K-order Ranking Preference Optimization for Large Language Models

Anonymous ACL submission

Abstract

To adapt Large Language Models (LLMs) to ranking tasks, existing list-wise methods, represented by list-wise Direct Preference Optimization (DPO), focus on optimizing partial-order or full-order list ranking consistency for LLMs to enhance their ranking abilities. However, we argue that optimizing top-K ranking consistency could be more appropriate for real-world applications. There are two main reasons: (1) users are typically concerned with only the top-K results, making top-K ranking more important, and (2) tail items often lack precise feedback, making top-K ranking more reliable. Based on this, we propose ***K***-order Ranking **P**reference **O**ptimization (KPO) by extending the DPO's Plackett-Luce model to accommodate top-K rankings. Additionally, recognizing that the number of important items can vary across queries, we extend KPO to dynamically determine appropriate K for different samples and introduce a curriculum learning strategy to boost training efficiency. Extensive experiments demonstrate the effectiveness of KPO, highlighting its high sample efficiency and robustness to noise. The code is available at https://anonymous.4open.science/r/KPO-BEAF.

1 Introduction

Large Language Models (LLMs) have shown great potential in addressing a wide range of real-world tasks (Hadi et al., 2023; Minaee et al., 2024). By leveraging their semantic reasoning abilities and extensive world knowledge, LLMs can more effectively capture the nuanced relationships between queries and candidate items, making them also promising for ranking tasks (Sun et al., 2023; Pradeep et al., 2023b) — the core of many real-world applications such as product search (Spatharioti et al., 2023; Fang et al., 2024) and recommendation (Chen et al., 2024b; Yue et al., 2023). However, as illustrated in Fig.(1), ranking tasks extend beyond evaluating the relevance of individual candidates to a user query; they require ranking a list of candidates. Yet, LLMs are not explicitly trained to optimize list-wise ranking preferences during pretraining. This limitation has sparked greater research efforts to enhance LLMs' list-wise ranking capabilities (Pradeep et al., 2023a).

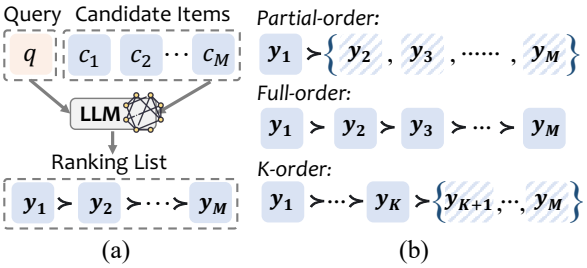


Figure 1: (a) Illustration of the LLM-based ranking task. (b) Comparison of three ranking strategies.

Among existing approaches, the list-wise Direct Preference Optimization (DPO) method (Rafailov et al., 2023; Chen et al., 2024b) has emerged as a promising technique for optimizing LLMs to generate ranked outputs that align with human preferences **directly at the list level**. According to the ranking consistency optimized for a list, existing methods can be categorized into:

- *Partial-order Method* (e.g., S-DPO (Chen et al., 2024b)), which simply optimizes the ranking consistency where "the best item is better than all others," i.e., $y_1 \succ$ all others. This method focuses on the ranking of the best one, failing to optimize the fine-grained ranking consistency.
- *Full-order Method* (e.g., DPO_{PL} (Rafailov et al., 2023)), which optimizes complete and fine-grained ranking consistency, i.e., $y_1 \succ y_2 \succ \cdots \succ \ldots$ Ideally, this method ensures optimal ranking alignment, but the optimization's inherent difficulties could limit its practical performance.

Given these, we argue that optimizing top-K ranking consistency would be more appropriate

for real-world ranking tasks (Adomavicius and Zhang, 2016; Le and Lauw, 2021). In practical scenarios, users typically have limited attention and focus only on the most relevant items, making top-K optimization sufficient to meet their needs. Moreover, this limited attention makes it difficult to obtain accurate preference ranks for less relevant items, rendering ranking optimization for long-tail items inherently unreliable. Therefore, we propose top-K order ranking preference alignment for LLMs—optimizing the model to align fine-grained ranking consistency for the top-K items while disregarding it for others, (i.e., optimizing $y_1 \succ \ldots \succ y_K \succ$all others), as shown in Fig.(1).

