Permutative Preference Alignment from Listwise Ranking of Human Judgments

Yang Zhao[1†] **Yixin Wang**[2] **Mingzhang Yin**[3†]
[1]University of Texas at Austin [2]University of Michigan, Ann Arbor [3]University of Florida
yangzhao25@utexas.edu yixinw@umich.edu
mingzhang.yin@warrington.ufl.edu

Abstract

Aligning Large Language Models (LLMs) with human preferences is crucial in ensuring desirable and controllable model behaviors. Current methods, such as Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO), rely on the Bradley-Terry (B-T) model to maximize the likelihood of pairwise choices. However, when multiple responses are available, the B-T model fails to guarantee an accurate list ranking of the responses. To address this issue, we propose Permutative Preference Alignment (PPA), a novel offline listwise approach that incorporates the Normalized Discounted Cumulative Gain (NDCG)—a widely-used ranking metric—as an alternative training objective for LLM alignment. We develop an end-to-end alignment algorithm by approximating NDCG with a differentiable surrogate loss. Experiments demonstrate that PPA outperforms existing pairwise and listwise methods on evaluation sets and general benchmarks such as AlpacaEval. Furthermore, we show that NDCG-based approaches improve ranking accuracy more effectively than B-T-based methods and provide a theoretical explanation for this improvement.

1 Introduction

Large Language Models (LLMs) trained on massive datasets have demonstrated impressive capabilities in natural language processing (Achiam et al., 2023; Dubey et al., 2024). Aligning these models with human preferences is essential for reliable and controllable model behaviors. Pairwise methods, such as Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO) (Rafailov et al., 2023), employ the Bradley-Terry (B-T) model (Bradley and Terry, 1952) to maximize the likelihood of pairwise preferences, demonstrating strong performances (Christiano et al., 2017; Ouyang et al., 2022). Various pairwise-based offline preference optimization methods have been developed, such as RRHF (Yuan et al., 2023), SLiC (Zhao et al., 2023), RPO (Yin et al., 2024), SimPO (Meng et al., 2024), and LiPO-λ (Liu et al., 2024), which depend on the human preferences elicited from pairwise comparisons. These contrastive methods essentially classify preferred and non-preferred responses as positive and negative samples, naturally suited for the binary responses in the data sets like Reddit TL;DR and AnthropicHH (Stiennon et al., 2020; Bai et al., 2022).

However, multi-response data are often available, where a single prompt corresponds to several responses with assigned rewards (Ouyang et al., 2022; Yuan et al., 2023; Dong et al., 2023; Köpf et al., 2024). For such data, DPO cannot ensure the correct ranking of individual pairs, as it infers relative quality rankings of responses by maximizing the pairwise choice probability, potentially leading to an inaccurate overall list ranking. Among LTR metrics, NDCG emerges as the ideal training objective because it handles graded relevance and incorporates position-based discounting. Unlike MAP, MRR, Precision, and Recall which suffer from binary relevance limitations and non-differentiability. NDCG (Vargas and Castells, 2011) can be effectively approximated with differentiable surrogates. This property makes it uniquely suited for gradient-based alignment optimization.

In this work, we propose Permutative Preference Alignment (PPA), a new listwise alignment approach to align human preferences by maximizing NDCG. Models can be viewed as score functions that assign reward scores to responses. The alignment is learning to rank these responses to match the permutation derived from ground truth labels. We employ the smooth surrogate loss NeuralNDCG (Pobrotyn and Białobrzeski, 2021) to approximate NDCG to overcome its non-differentiable nature.

In real-world generation tasks, we find that the

† Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 310–334
November 4-9, 2025

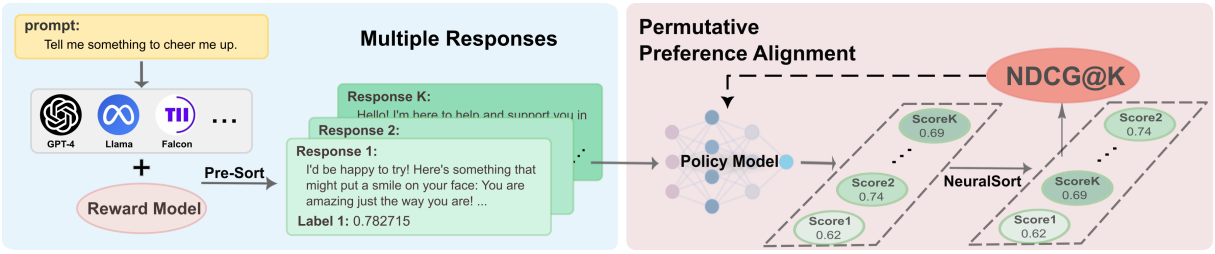


Figure 1: An illustration of Permutative Preference Alignment (PPA) workflow. Each response is assigned a ground truth label by the reward model and pre-sorted in descending order. Reward scores are then derived from the policy and re-sorted to a new permutation. PPA calculates NDCG@K from the difference between two permutations and then optimizes the policy model.

proposed PPA achieves higher ranking accuracy than B-T model-based methods, a crucial metric in evaluating policy performance. Existing literature shows that B-T-based RLHF and DPO struggle to improve ranking accuracy because they maximize the reward margin between preferred and non-preferred responses (Chen et al., 2024). In contrast, maximizing NDCG only requires the preferred responses to have higher reward scores than the non-preferred ones. Based on this distinction, we provide a theoretical explanation for the improvement of PPA in ranking accuracy.

In empirical studies, we comprehensively evaluate model performance with various pairwise and listwise baselines. In addition, we construct a multiple response dataset assigned with rewards based on UltraFeedback (Cui et al., 2023) and SimPO. The proposed PPA consistently achieves the best performance on both evaluation datasets and general benchmarks like AlpacaEval (Li et al., 2023).

Our contributions are summarized as follows:

- We identify potential limitations of DPO in list ranking and introduce NDCG as a training objective to improve ranking performance.
- We propose PPA as a new listwise alignment method that leverages multiple responses, which demonstrates superior performance over existing pairwise and listwise approaches across various model scales.
- We illustrate that NDCG-based method is more effective than Bradley-Terry-based methods in improving ranking accuracy and propose a theoretical explanation.

2 Related Work

Recent approaches for aligning language models with human preferences typically fall into three categories. **Pairwise preference methods** like DPO (Rafailov et al., 2023) use the Bradley-Terry model to optimize binary preferences without explicit reward models. But in multiple-response scenarios, they focus on average contrastive probability rather than ensuring all individual pairs align with ground truth labels. **Multiple response alignment** methods like RRHF, LiPO-λ, DPO-PL, PRO, and LIRE (Yuan et al., 2023; Liu et al., 2024; Rafailov et al., 2023; Song et al., 2024; Zhu et al., 2024) expand candidate responses from various LLMs and optimize the model with Bradley-Terry-based or listwise algorithms. Pairwise methods encounter similar limitations as in binary-response conditions, while listwise methods fail to optimize the established evaluation metrics prevalent in the LTR literature, like NDCG. **Learning to Rank (LTR)** techniques offer promising directions, particularly listwise approaches (Xia et al., 2008a) that consider entire ranking lists as training instances, better capturing response relationships compared to pointwise and pairwise methods. Despite their potential, current listwise techniques have not achieved state-of-the-art performance in LTR, highlighting opportunities for improvement. The further related work details are provided in Appendix A.

3 Preliminaries

Our approach adopts the Learning to Rank (LTR) framework and the list permutations.

3.1 Problem Setting

Following the setup in LiPO (Liu et al., 2024), we assume access to an offline static dataset $\mathcal{D} = \{x^{(i)}, \mathbf{Y}^{(i)}, \mathbf{\Psi}^{(i)}\}_{i=1}^{N}$, where $\mathbf{Y} = (y_1, ..., y_K)$ is a list of responses from various generative models of size K given the prompt x. Each response is associated with a label from $\mathbf{\Psi} = (\psi_1, ..., \psi_K)$, also known as the *ground truth labels* in the LTR

literature. The label ψ measures the quality of responses, which can be generated from human feedback or a pre-trained reward model. We obtain the score $\mathbf{\Psi}$ from a reward model:

$$\psi_k = RM(x, y_k), \tag{1}$$

where $\psi_k \in [0, 1]$. The label is fixed for a response, representing the degree of human preference.

For each prompt-response pair, we also compute a *reward score* representing the likelihood of the generating probability of the response:

$$s_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}. \tag{2}$$

Here, π_{ref} is a reference model which we set as the SFT model. $\pi_\theta(y|x)$ and $\pi_{\text{ref}}(y|x)$ means the probability of the response y given the prompt x under the policy model and the reference model. Similar to DPO (Rafailov et al., 2023), the partition function is omitted due to the symmetry in the choice model of multiple responses. Unlike the fixed labels ψ_k, the reward scores $\mathbf{s} = \{s_\theta(x, y_1), ..., s_\theta(x, y_K)\}$ depend on the model π_θ and are updated during the model training.

3.2 NDCG Metric

NDCG is widely used for evaluating the ranking model performance (Järvelin and Kekäläinen, 2002), which directly assesses the quality of a permutation from the listwise data. Assume the list of responses $\mathbf{Y} = (y_1, ..., y_K)$ have been pre-ranked in the descending order based on labels $\mathbf{\Psi} = (\psi_1, ..., \psi_K)$ from Eq 1, where $\psi_i \geq \psi_j$ if $i \geq j$. The Discounted Cumulative Gain at k-th position ($k \leq K$) is defined as:

$$\text{DCG@}k = \sum_{j=1}^{k} G(\psi_j) D(\tau(j)), \tag{3}$$

where ψ_j denotes the ground truth labels of the response y_j, and $\tau(j)$ is the descending rank position of y_j based on the reward scores $\mathbf{s}$ computed by the current model π_θ. Typically, the discount function and the gain function are set as $D(\tau(j)) = \frac{1}{\log_2(\tau(j)+1)}$ and $G(\psi_j) = 2^{\psi_j} - 1$. An illustration is provided in Appendix D.2.

The NDCG at k is defined as

$$\text{NDCG@}k = \frac{1}{\text{maxDCG@}k} \text{DCG@}k, \tag{4}$$

where maxDCG@k is the maximum value of DCG@k, computed by ordering the responses $\mathbf{Y}$ by their ground truth labels $\mathbf{\Psi}$. The normalization ensures that NDCG is within the range (0, 1).

The value k of NDCG@k ($k \leq K$) indicates that we focus on the ranking of the top k elements while ignoring those beyond k. For example, when $k = 2$, we only need to correctly order the first 2 elements, regardless of the order of the remaining $K - 2$ elements in the list.

4 Permutative Preference Alignment

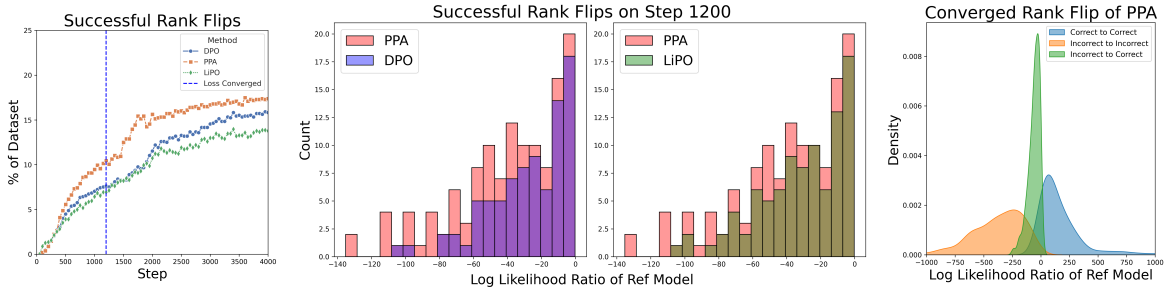
In LLM alignment, the reward scores $\mathbf{s}$ in Equation (2) are key to connecting the loss to the model parameters θ. However, there is a gap between using NDCG as an evaluation metric and as a training objective. Since NDCG is non-differentiable with respect to reward scores $\mathbf{s}$, gradient descent cannot be directly applied for optimization.

