre-trained language models. *IEEE Transactions on Knowledge and Data Engineering*, 36:1413–1430.

OpenAI Josh, Achiam, and et al. 2023. Gpt-4 technical report.

Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. 2014. Position-aware listmle: a sequential learning process for ranking. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 449–458, Arlington, Virginia, USA. AUAI Press.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *ArXiv*, abs/2405.14734.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *ArXiv*, abs/2402.13228.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *ArXiv*, abs/2305.18290.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Feifan Song, Yu Bowen, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2023. Preference ranking optimization for human alignment. *ArXiv*, abs/2306.17492.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D. Manning, and Chelsea Finn. 2023. Fine-tuning language models for factuality. *ArXiv*, abs/2311.08401.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *ArXiv*, abs/2310.16944.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652.