Towards the top-K order ranking preference alignment, we propose ***K**-order Ranking **P**reference **O**ptimization* (KPO). The core idea is to extend existing DPO methods' Plackett-Luce preference model (Plackett, 1975), originally designed for full rankings, to accommodate top-K rankings. Intuitively, KPO works by increasing the relative log probability of **each** top-K item over all its subsequent items, ensuring both the fine-grained order among the top-K items and the order between the top-K items and the others. As discussed, KPO is expected to outperform full-order methods due to its closer alignment with real-world scenarios. Additionally, theoretical analysis demonstrates that KPO surpasses existing partial-order methods.

Taking it a step further, in real-world scenarios, the number of most relevant items can vary across queries. To address this, we extend KPO to handle varying K values across samples, incorporating a strategy to adaptively determine K based on LLM confidence in assessing item relevance. Furthermore, to accommodate varying K, we incorporate a curriculum learning strategy into KPO to simplify the learning process. Specifically, we guide KPO to focus on K-order optimization progressively, starting from smaller K and gradually increasing to larger K. This approach is motivated by the fact that higher K introduces greater learning challenges, as it requires distinguishing more complete and fine-grained rankings.

The main contributions of this work can be summarized as follows:

- We propose optimizing top-K ranking consistency for LLM ranking preference alignment to better match real-world needs and constraints.
- We propose KPO for top-K order ranking alignment, incorporating an adaptive strategy to determine suitable K values for different samples and a curriculum learning strategy to enhance training effectiveness.
- Extensive experimental results validate KPO's effectiveness while showcasing its high sample efficiency and robustness to noisy logits.

2 Related Work

In this section, we delve into related studies from two perspectives: LLM-based ranking and preference alignment in LLMs.

2.1 LLM-based Ranking

With the rise of LLMs with strong reasoning abilities, researchers have increasingly explored their potential in ranking tasks (Sun et al., 2023; Qin et al., 2024; Ma et al., 2024; Yue et al., 2023). Studies in this area generally follow two approaches: zero-shot usage or fine-tuning for enhanced performance. In the zero-shot setting, methods like RankGPT (Sun et al., 2023) leverage ChatGPT (OpenAI, 2022, 2023) to rank candidate passages based on a query. Fine-tuned models, such as RankLLaMA (Ma et al., 2024), use point-wise training to estimate relevance scores, improving reranking precision. LlamaRec (Yue et al., 2023) further extends this by introducing a two-stage framework with a verbalizer-based method for generating probability distributions over candidate items. These advancements highlight the growing role of LLMs in ranking tasks, particularly for search and recommendation applications.

2.2 Preference Alignment in LLMs

Preference alignment helps LLMs differentiate between "good" and "bad" answers using human-labeled data (Ouyang et al., 2022). For example, DPO (Rafailov et al., 2023) fine-tunes LLMs with pair-wise preference data, while KTO (Ethayarajh et al., 2024), inspired by Kahneman-Tversky's prospect theory (Tversky and Kahneman, 1992), simplifies this process by utilizing point-wise labels. However, both approaches face limitations in effectively handling ranking tasks that require aligning LLMs with multi-item ranking information. Extensions such as DPO_{PL} (Rafailov et al., 2023) and S-DPO (Chen et al., 2024b) adapt DPO for list-wise settings: DPO_{PL} targets full-order rankings, while S-DPO handles partial-order rankings. Nonetheless, these methods overlook K-order ranking, a critical aspect of ranking tasks.

3 Problem Definition

Consider a ranking dataset $\mathcal{D}$ comprising query-candidate pairs, where each k-th instance $(q^{(k)}, \mathcal{C}^{(k)}) \in \mathcal{D}$ consists of: (1) A query $q^{(k)}$ representing an information need (e.g., search query, recommendation context). (2) A candidate set $\mathcal{C}^{(k)} = \{c_1^{(k)}, c_2^{(k)}, \ldots, c_M^{(k)}\}$ containing M items to be ranked.

The LLM takes as input a concatenated sequence $x^{(k)} = (q^{(k)}, \mathcal{C}^{(k)})$ and aims to generate a permutation $\mathcal{Y}^{(k)} = \{y_1^{(k)} \succ y_2^{(k)} \succ \cdots \succ y_M^{(k)}\}$, where $\forall y_i^{(k)} \in \mathcal{C}^{(k)}$, the symbol $\succ$ represents a pair-wise preference relationship. When ambiguity is absent, we omit the superscript $^{(k)}$ for notational simplicity (e.g., y_i instead of $y_i^{(k)}$).