To overcome this limitation, surrogate losses (Valizadegan et al., 2009) have been developed. These losses approximate the NDCG value by converting its discrete and non-differentiable characteristics into a continuous and score-differentiable form, suitable for backpropagation. The original NDCG is computed by iterating over each list element's gain value and multiplying it by its corresponding position discount, a process known as the *pairing between gains and discounts*. Thus, surrogate losses can be interpreted in two parts: pairing gains and discounts to approximate the NDCG value, and ensuring these functions are differentiable to enable gradient descent optimization. We will leverage NeuralNDCG (Pobrotyn and Białobrzeski, 2021) as such a surrogate loss.

4.1 PPA Objective

Our PPA incorporates a score-differentiable sorting algorithm-NeuralSort (Grover et al., 2019)-to align gain values $G(\cdot)$ with position discounts $D(\cdot)$. This sorting operation is achieved by left-multiplying a permutation matrix $P_{\text{sort}(\mathbf{s})}$ in Eq 19 with the score vector $\mathbf{s}$ to obtain a list of scores sorted in descending order. The element $P_{\text{sort}(\mathbf{s})}[i, j]$ denotes the probability that response y_j is ranked in the i-th position after re-sorting based on $\mathbf{s}$. Applying this matrix to the gains $G(\cdot)$ results in the sorted gains vector $\widehat{G(\cdot)}$, which is aligned with discounts. Details about NeuralSort are shown in the Appendix D. For simplicity, we denote $\widehat{P}_{\text{sort}(\mathbf{s})}$ as $\widehat{P}$.

Similar to the original NDCG, but with the gain function $G(\cdot)$ replaced by $\widehat{G(\cdot)} = \widehat{P} \cdot G(\cdot)$ to ensure proper alignment between gains and discounts. The



(a) Successful rank flip ratio. (b) Successful rank flip histogram. (c) Rank flip distribution.

Figure 2: Comparisions among PPA, DPO, and LiPO on rank flips. (a) PPA demonstrates a higher efficiency in successful rank flips. The dashed line refers to the steps in which the loss objective is converged for all three methods. (b) PPA demonstrates more successful rank flips in loss-converged steps compared to DPO and LiPO. (c) The successful flip (Incorrect to Correct) distribution is highly constrained to reference ranking accuracy y_w and y_l.

estimated gain at rank j can be interpreted as a weighted sum of all gains, where the weights are given by the entries in the j-th row of $\widehat{P}$. Since $\widehat{P}$ is a row-stochastic matrix, each row sums to one, though the columns may not. This can cause $\widehat{G}$ to disproportionately influence the NDCG value at certain positions. To address this issue, we use the Sinkhorn scaling (Sinkhorn, 1964) on $\widehat{P}$ to ensure each column sums to one. Then we get:

$$\text{NeuralNDCG@}k\ (\tau; \mathbf{s}, \mathbf{\Psi}) = N_k^{-1} \sum_{j=1}^{k} (\text{scale}(\widehat{P}) \cdot G(\mathbf{\Psi}))_j \cdot D(j) \tag{5}$$

where N_k^{-1} represents the maxDCG@k (for $k \leq K$) as defined in Eq 4. The function scale$(\cdot)$ denotes Sinkhorn scaling, and $G(\cdot)$ and $D(\cdot)$ are the gain and discount functions, respectively, as in Eq 3. The proposed PPA is illustrated in Figure 1. Intuitively, the gain function should be proportional to the label, effectively capturing the relative ranking of different responses. The discount function penalizes responses appearing later in the sequence, as in many generation or recommendation tasks, the focus is on the top-ranked elements. Thus, higher-ranked responses have a more significant impact on the overall loss in NeuralNDCG. Further illustrations are provided in Appendix D.2.

Finally, we derive the PPA objective:

$$\mathcal{L}_{\text{PPA}}(\pi_\theta, k; \pi_{\text{ref}}) = -\mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\big[\textstyle\sum_{j=1}^{k} (\text{scale}(\widehat{P}) \cdot G(\mathbf{\Psi}))_j \cdot D(j)/N_k\big]. \tag{6}$$

4.2 Other Approximation of NDCG

In addition to aligning gains and discounts, we can modify the discount function to be differentiable. ApproxNDCG (Qin et al., 2010) is proposed as an approximation to the rank position in the NDCG equation (Eq 3) using the sigmoid function:

$$\begin{aligned} \widehat{\tau(j)} &= 1 + \sum_{i \neq j} \frac{\exp\left(-\alpha(s_j - s_i)\right)}{1 + \exp\left(-\alpha(s_j - s_i)\right)} \\ &= 1 + \sum_{i \neq j} \sigma(\alpha(s_i - s_j)) \end{aligned} \tag{7}$$

As observed, if $s_i \gg s_j$, the descending rank position of y_j will increase by 1. Note that the hyperparameter α controls the precision of the approximation. We then obtain the estimated $\widehat{\tau(j)}$ and subsequently the ApproxNDCG objective:

$$\mathcal{L}_{\text{ApproxNDCG@K}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\big[\sum_{j=1}^{k} G(\psi_j) \cdot D(\widehat{\tau(j)})/N_k\big]. \tag{8}$$

5 Theoretical Analysis

5.1 Optimal Property

Property 5.1. *(Optimal property) When Equation (6) reaches the optimal value, the policy $\pi^*_{\theta,NDCG}$ is aligned with the reward model in terms of the response ranking permutations.*

The proof is shown in Appendix B.1. NDCG achieves its maximum value if and only if the list permutation matches the ground-truth permutation. The distance between NDCG and its maximum value of 1 reflects the current alignment gap between the policy and the reward model.

Proposition 5.1. *For DPO, in pairwise setting, correct ranking $s_w \geq s_l$ is achieved if and only if $\mathcal{L}_{DPO} \leq \log 2$. But in listwise scenarios where list size > 2, this condition on $\mathcal{L}_{DPO}$ no longer guarantees the correct overall ranking.*

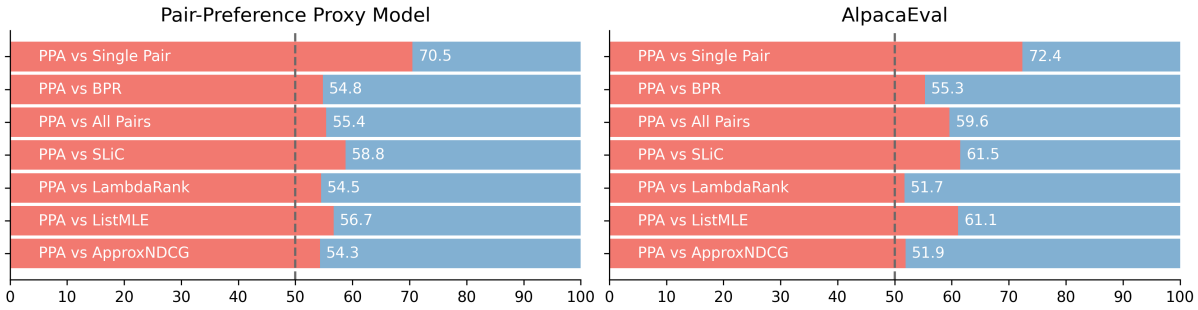


Figure 3: PPA outperforms other approaches on direct comparisons with Mistral-7B. The win rates are derived from comparisons between PPA and other methods on their optimal settings. We employ the Pair-Preference Proxy model on evaluation sets and GPT-4 on AlpacaEval as the judge models.

The proof is provided in Appendix B.2. This proposition indicates the limitations of DPO in handling multi-response scenarios. However, the NDCG-based method monitors the current alignment state and guarantees the correct list ranking.

5.2 Ranking Accuracy

Definition 5.1. *(Ranking Accuracy (Chen et al., 2024)) The ranking accuracy $\mathcal{R}$ of a model π_θ on a pairwise preference datapoint (x, y_w, y_l) is*

$$\mathcal{R}(x, y_w, y_l; \pi_\theta) = \begin{cases} 1 & \pi_\theta(y_w \mid x) \geq \pi_\theta(y_l \mid x), \\ 0 & \textit{otherwise.} \end{cases}$$

Definition 5.2. *(Successful rank flip) A successful rank flip is referred to as the model's ranking of responses shifts to favor the preferred over the non-preferred option:*

$$\pi_{ref}(y_w|x) \leq \pi_{ref}(y_l|x) \rightarrow \pi_\theta(y_w|x) \geq \pi_\theta(y_l|x)$$

Proposition 5.2. *Assume log-likelihood ratio on reference model $X = \log \frac{\pi_{ref}(y_w|x)}{\pi_{ref}(y_l|x)} \sim N(0, \sigma^2)$ and after alignment training $Y = \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} \sim N(\mu_Y, \sigma_Y^2)$, we can get that the probability of successful rank flip of NDCG-based method is greater than DPO, which is $P_{NDCG}(Y > 0|X < 0) \geq P_{DPO}(Y > 0|X < 0)$.*

The proof is provided in Appendix B.3, which is also illustrated in Figure 2. In alignment training, the objective is to increase the probability of the preferred response over the non-preferred one. A higher successful rank flip ratio indicates greater efficiency in achieving alignment.

6 Experiments

Baselines. We employ various pairwise and listwise alignment baselines to explore the connection between LLM alignment and ranking tasks. Their optimization objectives are detailed in Table 5 of the appendix. We introduce three paradigms of positive-negative pairs for DPO on multiple responses. LiPO-λ (Liu et al., 2024) incorporates LambdaRank from the LTR literature, acting as a weighted version of DPO. SLiC and RRHF employ a similar hinge contrastive loss. ListMLE utilizes the Plackett-Luce Model (Plackett, 1975) to represent the likelihood of list permutations. For further information, please see Appendix C.

Datasets. We construct a multi-response dataset named *ListUltraFeedback*[*]. This dataset combines four responses from UltraFeedback and five generated responses from the fine-tuned Llama3-8B model[†] in SimPO (Cui et al., 2023; Meng et al., 2024), all based on the same prompts. All responses are assigned ground truth labels using the Reward Model ArmoRM (Wang et al., 2024). This model is the leading open-source reward model, outperforming both GPT-4 Turbo and GPT-4o in RewardBench (Lambert et al., 2024) at the time of our experiments. To ensure clear distinction between positive and negative samples, while maintaining diversity, we select two responses with the highest scores and two with the lowest. Additionally, we randomly draw four responses from the remaining pool. Details of the dataset are presented in Table 7 of the appendix.

Training Details. We use Qwen2-0.5B, Mistral-7B, and Llama3.1-8B (qwe, 2024; Jiang et al., 2023; Dubey et al., 2024) as our foundation

[*] https://huggingface.co/datasets/NDCG-alignment/ListUltraFeedback

[†] https://huggingface.co/datasets/princeton-nlp/llama3-ultrafeedback-armorm

Method	Type	Proxy Model		General Benchmark		Avg.
		Pair-Preference	Scoring	AlpacaEval	MT-Bench	
Single Pair	Pairwise	60.75	56.86	57.95	52.81	57.09
BPR	Pairwise	60.32	58.33	58.74	55.00	58.10
All Pairs	Pairwise	63.82	60.54	57.23	53.13	58.68
RankNet	Pairwise	62.27	59.04	58.94	54.26	58.63
SLiC	Pairwise	63.31	60.70	61.00	53.75	59.69
LambdaRank	Listwise	62.30	59.04	58.72	55.31	58.84
ListMLE	Listwise	63.03	59.76	57.05	53.13	58.24
ApproxNDCG	Listwise	61.46	58.59	58.16	**55.94**	59.33
PPA	Listwise	**64.25**	**61.36**	**61.64**	53.44	**60.17**

Table 1: The proposed PPA and ApproxNDCG outperform existing baselines across various evaluation benchmarks. The win rates are derived from comparisons between the preference-aligned Qwen2-0.5B and its SFT model. We set $\beta = 0.1$ in Eq 2 for all methods except $\beta = 0.05$ for SLiC to achieve the optimal performance.

models, representing different parameter scales. Following the training pipeline in DPO, Zephyr, and SimPO, we start with supervised fine-tuning (SFT) on UltraChat-200k (Ding et al., 2023). We then apply various pairwise and listwise approaches to align preferences on our multiple response dataset, ListUltraFeedback. Adhering to the settings in HuggingFace Alignment Handbook (Tunstall et al., 2023), we use a learning rate of 5×10^{-7} and a total batch size of 128 for all training processes. The models are trained using the AdamW optimizer (Kingma and Ba, 2014) on 4 Nvidia V100-32G GPUs for Qwen2-0.5B models and 16 Nvidia V100-32G GPUs for Mistral-7B. Unless noted otherwise, we fix $\alpha = 25$ for ApproxNDCG and $\tau = 1$ for PPA to achieve optimal performance, as determined by ablation studies presented in Section 6.2. Both models and datasets are open-sourced, ensuring high transparency and ease of reproduction. Further training details can be found in Appendix E.

Evaluation. The KL-divergence in the original RLHF pipeline is designed to prevent the Policy model from diverging excessively from the SFT model, thus avoiding potential manipulation of the Reward Model. As we employ ArmoRM in the construction of the training dataset, we incorporate various judging models and evaluation benchmarks, such as different Reward models and AlpacaEval (Li et al., 2023) with GPT-4, to reduce the impact of overfitting on ArmoRM. We design 2 pipelines to thoroughly analyze the performance of PPA, using the Win Rate of policy models against SFT models as our primary metric. Details of evaluation datasets are presented in Table 7 of the appendix.

In the *Proxy Model* pipeline, we deploy the Scoring Reward Model ArmoRM[‡] (Wang et al., 2024) and the Pair-Preference Reward Model[§] (Dong et al., 2024) as Proxy Models to calculate the win rate on ListUltraFeedback. Both Proxy models surpass GPT-4 Turbo and GPT-4o in rewarding tasks on RewardBench (Lambert et al., 2024). The Scoring model provides a score in the range $(0, 1)$ for a given prompt and response, while the Pair-Preference model outputs the winner when given a prompt and two responses, offering a more intuitive approach for pairwise comparisons.

In the *General Benchmark* pipeline, we evaluate our models using two widely recognized benchmarks: AlpacaEval (Li et al., 2023) and MT-Bench (Zheng et al., 2023), which assess the model's comprehensive conversational abilities across various questions. Consistent with the original setup, we employ GPT-4 Turbo (Achiam et al., 2023) as the standard judge model to determine which of the two responses exhibits higher quality.

6.1 Main Results

PPA significantly outperforms existing preference optimization baselines. We list win rates of various alignment approaches across diverse evaluation benchmarks in Figure 3, Table 1, and Table 3. Many approaches achieve their best performance with a list size of 8. As shown in Figure 4, PPA consistently outperforms other approaches when $K > 4$, with performance improving as the list size increases. This trend is also evident across different values of β in Table 10 of the appendix. PPA's advantage over pairwise

[‡] https://huggingface.co/RLHFlow/ArmoRM-Llama3-8B-v0.1

[§] https://huggingface.co/RLHFlow/pair-preference-model-LLaMA3-8B

and ListMLE methods lies in its efficiency in improving permutation performance. Traditional contrastive pairwise approaches maximize the likelihood of y_w over y_l, which adversely affects the generation quality of LLMs when high-quality responses are treated as negative samples. In contrast, PPA provides a more holistic approach to handling the relationships between responses.

PPA achieves higher ranking accuracy than DPO and LiPO. As illustrated in Figure 2, PPA demonstrates a higher efficiency on successful rank flips. The difference in model performance is attributed to the distinction between the optimization objectives of the Bradley-Terry-based methods and NDCG-based methods. Furthermore, Figure 2c reveals a significant correlation between the type of rank flip and the log-likelihood ratio of the reference model. Based on this observation, we hypothesize that the log-likelihood ratio of the policy model encodes the conditional probability of the log-likelihood ratio of the reference model. Proposition 5.2 provides a potential theoretical explanation for this improvement.

List Size matters more than the number of pairwise comparisons. To assess the effect of varying response quantities and comparison methods, we conduct empirical studies across different list sizes (i.e., the number of responses the model can access) and comparison methods. Specifically, we test four pairwise comparison methods: Single Pair, BPR, OvW (i.e., Others vs Worst), and All Pairs in Table 5 of the appendix, which differ in the number of comparisons (1 comparison for Single Pair, $K-1$ for BPR and OvW, and $\binom{K}{2}$ for All Pairs). Table 1 illustrates that both BPR and All Pairs methods outperform Single Pair, with no significant difference observed when List Size $= 8$. This trend is particularly pronounced with the Mistral-7B model in Table 11 of the appendix. These findings suggest that the list size plays a more critical role than the specific number of pairwise comparisons.

6.2 Ablation Study

Score Function Scale. The parameter β controls the scaling of the score function (Eq 2) and the deviation from the base reference policy π_{ref}, which significantly influences model performance. Following the existing work setting (Rafailov et al., 2023; Meng et al., 2024; Liu et al., 2024), we search β among $[0.01, 0.05, 0.1, 0.5]$ and conduct sensitivity analysis. Figure 4 shows that all methods achieve their best performance at $\beta = 0.1$ except SLiC. PPA consistently achieves the best performance on both $\beta = 0.05$ and $\beta = 0.1$. Detailed results are in Table 9 of the appendix.

Approximation Tradeoff. The temperature parameter τ controls the approximation accuracy and gradient variance of NeuralNDCG (Pobrotyn and Białobrzeski, 2021). We visualize the values of NDCG and NeuralNDCG on specific data and assess model performance with various τ. The results in Figure 5 reveal that as NeuralNDCG more closely approximates true NDCG, model performance tends to decline. This may occur because training involves multiple high-quality responses with similar ground truth labels. Enforcing responses to conform to NDCG's step-wise structure can reduce the likelihood of good responses.

Additionally, as the approximation accuracy of NeuralNDCG increases, more plateaus appear due to NDCG's inherent step-wise nature. On these plateaus, gradients become zero, preventing model optimization. We visualize the loss landscape in Figure 6 in the Appendix and find that with low τ, the gradients become highly spiky and discontinuous, making optimization challenging and potentially unstable. In contrast, higher τ values yield smoother and more navigable gradients. Further discussion is provided in Appendix D.3. A similar observation is confirmed on ApproxNDCG in Appendix G.

PPA Setup. We perform an ablation study on key components of PPA. As shown in Table 2, we find that (i) when evaluating NDCG@4 for multiple responses with a list size of 8, the performance is comparable to PPA with a list size of 4. This suggests that PPA's effectiveness is more influenced by the list size rather than the k value in NDCG@k. (ii) The choice of gain function: $G_i = 2^{\psi_i} - 1$ or $G_i = \psi_i$, does not significantly impact model performance. The critical factor is that the gain provides the correct order of responses. (iii) Omitting Sinkhorn scaling (Sinkhorn, 1964) on $\widehat{P}$ significantly degrades performance. Without scaling, $\widehat{P}_{\text{sort}}$ may not be column-stochastic, meaning each column may not sum to one. Then the weighted sum of $G(\cdot)$ could disproportionately contribute to the estimated gain

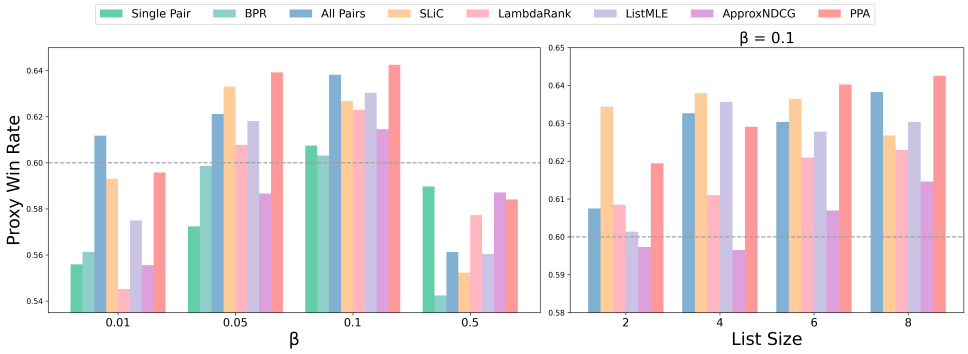


Figure 4: PPA outperforms other methods across different β and list sizes. The Proxy win rates are calculated by Pair-Preference Proxy model by comparing preference-aligned Qwen2-0.5B against its SFT model.

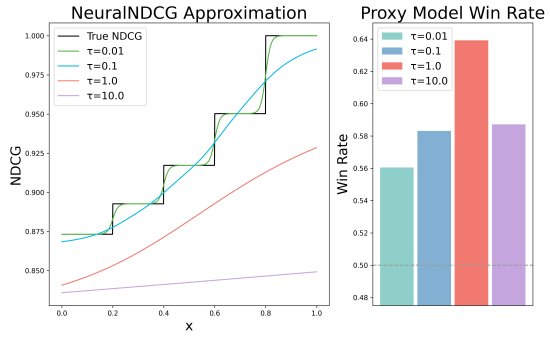


Figure 5: Higher NDCG approximation accuracy does not always lead to better performance. Given ground truth label $\psi = [1.0, 0.8, 0.6, 0.4, 0.2]$ and scores $\mathbf{s} = [x, 0.8, 0.6, 0.4, 0.2]$, an illustration of NeuralNDCG Approximation Accuracy with different τ and Pair-Preference Proxy win rates against SFT.

function $\widehat{G(\cdot)}$ and adversely affect performance.

Method	Pair-Preference	Scoring
All Pairs	63.82	60.54
PPA	**64.25**	**61.36**
Top-4	61.92	59.35
w/o Power	63.49	61.28
w/o Scale	57.32	56.20
$1/\tau$	62.78	59.76
$1/\sqrt{\tau}$	63.16	60.19
$1/\tau^2$	62.47	60.19

Table 2: Ablation results for PPA Setup on Qwen2-0.5B: (Top) original setup; (Middle) ablation study on key components: k-value, gain function, and the Sinkorn Scale function; (Bottom) different discount settings.

Model Scale Up To thoroughly assess the performance of PPA, we employ the Llama3.1-8B and Mistral-7B model as the LLM. Following the SimPO pipeline, we use their Instruct version as the SFT model. They are then aligned with multiple preferences on ListUltraFeedback, and the performance is validated across several benchmarks, as shown in Table 3, Table 11, and Table 13 of the appendix. Hyperparameter details and additional results are provided in Appendix E and F.2.

PPA demonstrates competitive performance on win rates against the SFT model. To clearly illustrate PPA's advantages over other methods, we compare their generated responses and present PPA's win rates in Figures 3 and 8 of the appendix.

Method	Pair-Preference	Scoring	AlpacaEval
All Pairs	72.96	74.39	59.64
SLiC	72.84	75.04	60.20
ListMLE	72.46	74.77	59.83
LambdaRank	71.80	72.74	60.38
PPA	**74.34**	**75.58**	**61.32**

Table 3: Model Scale Up: Our method PPA outperforms other approaches on Llama3.1-8B.

Human Evaluation Human assessments provide crucial validation beyond proxy metrics. To address it, we conducted a comprehensive human evaluation involving 20 raters on the Prolific platform. Each rater compared 15 pairs of model outputs, with a total of 300 pairs evaluated across three matchups (PPA vs SFT, PPA vs DPO, PPA vs LiPO). The pairs were randomly selected to ensure an unbiased assessment.