We instantiate this framework through two representative ranking applications:

- *Sequential Recommendation:* The query $q \triangleq [v_1, v_2, \ldots, v_m]$ encodes a user's interaction history, where v_j denotes the j-th consumed item.
- *Product Search:* The query q represents a textual search intent (e.g., "wireless noise-canceling headphones").

Both tasks share the core challenge of learning context-aware preference relations, but differ fundamentally in their query semantics - making them ideal testbeds for evaluating the generalization of ranking frameworks.

4 Methodology

We first review foundational work in preference modeling to establish the necessary background. Then, we introduce the proposed model in detail.

4.1 Preliminary

Preference Modeling. Preference modeling aims to learn a function that captures human preferences over a set of candidate items, enabling applications such as recommender systems, information retrieval, and human-AI alignment. One common approach is the Bradley-Terry (BT) model (Bradley and Terry, 1952), which provides a probabilistic framework for pair-wise preference learning, defining the likelihood of selecting y_1 over y_2 given context x as:

$$\hat{p}(y_1 \succ y_2 \mid x) = \frac{\exp(r(x, y_1))}{\exp(r(x, y_1)) + \exp(r(x, y_2))}, \tag{1}$$

where $r(x, y)$ is a task-specific reward function that quantifies the relative preference for candidate y in context x. To learn a policy model that aligns with preferences, a widely adopted approach is Direct Preference Optimization (DPO) (Rafailov et al., 2023). DPO formulates the reward function in terms of the policy model π_θ and a reference model π_{ref}:

$$r(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x), \tag{2}$$

where β controls the divergence between π_θ and π_{ref}. The partition function $Z(x)$ is defined as:

$$Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right). \tag{3}$$

Full-order Preference Modeling. While the pair-wise BT model in Eq. (1) is effective for binary comparisons, it struggles with ranking tasks involving multiple candidate items. To address this limitation, prior work (e.g., DPO_{PL} (Rafailov et al., 2023)) has generalized BT to the *list-wise* Plackett-Luce (PL) model (Plackett, 1975), which represents rankings as a full-order sequence $y_1 \succ y_2 \succ \cdots \succ y_M$:

$$\hat{p}(y_1 \succ y_2 \succ \cdots \succ y_M \mid x) = \prod_{i=1}^{M-1} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))}. \tag{4}$$

However, full-order methods risk overemphasizing irrelevant item relationships, making optimization more challenging.

Partial Preference Modeling. To mitigate this, S-DPO (Chen et al., 2024b) simplifies the PL model by structuring preferences as a single positive candidate against multiple negatives. This modification models preference as $y_1 \succ \{y_2, \ldots, y_M\}$:

$$\hat{p}(y_1 \succ \{y_2, \ldots, y_M\} \mid x) = \frac{\exp(r(x, y_1))}{\sum_{j=1}^{M} \exp(r(x, y_j))}. \tag{5}$$

While S-DPO reduces computational complexity, it oversimplifies the ranking problem by ignoring nuanced distinctions among top candidates. In real-world applications such as top-K recommendation (Kweon et al., 2024; Luo et al., 2024) and top-K retrieval (Ciaccia and Martinenghi, 2024; Lee et al., 2023), users are primarily interested in the relative ordering of the most relevant items. This motivates our proposal for a hybrid approach that combines the strengths of full-order and partial-order models, focusing specifically on accurate top-K preference modeling.

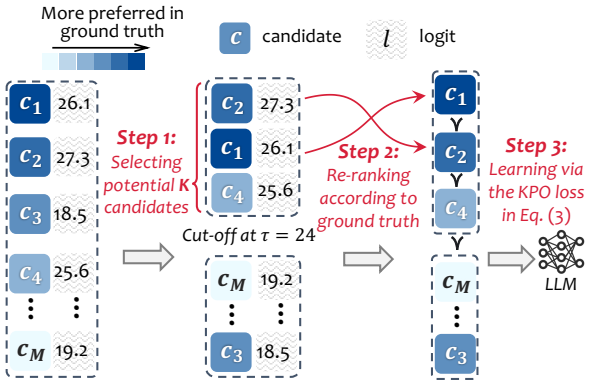


Figure 2: The KPO framework consists of: (1) selecting K candidates via LLM logits above threshold τ, (2) re-ranking top-K candidates to match ground truth, and (3) training the model with the KPO loss in Eq. (7).