The results in Table 4 demonstrate that PPA achieves consistent improvements over baselines like SFT, DPO, and LiPO, as confirmed by both human raters and automatic metrics. They also validate the consistency between the reward models and human judgments.

Comparison	PPA vs SFT	PPA vs DPO	PPA vs LiPO
Pair-Preference	70.5	55.4	54.5
Scoring	68.9	55.2	57.0
AlpacaEval	72.4	59.6	51.7
HumanEval	68.0	55.0	54.0

Table 4: Human Evaluation result for comparisons. It shows PPA outperforms other baselines, which aligns with other automatic metrics.

7 Conclusion

We propose Permutative Preference Alignment (PPA) to align listwise human judgments by optimizing the ranking metric NDCG. Empirical studies show that PPA consistently outperforms existing pairwise and listwise baselines across various setups. We identify the potential limitation of Bradley-Terry-based methods like DPO. PPA also demonstrates a higher efficiency in improving ranking accuracy and we propose a theoretical explanation for this improvement.

Limitations

Our study has several limitations and suggests promising directions for future research. In constructing multiple responses, a pre-trained Reward Model serves as the judge model, which might not fully align with real-world human preferences. Future studies can develop more robust data construction methods to ensure responses remain harmless. Additionally, the extensive LTR literature remains underexplored, indicating potential for further research and applications in alignment fields.

Acknowledgements

YW was supported in part by the Office of Naval Research under grant N00014-23-1-2590, the National Science Foundation under grant No. 2231174, No. 2310831, No. 2428059, No. 2435696, No. 2440954, and a Michigan Institute for Data Science Propelling Original Data Science (PODS) grant. MY was supported in part by a Marketing Science Institute fund.

References

2024. Qwen2 technical report.

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

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Angelica Chen, Sadhika Malladi, Lily H. Zhang, Xinyi Chen, Qiuyi Zhang, Rajesh Ranganath, and Kyunghyun Cho. 2024. Preference learning algorithms do not learn preference rankings. *Preprint*, arXiv:2405.19534.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. UltraFeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

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. *Preprint*, arXiv:2305.14233.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf. *Preprint*, arXiv:2405.07863.

Pinar Donmez, Krysta M Svore, and Christopher JC Burges. 2009. On the local optimality of lambdarank. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 460–467.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *Preprint*, arXiv:1511.06939.

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

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. 2013. Learning to rank for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 493–494.

Dahyun Kim, Yungi Kim, Wonho Song, Hyeonwoo Kim, Yunsu Kim, Sanghoon Kim, and Chanjun Park. 2024. sdpo: Don't use your data all at once. *arXiv preprint arXiv:2403.19270*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

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

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

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8716–8723.

Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, Peter J. Liu, and Xuanhui Wang. 2024. Lipo: Listwise preference optimization through learning-to-rank. *Preprint*, arXiv:2402.01878.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2023. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Tong Liu, Xiao Yu, Wenxuan Zhou, Jindong Gu, and Volker Tresp. 2025. Focalpo: Enhancing preference optimizing by focusing on correct preference rankings. *arXiv preprint arXiv:2501.06645*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

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

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization. *arXiv preprint arXiv:2403.19159*.

Robin L Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202.

Przemysław Pobrotyn and Radosław Białobrzeski. 2021. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *Preprint*, arXiv:2102.07831.

Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval*, 13:375–397.

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. *Preprint*, arXiv:2305.18290.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. Bpr: Bayesian personalized ranking from implicit feedback. *Preprint*, arXiv:1205.2618.

Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacroce, Ahmed Awadallah, and Tengyang Xie. 2024. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*.

Richard Sinkhorn. 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. *Preprint*, arXiv:2306.17492.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Shengyi Huang, Kashif Rasul, Alexander M. Rush, and Thomas Wolf. 2023. The alignment handbook. `https://github.com/huggingface/alignment-handbook`.

Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. 2009. Learning to rank by optimizing ndcg measure. *Advances in neural information processing systems*, 22.

Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008a. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008b. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. 2023. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*.

Yueqin Yin, Zhendong Wang, Yi Gu, Hai Huang, Weizhu Chen, and Mingyuan Zhou. 2024. Relative preference optimization: Enhancing llm alignment through contrasting responses across identical and diverse prompts. *arXiv preprint arXiv:2402.10958*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears. *Preprint*, arXiv:2304.05302.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *Preprint*, arXiv:2305.10425.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Mingye Zhu, Yi Liu, Lei Zhang, Junbo Guo, and Zhendong Mao. 2024. Lire: listwise reward enhancement for preference alignment. *arXiv preprint arXiv:2405.13516*.

A Related Work

Pairwise Preference Optimization Direct Preference Optimization (DPO) (Rafailov et al., 2023) removes the necessity for an explicit reward model within the RLHF framework by introducing a novel algorithm to compute reward scores for each response. Similar to RLHF, DPO uses the Bradley-Terry (BT) model (Bradley and Terry, 1952) to align binary human preferences in a contrastive manner. Subsequent research, including methods like IPO, KTO, RPO, SimPO, and others (Liu et al., 2023; Xu et al., 2023; Azar et al., 2024; Yin et al., 2024; Ethayarajh et al., 2024; Hong et al., 2024; Park et al., 2024; Meng et al., 2024), focus on refining the reward function and the BT model to enhance performance and simplify the process. Additionally, iterative methods are developed to align pairwise preferences with a dynamic reference model (Rosset et al., 2024; Pang et al., 2024; Kim et al., 2024; Yuan et al., 2024). They classify preferred responses y_w as positive samples and non-preferred responses y_l as negative samples. They infer relative quality rankings of responses by maximizing the pairwise choice probability of $r(x, y_w)$ over $r(x, y_l)$. However, under multiple response scenarios, they focus on maximizing the average value of pairwise contrastive probability. It fails to guarantee a hard constraint that every individual pair is correctly aligned with ground truth labels. FocalPO (Liu et al., 2025) assigns greater weights to more informative ranking pairs, which shares similar insights with the NDCG metric.

Multiple Responses Alignment Recent research has introduced simple and efficient methods to align human preferences across multiple responses. These approaches expand candidate responses from various LLMs such as ChatGPT, Alpaca, and GPT-4, assigning rewards via reward models or human feedback. RRHF(Yuan et al., 2023) employs the same hinge objective as SLiC (Zhao et al., 2023) on multiple responses through pairwise comparisons. LiPO-λ (Liu et al., 2024) incorporates LambdaRank (Donmez et al., 2009) where higher-quality responses against lower-quality ones receive greater weights, acting as a weighted version of DPO. However, when handling high-quality response pairs, incorrectly classifying one of them as the negative sample and minimizing its likelihood can adversely affect LLM generation quality. Listwise methods offer a more nuanced approach to handling relationships between responses. DPO-PL (Rafailov et al., 2023) and PRO (Song et al., 2024) employ the same PL framework (Plackett, 1975) but differ in their reward functions. LIRE (Zhu et al., 2024) calculates softmax probabilities with a consistent denominator and multiplies them by corresponding rewards, functioning as a pointwise algorithm since permutations do not alter loss values. Despite their potential, current listwise techniques are not yet state-of-the-art in the learning-to-rank (LTR) literature, indicating a need for further research.

Learning to Rank (LTR) LTR involves a set of machine learning techniques widely applied in information retrieval, web search, and recommender systems (Liu et al., 2009; Karatzoglou et al., 2013; Hidasi et al., 2016; Li et al., 2024). The goal is to train a ranking model by learning a scoring function $s = f(x, y)$ that assigns scores to elements for ranking purposes. The loss is computed by comparing the current permutation with the ground truth, which updates the model parameters θ. Loss functions in LTR are generally categorized into three types: pointwise, pairwise, and listwise. Pointwise and pairwise methods convert the ranking task into classification problems, often overlooking the inherent structure of ordered data. Conversely, listwise approaches (Xia et al., 2008a) directly tackle the ranking problem by considering entire ranking lists as training instances. This approach fully exploits the relative proximities within multiple responses, enhancing the understanding of the ranking relationships.

B Proof of Theoretical Analysis

B.1 Proof of Property 5.1

Assume access to a list of ground truth labels in descending order $\Psi = \{\psi_1, ..., \psi_K\}$, where $\psi_i \geq \psi_j$ if $i < j$. Now we have a score vector $\mathbf{s} = \{s_1, ..., s_k\}$, the descending rank position of s_i is denoted by

$$\tau(i) = 1 + \sum_{j=1}^{k} \mathbb{I}_{s_i < s_j}.$$

According to the definition of NDCG Eq 4, the maximum NDCG value is achieved when $\tau(i) = i$, which is equivalent to $s_i^* \geq s_j^*$ if $i < j$. The permutation of $\mathbf{s}^*$ is the same as the permutation

of ground truth labels $\Psi = \{\psi_1, ..., \psi_K\}$, where $\psi_i \geq \psi_j$ if $i < j$. Then we can say the current policy model π^*_{NDCG} is aligned with the reward model RM in terms of response permutations.

B.2 Proof of Proposition 5.1

Under pairwise (x, y_w, y_l) scenario,

$$\mathcal{L}_{\text{DPO}} = -\log \sigma(s_w - s_l).$$

When $s_w = s_l$, $\mathcal{L}_{\text{DPO}} = \log 2$. As $\sigma(x)$ is an increasing function, it is easy to derive that $\mathcal{L}_{\text{DPO}} \leq \log 2 \Leftrightarrow s_w \geq s_l$.

But when list size > 2, the condition of $\mathcal{L}_{\text{DPO}} \leq \log 2$ no longer guarantees the correct overall ranking. Here we take a triplet datapoint (x, y_1, y_2, y_3) where $\psi_1 \geq \psi_2 \geq \psi_3$ for example:

$$\begin{aligned}
&\mathcal{L}_{\text{DPO}}(x, y_1, y_2, y_3|\pi_\theta) \\
&= -\frac{1}{3} \times [\log \sigma(s_1 - s_2) + \log \sigma(s_1 - s_3) + \log \sigma(s_2 - s_3)] \\
&= -\frac{1}{3} \times [\log \frac{e^{s_1}}{e^{s_1} + e^{s_2}} + \log \frac{e^{s_1}}{e^{s_1} + e^{s_3}} + \log \frac{e^{s_2}}{e^{s_2} + e^{s_3}}]
\end{aligned}$$

We assume $(s_1, s_2, s_3) = (0.7, 0.5, 0.6)$ and we can calculate the DPO loss:

$$\begin{aligned}
\mathcal{L}_{\text{DPO}} &= -\frac{1}{3} \times [\log \sigma(0.2) + \log \sigma(0.1) + \log \sigma(-0.1)] \\
&= 0.662 < \log 2
\end{aligned}$$

But obviously $s_1 > s_3 > s_2$ is not a correct list ranking. In this case, even if the DPO loss falls below a certain threshold, it does not guarantee an accurate list ranking. This limitation arises because the optimization process focuses on adjusting the scores s_i and s_j to increase the overall expected value but does not enforce hard constraints like $s_i > s_j$ for each individual pair. The DPO objective adjusts $s_i - s_j$ in a soft, overall sense but may not result in all differences being positive.

B.3 Proof of Proposition 5.2

We assume the log-likelihood ratios of all models follow normal distributions. Before alignment training, it is rational to hypothesize that the reference model cannot always distinguish and prioritize the preferred response y_w, so we set 0 to its mean:

$$X = \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} \sim N(0, \sigma_X^2)$$

Then after training, the distribution of log-likelihood ratio shifts on the policy model:

$$Y = \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} \sim N(\mu_Y, \sigma_Y^2),$$

As X and Y both follow normal distributions, we can use an independent variable c from normal distributions to represent the *training effect*. Since the parameters of independent normal distributions are additive, it is easier to write as follows

$$Y = X + c,\ c \sim N(\mu_c, \tau_c^2),$$

where $\mu_c = \mu_Y, \tau_c^2 = \sigma_Y^2 - \sigma_X^2$.

As $s_w = \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$, we have

$$\begin{aligned}
c = Y - X &= \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} - \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} \\
&= \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} = \frac{s_w - s_l}{\beta}
\end{aligned}$$

Under pairwise scenarios, DPO-PL is equivalent to DPO-BT. The training objectives of DPO and NDCG are as follows:

$$\begin{aligned}
&\text{DPO}: \\
&\quad \max \mathbb{E}_{x,\mathbf{y},\psi\sim\mathcal{D}}(s_w - s_l) \equiv \max E(c) \\
&\text{NDCG}: \\
&\quad \max \mathbb{E}_{x,\mathbf{y},\psi\sim\mathcal{D}}(\mathbb{I}_{s_w > s_l}) \equiv \max E(P(c > 0))
\end{aligned} \tag{9}$$

Based on the different training objectives of DPO and NDCG, we make the following assumptions. For NDCG method, the training objective is maximizing the number of cases where $s_w > s_l$, we assume a small mean μ_{NDCG} that is just enough to make s_w likely to be greater than s_l. The variance τ^2_{NDCG} is also small, leading to a distribution concentrated around its mean. The assumptions above can result in a high probability of $P(c > 0)$, which is aligned with NDCG's objective. During training, the algorithm adjust s_w and s_l to reduce the variability τ^2_{NDCG} of c, keeping differences small but positive.

Regarding DPO (same for LiPO and other pairwise contrastive methods), we assume a larger positive mean μ_{DPO} which increases the overall score margin. c of DPO has a large variance τ^2_{DPO} as well, allowing for more variability in the differences. During training, the distributions of s_w and s_l are more spread out, leading to larger differences in $s_w - s_l$. The algorithm accepts higher variability and occasional negative differences to achieve a higher overall sum of $s_w - s_l$.

Figure 2c indicates that log-likelihood ratio of the policy model is conditional on that of the refer-

ence model then we have

$$\text{NDCG} : \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} | \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} \sim N(\log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} + \mu_{\text{NDCG}}, \sigma^2 + \tau^2_{\text{NDCG}})$$

$$\text{DPO} : \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} | \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} \sim N(\log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} + \mu_{\text{DPO}}, \sigma^2 + \tau^2_{\text{DPO}})$$

where $\mu_{\text{NDCG}} \leq \mu_{\text{DPO}}$, $\tau^2_{\text{NDCG}} \leq \tau^2_{\text{DPO}}$, $P(c_{\text{NDCG}} > 0) \geq P(c_{\text{DPO}} > 0)$.

As $c \sim N(\mu_c, \tau_c^2)$, we have $z = (c - \mu_c)/\tau_c \sim N(0, 1)$, then we can derive

$$P(c > 0) = P(z > -\frac{\mu_c}{\tau_c}) = P(z < \frac{\mu_c}{\tau_c}) = \Phi(\frac{\mu_c}{\tau_c}),$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function (CDF), which is an increasing function. Under assumptions above $P(c_{\text{NDCG}} > 0) \geq P(c_{\text{DPO}} > 0)$, we have

$$\frac{\mu_{\text{NDCG}}}{\tau_{\text{NDCG}}} \geq \frac{\mu_{\text{DPO}}}{\tau_{\text{DPO}}} \tag{10}$$

By definition, we refer to turning an incorrect pair from the reference model (i.e., $\frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} < 1$) into a correct pair in the policy model (i.e., $\frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} > 1$) as a successful rank flip. Then we can represent the probability of it via the annotations above:

$$P(\log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} > 0 | \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} < 0) := P(Y > 0|X < 0) = \frac{P(Y > 0, X < 0)}{P(X < 0)}.$$

Since $X \sim N(0, \sigma^2)$, $P(X < 0) = 0.5$, so:

$$P(Y > 0|X < 0) = 2P(Y > 0, X < 0) = 2\int_{x=-\infty}^{0} P(Y > 0|X = x) f_X(x)\, dx.$$

Given $X = x, Y = x + c$,

$$P(Y > 0|X = x) = P(c > -x) = P(z > \frac{-x - \mu_c}{\tau_c}) = \Phi(\frac{x + \mu_c}{\tau_c}).$$

As $X \sim N(0, \sigma^2)$, we replace x with parameter $s = x/\sigma \sim N(0, 1)$, then we have:

$$P(Y > 0|X < 0) = 2\int_{s=-\infty}^{0} \Phi(\frac{\sigma s + \mu_c}{\tau_c}) \phi(s)\, ds, \tag{11}$$

where $\phi(s)$ is the PDF of the standard normal distribution and $\Phi(\cdot)$ is the standard normal's CDF. Since $\Phi(\cdot)$ is an increasing function and in Eq 10 we have $\mu_{\text{NDCG}}/\tau_{\text{NDCG}} \geq \mu_{\text{DPO}}/\tau_{\text{DPO}}$ and $\tau_{\text{NDCG}} \leq \tau_{\text{DPO}}$, we have

$$\Phi(\frac{\sigma s + \mu_{\text{NDCG}}}{\tau_{\text{NDCG}}}) \geq \Phi(\frac{\sigma s + \mu_{\text{DPO}}}{\tau_{\text{DPO}}}),$$

Finally we can derive:

$$P_{\text{NDCG}}(Y > 0|X < 0) \geq P_{\text{DPO}}(Y > 0|X < 0)$$

C Details of Baselines

Table 5 shows the types and objectives of the baselines we consider in the empirical study.

To ensure variable consistency and comparability of experiments, we choose the original DPO algorithm as our reward score function Eq 2 and pairwise baseline method and assess its performance in both binary-response and multi-response scenarios.

DPO-BT In detail, we implement four variants of the original sigmoid-based pairwise DPO based on the Bradley-Terry (BT) methods while aligning multiple responses. The first one is **Single Pair** paradigm, where we compare only the highest-scoring and lowest-scoring responses, which is equivalent to the original DPO in the pairwise dataset scenario.

$$\mathcal{L}_{\text{Single Pair}}(\pi_\theta; \pi_{\text{ref}}) = - \mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}} \left[\log \sigma\left(s_1 - s_K\right)\right], \tag{12}$$

Then we introduce the **Bayesian Personalized Ranking (BPR)** (Rendle et al., 2012) algorithm that computes the response with the highest score against all other negative responses based on Bayes' theorem[¶], which is widely used in recommender system (Hidasi et al., 2016).

$$\mathcal{L}_{\text{BPR}}(\pi_\theta; \pi_{\text{ref}}) = - \mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}} \left[\frac{1}{K-1} \sum_{j\neq 1}^{K} \log \sigma\left(s_1 - s_j\right)\right], \tag{13}$$

[¶]The **BPR** variant Eq 13 can be viewed as the expected loss function in the following scenario: we have a multiple responses dataset, but we only retain the highest-scoring response and randomly select one from the remaining. Finally, we construct a binary responses dataset for pairwise preference optimization, which is a widely used method for building pairwise datasets (Tunstall et al., 2023; Meng et al., 2024).

Method	Type	Objective
DPO - Single Pair (12)	Pairwise	$-\log\sigma\left(s_1 - s_K\right)$
DPO - BPR (13)	Pairwise	$-\frac{1}{K-1}\sum_{j\neq 1}^{K}\log\sigma\left(s_1 - s_j\right)$
DPO - Others vs Worst (14)	Pairwise	$-\frac{1}{K-1}\sum_{j\neq K}^{K}\log\sigma\left(s_j - s_K\right)$
DPO - All Pairs (15)	Pairwise	$-\binom{K}{2}^{-1}\sum_{\psi_i>\psi_j}\log\sigma\left(s_i - s_j\right)$
SLiC (17)	Pairwise	$-\binom{K}{2}^{-1}\sum_{\psi_i>\psi_j}\max(0, 1-(s_i - s_j))$
LambdaRank (16)	Listwise	$-\binom{K}{2}^{-1}\sum_{\psi_i>\psi_j}\Delta_{i,j}\log\sigma\left(s_i - s_j\right)$ where $\Delta_{i,j} = \lvert G_i - G_j\rvert\cdot\lvert D(\tau(i)) - D(\tau(j))\rvert$
ListMLE (18)	Listwise	$-\log\prod_{k=1}^{K}\frac{exp(s_k)}{\sum_{j=k}^{K}exp(s_j)}$
ApproxNDCG (8)	Listwise	$-N_k^{-1}\sum_{j=1}^{k}G(\psi_j)\cdot D(\widehat{\tau(j)})$
PPA (6)	Listwise	$-N_k^{-1}\sum_{j=1}^{k}(\text{scale}(\widehat{P})\cdot G(\mathbf{\Psi}))_j\cdot D(j)$

Table 5: Pairwise and listwise baselines given multiple-response data $\mathcal{D} = (x, \mathbf{Y}, \mathbf{\Psi})$.

The third one is **Others vs Worst**, which calculates other responses against the worst one:

$$\mathcal{L}_{\text{OvW}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\left[\frac{1}{K-1}\sum_{j\neq K}^{K}\log\sigma\left(s_j - s_K\right)\right], \tag{14}$$

In the last BT variant, we consider all pairs that can be formed from K responses, which is similar to PRO (Song et al., 2024). This approach allows the model to gain more comprehensive information than the aforementioned methods, including preference differences among intermediate responses, which is referred to as **All Pairs**:

$$\mathcal{L}_{\text{All Pairs}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\left[\frac{1}{\binom{K}{2}}\sum_{\psi_i>\psi_j}\log\sigma\left(s_i - s_j\right)\right], \tag{15}$$

where $\binom{K}{2}$ denotes the number of combinations choosing 2 out of K elements.

LiPO-λ Deriving from the LambdaRank (Donmez et al., 2009), the objective of LiPO-λ (Liu et al., 2024) can be written as follows:

$$\mathcal{L}_{\text{LambdaRank}}(\pi_\theta; \pi_{\text{ref}}, \beta) = -\mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\left[\frac{1}{\binom{K}{2}}\sum_{\psi_i>\psi_j}\Delta_{i,j}\log\sigma\left(s_i - s_j\right)\right], \tag{16}$$

where $\Delta_{i,j} = |G_i - G_j|\cdot\left|\frac{1}{D(\tau(i))} - \frac{1}{D(\tau(j))}\right|$. $\Delta_{i,j}$ is referred to as the Lambda weight. G is known as a gain function, with $G_i = 2^{\psi_i} - 1$ being a commonly used example. The function D serves as a discount function, with $D(\tau(i)) = \log_2(1 + \tau(i))$, where $\tau(i)$ is the same as in Eq 3.

SLiC Following the analogous objectives proposed in RRHF (Yuan et al., 2023) and SLiC (Zhao et al., 2023), we integrate the pairwise ReLU-based loss as one of our baselines:

$$\mathcal{L}_{\text{SLiC}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\left[\frac{1}{\binom{K}{2}}\sum_{\psi_i>\psi_j}\max(0, 1-(s_i - s_j))\right], \tag{17}$$

DPO-PL The DPO objective can also be derived under the Plackett-Luce Model (Plackett, 1975; **?**) in a listwise manner, which is equivalent to the ListMLE (Xia et al., 2008b) method:

$$\mathcal{L}_{\text{ListMLE}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,\mathbf{Y},\mathbf{\Psi})\sim\mathcal{D}}\left[\log\prod_{k=1}^{K}\frac{exp(s_k)}{\sum_{j=k}^{K}exp(s_j)}\right], \tag{18}$$

D NeuralSort Details

D.1 NeuralSort relaxation

To approximate the sorting operator, we need to approximate this permutation matrix. In NeuralSort (Grover et al., 2019), the permutation matrix is approximated using a unimodal row stochastic

matrix $\widehat{P}_{\text{sort}(\mathbf{s})}(\tau)$, defined as:

$$\widehat{P}_{\text{sort}(\mathbf{s})}[i,:](\tau) = \text{softmax}\left[\frac{((n+1-2i)\mathbf{s} - A_{\mathbf{s}}\mathbf{1})}{\tau}\right]. \tag{19}$$

Here, $A_{\mathbf{s}}$ is the matrix of absolute pairwise differences of elements in $\mathbf{s}$, where $A_{\mathbf{s}}[i,j] = |s_i - s_j|$, and $\mathbf{1}$ is a column vector of ones. The row of $\widehat{P}_{\text{sort}(\mathbf{s})}$ always sums to one. The temperature parameter $\tau > 0$ controls the accuracy of the approximation. Lower values of τ yield better approximations but increase gradient variance:

$$\lim_{\tau \to 0} \widehat{P}_{\text{sort}(\mathbf{s})}(\tau) = P_{\text{sort}(\mathbf{s})}. \tag{20}$$

A specific simulation is shown in Table 6 of the appendix. For simplicity, we denote $\widehat{P}_{\text{sort}(\mathbf{s})}$ as $\widehat{P}$.