4.2 KPO

Our goal is to derive a K-order preference: $y_1 \succ \cdots \succ y_K \succ \{y_{K+1}, \ldots, y_M\}$ where $y_i \in \mathcal{C}$, $\{y_1, y_2, \ldots, y_K\}$ correspond to the top-K relevant items, and $\{y_{K+1}, \ldots, y_M\}$ represent the remaining irrelevant items.

Based on the PL model in Eq. (4), we can define the K-order preference model as below:

$$\hat{p}(y_1 \succ \cdots \succ y_K \succ \{y_{K+1}, \ldots, y_M\} \mid x) = \prod_{i=1}^{K} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))}. \tag{6}$$

Due to space constraints, the detailed derivation of Eq. (6) is provided in Appendix A.1.

Remark: The proposed K-order preference framework generalizes existing approaches, with the DPO, DPO$_{\text{PL}}$, and S-DPO emerging as special cases of Eq. (6). When $M = 2$ and $K = 1$, it reduces to DPO's pair-wise preference modeling. When $K = M$, it recovers DPO$_{\text{PL}}$'s full-order ranking. When $K = 1$, it simplifies to S-DPO's partial-order formulation.

By following the implementation of the reward function $r(x, y)$ from Eq. (2) in DPO, we can derive the loss function $\mathcal{L}_{\text{KPO}}$ to maximize $\hat{p}$ on a ranking dataset $\mathcal{D}$ as follows:

$$\mathcal{L}_{\text{KPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}} \left[\sum_{i=1}^{K} \log \sigma \left(- \log \sum_{j=i+1}^{M} \exp \left(\beta \log \frac{\pi_\theta(y_j|x)}{\pi_{\text{ref}}(y_j|x)} - \beta \log \frac{\pi_\theta(y_i|x)}{\pi_{\text{ref}}(y_i|x)} \right) \right) \right]. \tag{7}$$

Theoretical Analysis: We analyze the optimal top-K ranking accuracy of KPO through the following theorem.

Theorem 1. *Let π^* be the optimal policy that maximizes the KPO objective. Given a dataset of aggregated preferences $\mathcal{D}_p = \{(x, y_1 \succ \cdots \succ y_K \succ \{y_{K+1}, \ldots, y_M\}\}$. Assume $\mathcal{D}_p$ contains ground-truth ranking probabilitie following the PL model. Specifically, for any item y_i and the subset of remaining items $\{y_{i+1}, \ldots, y_M\}$, the ranking probability is defined as follows:*

$$\alpha(x, y_i, y_{>i}) = \mathbb{P}(y_i \succ \{y_{i+1}, \cdots, y_M\}). \tag{8}$$

The top-K ranking accuracy of π^* is given by:

$$\mathcal{R}^*_{KPO}(\mathcal{D}_p, \pi_{\text{ref}}) = \mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}_p} \left[\prod_{l=1}^{K} \prod_{k=l+1}^{M} \mathbb{I} \left[\frac{w_l \pi_{\text{ref}}(y_l \mid x)}{w_k \pi_{\text{ref}}(y_k \mid x)} > 1 \right] \right], \tag{9}$$

where $\frac{w_l}{w_k}$ is defined as

$$\frac{w_l}{w_k} = \left(\frac{\alpha(x, y_l, y_{>l})}{\alpha(x, y_k, y_{>k})} \right)^{1/\beta} \cdot \prod_{i=l}^{k-1} (1 - \alpha(x, y_i, y_{>i}))^{-1/\beta}. \tag{10}$$

The proof is deferred to Appendix A.2.

According to Theorem 1, we can derive the optimal accuracy of S-DPO as:

$$\mathcal{R}^*_{\text{S-DPO}}(\mathcal{D}_p, \pi_{\text{ref}}) = \mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}_p} \left[\prod_{l=1}^{K} \prod_{k=l+1}^{M} \mathbb{I} \left[\frac{w'_l \pi_{\text{ref}}(y_l \mid x)}{w'_k \pi_{\text{ref}}(y_k \mid x)} > 1 \right] \right], \tag{11}$$

where $\frac{w'_l}{w'_k}$ is defined as

$$\frac{w'_l}{w'_k} = \left(\frac{\alpha(x, y_l, y_{>l})}{\alpha(x, y_k, y_{>k})} \right)^{1/\beta} \cdot \prod_{i=l}^{k-1} (1 - \alpha(x, y_i, y_{>i}))^{-1/\beta} \cdot \mathbb{I}[l = 1] + \mathbb{I}[l \neq 1]. \tag{12}$$

Based on Eq. (10) and Eq. (12), we can conclude that: $\frac{w_l}{w_k} > \frac{w'_l}{w'_k}$ for all $l \in \{2, \ldots, K\}$ and $k \in \{l+1, \ldots, M\}$. Therefore, we have $\mathcal{R}_{\text{KPO}}(\mathcal{D}_p, \pi_{\text{ref}}) > \mathcal{R}^*_{\text{S-DPO}}(\mathcal{D}_p, \pi_{\text{ref}})$, implying that the optimal ranking accuracy of KPO is greater than S-DPO. The detailed derivation is provided in Appendix A.3.