D.2 NDCG Approximation

Given the input ground truth labels $\mathbf{\Psi} = [5,4,3,2]^T$ and scores $\mathbf{s} = [9,1,5,2]^T$, the descending order of $\mathbf{\Psi}$ based on the current reward scores $\mathbf{s}$ is $\tau = [1,4,2,3]^T$. According to the formula introduced in Eq 3:

$$\text{DCG@4} = \sum_{j=1}^{k} G(\psi_j) \cdot D(\tau(j)) =$$
$$\frac{G(5)}{\log_2(1+1)} + \frac{G(4)}{\log_2(1+4)} + \frac{G(3)}{\log_2(1+2)} + \frac{G(2)}{\log_2(1+3)}$$

Building upon the preliminaries defined in (Grover et al., 2019), consider an n-dimensional permutation $\mathbf{z} = [z_1, z_2, \ldots, z_n]^T$, which is a list of unique indices from the set $1, 2, \ldots, n$. Each permutation $\mathbf{z}$ has a corresponding permutation matrix $P_{\mathbf{z}} \in 0, 1^{n \times n}$, with entries defined as follows:

$$P_{\mathbf{z}}[i,j] = \begin{cases} 1 & \text{if } j = z_i \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Let $\mathbb{Z}_n$ denote the set containing all $n!$ possible permutations within the symmetric group. We define the sort : $\mathbb{R}^n \to \mathbb{Z}_n$ operator as a function that maps n real-valued inputs to a permutation representing these inputs in descending order.

The $\text{sort}(\mathbf{s}) = [1,3,4,2]^T$ since the largest element is at the first index, the second largest element is at the third index, and so on. We can obtain the sorted vector simply via $P_{\text{sort}(\mathbf{s})} \cdot \mathbf{s}$:

$$P_{\text{sort}(\mathbf{s})} \cdot \mathbf{s} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 9 \\ 1 \\ 5 \\ 2 \end{pmatrix} = \begin{pmatrix} 9 \\ 5 \\ 2 \\ 1 \end{pmatrix} \tag{22}$$

Here we demonstrate the results by conducting NeuralSort Relaxation Eq 19 with different τ. When we integrate the NeuralNDCG formula in Eq 5, ideally, $\lim_{\tau \to 0} \widehat{P}_{\text{sort}(\mathbf{s})}(\tau) = P_{\text{sort}(\mathbf{s})}$, yielding the following result:

	$\widehat{P_{\text{sort}(\mathbf{s})}} \cdot \mathbf{s}$			
$\lim_{\tau \to 0}$	9	5	2	1
$\tau = 0.01$	9.0000	5.0000	2.0000	1.0000
$\tau = 0.1$	9.0000	5.0000	2.0000	1.0000
$\tau = 1.0$	8.9282	4.9420	1.8604	1.2643
$\tau = 10.0$	6.6862	4.8452	3.2129	2.2557

Table 6: Illustration of Sorting Operation of ground truth labels $\mathbf{\Psi} = [5,4,3,2]^T$ and scores $\mathbf{s} = [9,1,5,2]^T$ via NeuralSort (Grover et al., 2019) with different τ.

$$\widehat{G} = P_{\text{sort}(\mathbf{s})} \cdot \text{G}(\mathbf{\Psi}) =$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \text{G}(5) \\ \text{G}(4) \\ \text{G}(3) \\ \text{G}(2) \end{pmatrix} = \begin{pmatrix} \text{G}(5) \\ \text{G}(3) \\ \text{G}(2) \\ \text{G}(4) \end{pmatrix}$$

Then,

$$\text{NeuralDCG@4} = \sum_{j=1}^{k} (\widehat{G})_j \cdot D(j) =$$
$$\frac{G(5)}{\log_2(1+1)} + \frac{G(3)}{\log_2(1+2)} + \frac{G(2)}{\log_2(1+3)} + \frac{G(4)}{\log_2(1+4)}$$

which can be easily seen to be the same as DCG@4.

D.3 Approximation Tradeoff Analysis

We visualize the loss landscape of NeuralNDCG in Figure 6 and ApproxNDCG in Figure 7. We find that with low τ and α, the gradients become highly spiky and discontinuous, making optimization challenging and potentially unstable. In contrast, higher τ and α values yield smoother and more navigable gradients. However, when τ is too high (e.g., $\tau = 10.0$), the gradient curve becomes overly flat and the approximation error increases significantly, which can also hinder model performance. We observe this phenomenon consistently in both NDCG-based methods, NeuralNDCG and ApproxNDCG.

D.4 Pseudo Code of Neural Sort

The code for NeuralSort Eq 19 is provided below.

```
import torch
import torch.nn.functional as F
```

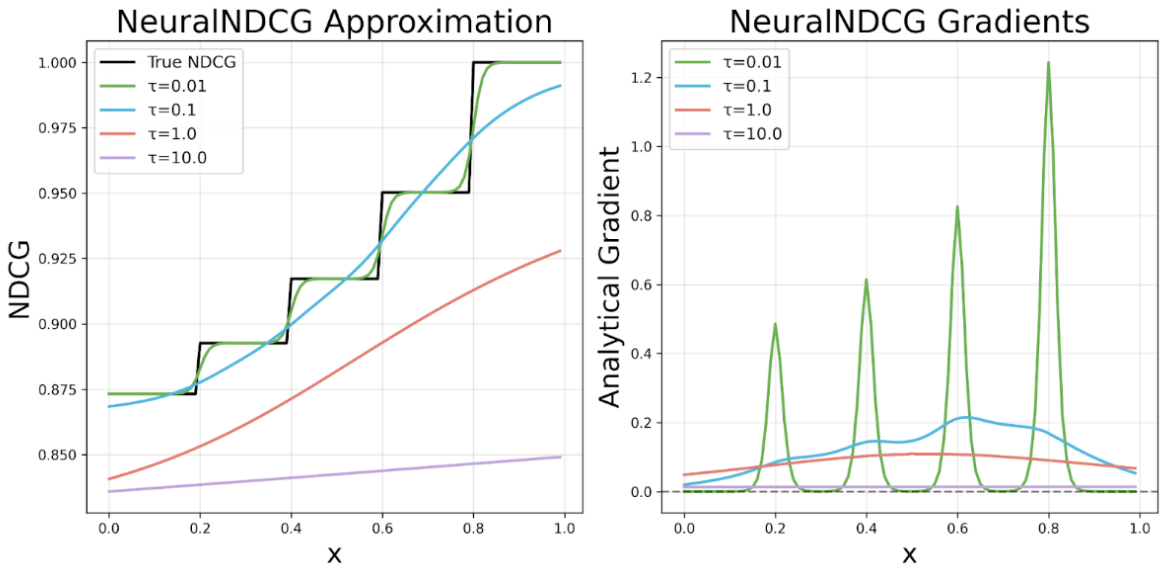


Figure 6: Given ground truth label $\psi = [1.0, 0.8, 0.6, 0.4, 0.2]$, the scores $\mathbf{s} = [x, 0.8, 0.6, 0.4, 0.2]$ and fix $\beta = 0.1$, we visualize the PPA Approximation Accuracy with different τ and its corresponding gradients.

```
def neuralsort(s, tau=1):
    #s.shape = [batch_size, list_size]
    s=s.unsqueeze(2)

    #A_s[i,j] = |s[i] - s[j]|
    A_s = s - s.transpose(1, 2)
    A_s = torch.abs(A_s)

    #B=A_s*ones
    n = s.size(1)
    one = torch.ones((n, 1),dtype=
        torch.float)
    B = torch.matmul(A_s, one @
        one.transpose(0, 1))

    #C=(n+1-2i)*s
    K = torch.arange(1, n + 1,dtype=
        torch.float)
    C = torch.matmul(s, (n + 1 - 2 * K)
        .unsqueeze(0))

  #P= softmax(((n+1-2i)*s-A_s*ones)/tau)
    P = (C - B).transpose(1, 2)
    P = F.softmax(P / tau, dim=-1)

    return P
```

Given the score $\mathbf{s} = [9, 1, 5, 2]^T$ provided in Appendix D.2, we have

$$A_s = \begin{pmatrix} 0 & 8 & 4 & 7 \\ 8 & 0 & 4 & 1 \\ 4 & 4 & 0 & 3 \\ 7 & 1 & 3 & 0 \end{pmatrix},$$

$$B = A_s \cdot \mathbf{1}_{k\times k} = \begin{pmatrix} 19 & 19 & 19 & 19 \\ 13 & 13 & 13 & 13 \\ 11 & 11 & 11 & 11 \\ 11 & 11 & 11 & 11 \end{pmatrix}$$

$$C = [(n+1-2i) * \mathbf{s}] = \begin{pmatrix} 27 & 9 & -9 & -27 \\ 3 & 1 & -1 & -3 \\ 15 & 5 & -5 & -15 \\ 6 & 2 & -2 & -6 \end{pmatrix}$$

Based on Eq 19, we can get $\widehat{P}_{\text{sort}}(s)$

$$\widehat{P}_{\text{sort}}(s) = \text{softmax}\left[\frac{C-B}{\tau}\right] = \begin{pmatrix} 0.98 & 1.5e-8 & 0.018 & 2.2e-6 \\ 0.017 & 2.3e-3 & 0.93 & 0.047 \\ 2.2e-7 & 0.26 & 0.035 & 0.71 \\ 6.8e-14 & 0.73 & 3.3e-5 & 0.27 \end{pmatrix}$$

Finally, we can get the permutation of the sorted score vector as shown in Table 6:

$$\widehat{P}_{\text{sort}}(s) \cdot \mathbf{s} = \begin{pmatrix} 8.9282 & 4.9420 & 1.8604 & 1.2691 \end{pmatrix}$$

We also show the sum of columns and rows:

$$\text{column sum}: \begin{pmatrix} 0.9991 & 0.9928 & 0.9872 & 1.0208 \end{pmatrix}$$

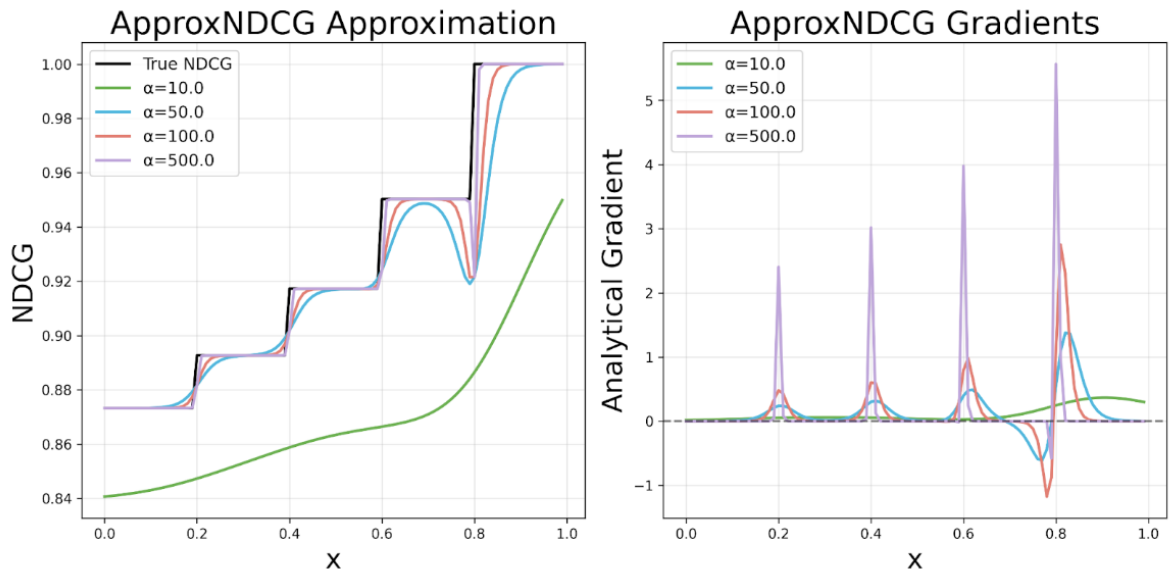


Figure 7: Given ground truth label $\psi = [1.0, 0.8, 0.6, 0.4, 0.2]$, the scores $\mathbf{s} = [x, 0.8, 0.6, 0.4, 0.2]$ and fix $\beta = 0.1$, we visualize the ApproxNDCG Approximation Accuracy with different α and its corresponding gradients.

$$\text{row sum} : \begin{pmatrix} 1.0000 & 1.0000 & 1.0000 & 1.0000 \end{pmatrix}^T$$

As discussed above, $\widehat{P}_{\text{sort}}(s)$ is not column-stochastic, meaning each column may not sum to one. This can cause some $G(\psi_i)$ to contribute to the overall loss objective disproportionately and adversely affect model performance. The ablation studies are shown in Table 2.

E Training Details

The detailed training hyperparameters of Mistral-7B are shown in Table 8.

Since Nvidia v100 is incompatible with the bf16 type, we use fp16 for mixed precision in deepspeed configuration. Notably, as the ListMLE method doesn't have normalization, it will encounter loss scaling errors with mixed precision settings.

F Supplementary Results

F.1 Proxy Models Results

The supplementary results of the Proxy Model Win Rate are shown in Table 9 and Table 10. For PPA, we fix $\tau = 1.0$. For ApproxNDCG, we fix $\alpha = 25$ because it is the parameter $\alpha \cdot \beta$ that controls the approximation accuracy of the sigmoid function in Eq 7.

F.2 Supplementary Results for Mistral-7B

We observe that decreasing the hyperparameter β may increase the performance when language models scale up to 7B parameters. All methods achieve their best performance with $\beta = 0.05$ except for Single Pair with $\beta = 0.01$. Our approach PPA consistently achieves the best overall performance, shown in Table 12.

To further explore the distribution shift during human preference alignment, we demonstrate the score distribution of all methods of which scores are assigned by the Reward model ArmoRM (Wang et al., 2024) in Figure 8. The PPA method causes the reward score distribution to shift more significantly to the right, resulting in fewer instances at lower scores. Consequently, when compared to the SFT model, its win rate is not as high as methods like All Pairs, SLiC, and ListMLE. However, it can outperform these methods in direct comparisons.

G ApproxNDCG Analysis

In ApproxNDCG, we observe similar results to NeuralNDCG; the model achieves optimal performance only when the approximation accuracy reaches a certain threshold. First, we prove that the Accuracy of ApproxNDCG is relevant to the multiplication of α and β when we employ the score function in Eq 2.

First, let's define the term for the probability ratio as R_{ji}:

$$R_{ji} = \frac{\pi_\theta(y_j|x)\pi_{ref}(y_i|x)}{\pi_{ref}(y_j|x)\pi_\theta(y_i|x)}$$

Note that from the properties of logarithms, we

Datasets	Examples	Judge Model	Notes
UltraChat200k	208k	-	SFT
ListUltraFeedback$_{\text{train}}$	59.9k	-	Permutative Preference Alignment
ListUltraFeedback$_{\text{test}}$	1968	RLHFlow Pair-Preference ArmoRM	Pair-Preference win rates Scoring win rates
AlpacaEval	805	GPT-4 Turbo	Pair-Preference win rates
MT-Bench	80	GPT-4 Turbo	Scoring win rates

Table 7: Details of training datasets and evaluation datasets.

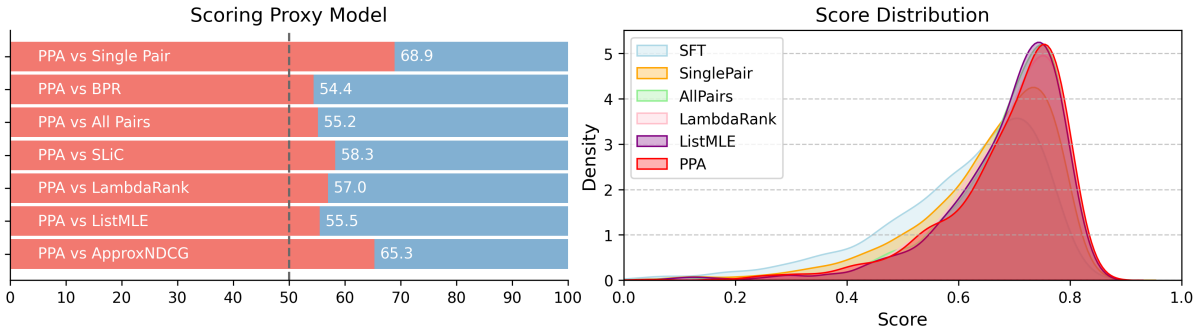


Figure 8: PPA demonstrates superior performance compared to other methods in Scoring Proxy model win rates on Mistral-7B, while also shifting the distribution of response reward scores more significantly to the right (i.e., increasing reward scores).

Hyperparameters	value
Mini Batch	1
Gradient Accumulation Steps	8
GPUs	16×Nvidia V100-32G
Total Batch Size	128
Learning Rate	5e-7
Epochs	1
Max Prompt Length	512
Max Total Length	1024
Optimizer	AdamW
LR Scheduler	Cosine
Warm up Ratio	0.1
Random Seed	42
β	{0.01, **0.05***, 0.1}
τ for PPA	1.0
α for ApproxNDCG	25
Sampling Temperature	0
Pair-Preference Proxy Model	RLHFlow Pair-Preference
Scoring Proxy Model	ArmoRM
GPT Judge	GPT-4-Turbo
AlpacaEval Judge	alpaca_eval_gpt4_turbo_fn

Table 8: Training hyperparameters for Mistral-7B and Llama3.1-8B models.

have:

$$\log(R_{ji}) = \log \frac{\pi_\theta(y_j|x)}{\pi_{ref}(y_j|x)} - \log \frac{\pi_\theta(y_i|x)}{\pi_{ref}(y_i|x)}$$

Using this simplification, the derivation becomes much more compact:

$$\begin{aligned}
\widehat{\tau(j)} &= 1 + \sum_{i \neq j} \frac{\exp\left(-\alpha(s_j - s_i)\right)}{1 + \exp\left(-\alpha(s_j - s_i)\right)} \\
&= 1 + \sum_{i \neq j} \frac{1}{1 + \exp(\alpha(s_j - s_i))} \\
&= 1 + \sum_{i \neq j} \frac{1}{1 + \exp(\alpha\beta \log R_{ji})} \\
&= 1 + \sum_{i \neq j} \frac{1}{1 + (R_{ji})^{\alpha\beta}} \\
&= \dots
\end{aligned} \tag{23}$$

Then, we illustrate the Approximation accuracy and model performance of ApproxNDCG with different hyperparameters $\alpha \cdot \beta$ in Figure 9.

Notice that the approximation accuracy of ApproxNDCG decreases as α increases, which is opposite to NeuralNDCG.

H Training Efficiency

The computational complexity of each method depends on evaluating $\pi_\theta(y_j|x)$ and $\pi_{\text{ref}}(y_j|x)$ for each $y_j \in \{\mathbf{Y}\}$ to get corresponding scores in Eq 2, which is $\mathcal{O}(K)$, where K is the list size of multiple responses. Subsequently, the pairwise comparison of multiple responses can be efficiently computed using PyTorch's broadcasting mechanism to perform matrix subtraction.The resulting matrix $P[i, j]$ represents the value of $s_i - s_j$. Therefore, for pairwise methods, it suffices to consider

Run Name	β	Pair-Preference	Scoring	β	Pair-Preference	Scoring
Single Pair	0.05	57.24	54.04	0.01	55.59	51.73
	0.1	**60.75**	56.86	0.5	58.97	**58.16**
BPR	0.05	59.86	56.86	0.01	56.13	55.16
	0.1	**60.32**	**58.33**	0.5	54.24	55.31
All Pairs	0.05	62.12	58.36	0.01	61.18	56.35
	0.1	**63.82**	**60.54**	0.5	56.12	55.77
SLiC	0.05	**63.31**	**60.70**	0.01	59.30	55.61
	0.1	62.68	60.34	0.5	55.23	55.44
LambdaRank	0.05	60.77	56.07	0.01	54.52	51.35
	0.1	**62.30**	**59.04**	0.5	57.72	56.71
ListMLE	0.05	61.81	57.60	0.01	57.49	55.16
	0.1	**63.03**	**59.76**	0.5	56.05	55.77
ApproxNDCG	0.05	58.66	54.34	0.01	55.56	50.76
	0.1	**61.46**	**58.59**	0.2	60.04	57.27
	0.5	58.71	57.39	1.0	56.61	56.00
PPA	0.05	63.92	60.09	0.01	59.58	55.46
	0.1	**64.25**	**61.36**	0.5	58.41	57.65

Table 9: Supplementary Results across different β on Qwen2-0.5B.

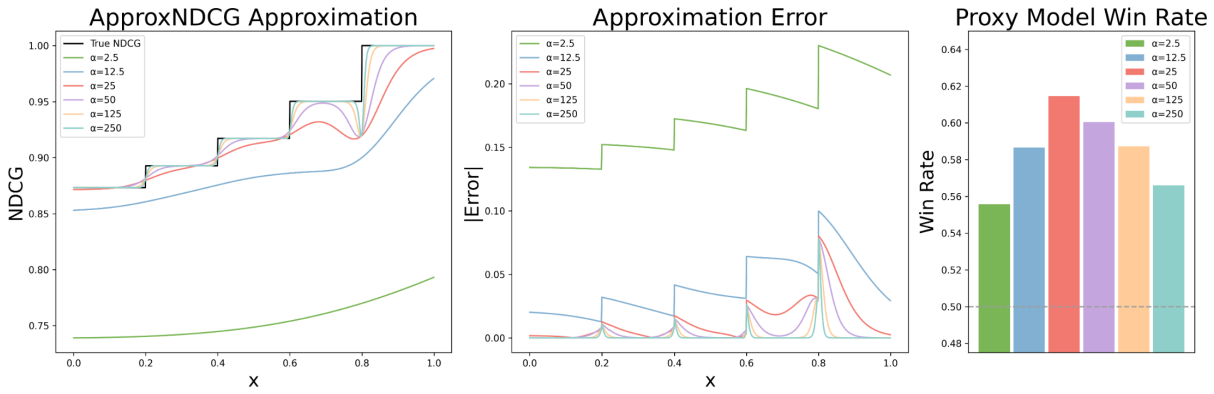


Figure 9: Given ground truth label $\psi = [1.0, 0.8, 0.6, 0.4, 0.2]$, the scores $\mathbf{s} = [x, 0.8, 0.6, 0.4, 0.2]$ and fix $\beta = 0.1$, we visualize the ApproxNDCG Approximation Accuracy with different α and its corresponding absolute value of error and Pair-Preference proxy model win rate against SFT model.

only the upper triangular matrix, excluding diagonal elements. This approach does not significantly increase training time when performing pairwise comparisons. The training times and GPU memory usage of Mistral-7B and Qwen2-0.5B models are shown in Table 14 and Table 15.

NeuralNDCG (as used in PPA) requires NeuralSort and Sinkhorn scaling for each list. In our experiments Table 16, we use lists of size $n = 8$, thus operating on an 8×8 matrix for each sample. The computational complexity per sample is $O(n^2 \cdot \text{iter})$, where iter is the number of Sinkhorn scaling iterations (set to 50 in our experiments). In contrast, DPO operates on all pairs within a list, with complexity $O(n^2)$ per sample.

It is important to note that specific training times and GPU memory usage can exhibit random fluctuations. This result is intended to demonstrate that the training times for pairwise and listwise methods on multiple responses with the same list size do not show significant differences.

I Response Samples

The average response length on the AlpacaEval dataset is shown in Table 17. We also provide response samples in Table 18 from each baseline and our method PPA demonstrates more details in generated responses.

Run Name	List Size	Pair-Preference	Scoring	Pair-Preference	Scoring
		$\beta = 0.1$		$\beta = 0.05$	
All Pairs	2	60.75	56.86	57.24	54.04
	4	63.26	**60.90**	61.59	**58.54**
	6	63.03	59.50	**62.83**	57.93
	8	**63.82**	60.54	62.12	58.36
SLiC	2	63.44	59.07	61.00	57.39
	4	**63.79**	**61.40**	**64.04**	60.54
	6	63.64	61.15	62.01	58.61
	8	62.68	60.34	63.31	**60.70**
LambdaRank	2	60.85	57.62	59.76	56.02
	4	61.10	58.05	59.88	55.51
	6	62.09	57.72	**62.02**	**56.81**
	8	**62.30**	**59.04**	60.77	56.07
ListMLE	2	60.14	57.01	57.14	53.53
	4	**63.57**	**61.23**	**61.94**	**58.49**
	6	62.78	60.92	61.18	57.83
	8	63.03	59.76	61.81	57.60
ApproxNDCG	2	59.73	57.72	**61.56**	**58.26**
	4	59.65	56.45	60.11	55.79
	6	60.70	57.32	59.53	56.35
	8	**61.46**	**58.59**	58.66	54.34
PPA	2	61.94	58.00	58.69	55.89
	4	62.91	59.96	62.65	58.56
	6	64.02	60.11	61.08	59.43
	8	**64.25**	**61.36**	**63.92**	**60.09**

Table 10: Supplementary Results across different list sizes on Qwen2-0.5B. In practice, we keep the response with the highest label and the one with the lowest label, then conduct random sampling from the remaining responses.

Method	Type	Proxy Model		General Benchmark		Avg.
		Pair-Preference	Scoring	AlpacaEval	MT-Bench	
Single Pair	Pairwise	71.90	70.66	74.75	52.19	67.38
BPR	Pairwise	84.43	82.37	86.69	63.44	79.23
Others vs Worst	Pairwise	82.95	80.84	84.64	62.78	77.80
All Pairs	Pairwise	**85.34**	83.31	82.79	61.56	78.25
SLiC	Pairwise	84.12	83.46	83.27	66.25	79.28
LambdaRank	Listwise	85.11	82.52	86.13	**69.06**	80.71
ListMLE	Listwise	83.79	**83.61**	83.46	66.56	79.35
ApproxNDCG	Listwise	82.04	74.64	85.80	67.50	77.50
PPA	Listwise	84.98	83.05	**87.54**	67.81	**80.85**

Table 11: PPA outperforms other baselines on win rates of aligned Mistral-7B against Zephyr-7B-SFT. We set $\beta = 0.01$ for Single Pair and $\beta = 0.05$ for other approaches to achieve the best performance. The other settings are the same as in Table 1.

Method	β	Pair-Preference	Scoring	AlpacaEval
Single Pair	0.1	61.26	70.21	64.45
BPR		79.73	77.59	78.39
All Pairs		79.22	78.43	77.65
SLiC		76.17	75.36	73.04
LambdaRank		80.82	78.53	81.01
ListMLE		78.58	79.22	75.12
ApproxNDCG		76.12	69.21	**82.50**
PPA		**83.13**	**81.66**	81.07
Single Pair	0.05	66.44	65.50	68.87
BPR		84.43	82.37	86.69
Others vs Worst		82.95	80.84	84.64
All Pairs		85.34	83.31	82.79
RankNet		85.61	84.22	86.94
SLiC		84.12	83.46	83.27
LambdaRank		85.11	82.52	86.13
ListMLE		83.79	83.61	83.46
ApproxNDCG		82.04	74.64	85.80
PPA		84.98	83.05	**87.54**
Single Pair	0.01	71.90	70.66	74.75
BPR		77.01	78.46	86.71
All Pairs		72.66	74.09	82.44
PPA		73.17	75.00	84.51

Table 12: Model Scale Up results in Mistral-7B.

Method	Pair-Preference	Scoring	AplacaEval	Arena-Hard
All Pairs	72.96	74.39	59.64	**61.64**
SLiC	72.84	75.04	60.20	59.17
ListMLE	72.46	74.77	59.83	55.18
LiPO	71.80	72.74	60.38	59.69
PPA	**74.34**	**75.58**	**61.32**	60.17

Table 13: Model Scale Up: PPA outperforms other approaches on Llama3.1-8B.

Run Name	List Size	Training Time	GPU Memory Usage
Single Pair	2	3h 28m	92.44%
BPR	8	12h 42m	93.43%
All Pairs	8	12h 38m	93.63%
SLiC	8	11h 42m	93.79%
LambdaRank	8	12h 14m	93.66%
ListMLE	8	12h 26m	93.29%
ApproxNDCG	8	12h 56m	93.64%
PPA	8	11h 39m	93.73%

Table 14: Training Time and GPU memory usage on 16×Nvidia V100-32G with Mistral-7B.

List Size	Approach	Training Time	List Size	Approach	Training Time
2	PPA	3h24m	4	PPA	4h27m
	DPO	3h17m		DPO	4h10m
	ListMLE	3h38m		ListMLE	4h23m
	LiPO	3h32m		LiPO	4h47m
	SLiC	3h28m		SLiC	4h14m
6	PPA	6h31m	8	PPA	7h23m
	DPO	6h12m		DPO	7h14m
	ListMLE	6h56m		ListMLE	7h34m
	LiPO	6h17m		LiPO	7h57m
	SLiC	6h24m		SLiC	8h04m

Table 15: Training Time and GPU memory usage on Qwen2-0.5B.

Method	Complexity	Time	Time per sample
DPO	$O(n^2)$	7.394s	0.739ms
PPA	$O(50 \cdot n^2)$	368.5s	36.850ms

Table 16: Wall-clock time of sorting operations for 10,000 samples on DPO and PPA.

Table 17: Average response length on AlpacaEval dataset.

Method	SFT	Single Pair	BPR	All Pairs	SLiC	LambdaRank	ListMLE	ApproxNDCG	NeuralNDCG
Length	828	877	1096	1054	1023	1140	1041	1284	1120

Table 18: Response samples from AlpacaEval where PPA outperforms other methods in quality judged by GPT-4 Turbo.

Prompt	Q: Who is Larry Page?
PPA	Larry Page is a co-founder, former CEO, and current CEO of Alphabet Inc., the parent company of Google. He was born on March 26, 1973, in East Lansing, Michigan, and graduated from Stanford University with a degree in computer science and artificial intelligence. Page, along with Sergey Brin, developed the PageRank algorithm, which forms the basis of Google's search engine. In 1998, they founded Google, which quickly became one of the most successful and influential technology companies in the world. Page served as Google's CEO from 1999 to 2001 and again from 2011 to 2015, during which time he oversaw the company's expansion into new markets, including mobile devices, cloud computing, and artificial intelligence. In 2015, Page stepped down as CEO of Google and became the CEO of Alphabet, the holding company that was created to separate Google's core internet business from its other ventures. As Alphabet's CEO, Page has focused on developing new technologies and expanding into emerging markets, such as healthcare and autonomous vehicles. Page is widely regarded as one of the most innovative and influential entrepreneurs of our time, and his contributions to the field of computer science and technology have had a profound impact on the world.

SFT	Larry Page is the co-founder and former CEO of Google, a multinational technology company. He is also the co-founder of Alphabet Inc., the parent company of Google. Page is known for his contributions to the development of the PageRank algorithm, which is used to determine the importance of websites in search engine rankings. He is also involved in various philanthropic efforts, including the creation of the Google Foundation, which supports projects related to education, health, and the environment.
Single Pair	Larry Page is an American entrepreneur, computer scientist, and co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, and graduated from Stanford University with a degree in computer science and artificial intelligence. Page is known for his innovative ideas and leadership in the technology industry, particularly in the areas of search engines, artificial intelligence, and renewable energy. He currently serves as the CEO of Alphabet Inc., the parent company of Google, and is considered one of the most influential figures in the tech world.
BPR	Larry Page is a co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, USA. Page and Brin started Google in 1998 while they were both PhD students at Stanford University. Page's primary role in the company has been as CEO from 2001 to 2019, and he currently serves as the CEO of Alphabet Inc., the parent company of Google. Under his leadership, Google has become one of the most successful and influential technology companies in the world, with a market capitalization of over $1 trillion. Page is also known for his philanthropic efforts, including funding research on renewable energy and supporting education initiatives.
All Pairs	Larry Page is a co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, USA. Page and Brin started Google in 1998 while they were both PhD students at Stanford University. Page is known for his focus on innovation, simplicity, and user experience, and he has been instrumental in shaping Google's vision and strategy. He served as CEO of Alphabet Inc., the parent company of Google, from 2015 to 2019, and currently serves as the CEO of Alphabet's subsidiary, Google. Page is also a philanthropist and has been involved in various charitable initiatives, including the PageRipples Foundation, which supports education and environmental causes.

SLiC	Larry Page is a co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, USA. Page and Brin started Google in 1998 while they were both PhD students at Stanford University. Page is currently the CEO of Alphabet Inc., the parent company of Google, and is responsible for overseeing the company's overall strategy and direction. He is also a prominent figure in the technology industry, known for his focus on innovation, sustainability, and philanthropy.
LambdaRank	Larry Page is a co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, USA. Page and Brin started Google in 1998 while they were both PhD students at Stanford University. Page served as the CEO of Google from 1999 to 2001, and then again from 2011 to 2015. He currently serves as the CEO of Alphabet Inc., the parent company of Google. Page is known for his focus on innovation, sustainability, and philanthropy, and has been recognized as one of the most influential people in the world.
ListMLE	Larry Page is a co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, USA. Page and Brin started Google in 1998 while they were both PhD students at Stanford University. Page is currently the CEO of Alphabet Inc., the parent company of Google, and is responsible for overseeing the company's overall strategy and direction. He is also a prominent figure in the technology industry, known for his focus on innovation, sustainability, and philanthropy.
ApproxNDCG	Larry Page is a co-founder of Google, along with Sergey Brin. He was born on March 26, 1973, in East Lansing, Michigan, and grew up in the Detroit area. Page and Brin met at Stanford University, where they developed the PageRank algorithm, which became the basis for Google's search engine.\n\n Page served as the CEO of Google from 1999 to 2001, and then again from 2011 to 2019. During his tenure, he oversaw the company's growth and expansion, including the development of new products and services, such as Google Maps, Google Earth, and Google Glass.\n\n Page is known for his focus on innovation, user experience, and sustainability. He has also been involved in various philanthropic efforts, including the creation of the Google.org foundation and the development of the Google Self-Driving Car Project.\n\n Today, Page is the CEO of Alphabet, the parent company of Google, and continues to play a key role in shaping the future of technology and the internet.