4.3 Query-adaptive KPO

In real-world ranking scenarios, the number of relevant candidates K often varies significantly across queries. For instance, a query like "NVIDIA A40 GPU" typically has a single authoritative result, while "budget wireless headphones" may involve multiple comparable options. To address this, we propose a query-adaptive extension of KPO that dynamically adjusts to each query's characteristics.

4.3.1 Query-adaptive KPO Loss

The key challenge lies in determining the appropriate K for each input $x = (q, \mathcal{C})$. We formalize this through a query-adaptive function $\mathcal{K}(x)$ that predicts the number of relevant candidates for a given query. This allows us to extend the KPO loss to its query-adaptive form:

$$\mathcal{L}_{\mathrm{KPO}}^{\mathcal{K}(x)}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathbb{E}_{(x,y_1,\ldots,y_M)\sim\mathcal{D}}\left[\sum_{i=1}^{\mathcal{K}(x)} \log\sigma\left(-\log\sum_{j=i+1}^{M}\exp\left(\beta\log\frac{\pi_\theta(y_j|x)}{\pi_{\mathrm{ref}}(y_j|x)} - \beta\log\frac{\pi_\theta(y_i|x)}{\pi_{\mathrm{ref}}(y_i|x)}\right)\right)\right] \tag{13}$$

4.3.2 K-aware Curriculum Learning

To effectively train the query-adaptive loss in Eq. (13), we propose a K-aware curriculum strategy (Bengio et al., 2009). This approach organizes training instances based on their complexity, where complexity is defined by the number of relevant candidates K. We treat queries with smaller K values as "simple samples", as they require the model to focus on only a few relevant items. Conversely, queries with larger K values are considered "challenging samples", demanding more complex ranking decisions.

Following this intuition, we sort the training data in ascending order of K, allowing the model to first learn from simpler queries before progressively handling more complex ones. This structured training not only facilitates smoother convergence but also ensures consistent K values within each batch, improving training stability.

4.3.3 Acquisition of Query-adaptive K

To determine query-adaptive K values for each input $x = (q, \mathcal{C})$, we leverage the output information of the LLM itself to select K relevant candidates, eliminating the need for additional information. Specifically, we first use the reference model π_{ref} to compute the logits $logits(q, c_i)$ for each candidate item $c_i \in \mathcal{C}$ based on the given query q. Items with logits exceeding a predefined hyperparameter threshold τ are regarded as relevant candidates. The number of such items is then counted to determine the query-adaptive K. Formally, this process is represented as $\mathcal{K}(x)$, defined as:

$$\mathcal{K}(x) = \mathcal{K}(q, \mathcal{C}) = \sum_{i=1}^{M} \mathbb{I}\left(logits(q, c_i) > \tau\right). \tag{14}$$

After obtaining the K values, we generate K-order ranking data for KPO training by first sorting candidate items based on their logits to select the top-K items. These top-K items are then re-ranked using ground truth relevance labels to ensure the correct relative order. The resulting training data is structured as $y_1 \succ \cdots \succ y_{\mathcal{K}(x)} \succ \{y_{\mathcal{K}(x)+1}, \ldots, y_M\}$. Details on obtaining the ground truth labels are provided in Appendix B.

The whole pipeline of the proposed method is illustrated in Fig.(2).

4.3.4 Analysis of Time Complexity

The optimization objective of KPO introduces an additional K-layer loop compared to S-DPO, which may raise concerns about time complexity.

To address this potential issue, we conduct an analysis of the time required for the actual optimization process. Specifically, the parameter update process can be divided into three phases:

- **Phase 1**: Compute M "rewards" ($r_i = \beta\log\frac{\pi_\theta(y_i|x)}{\pi_{\mathrm{ref}}(y_i|x)}$).
- **Phase 2**: Use the rewards to compute the loss.
- **Phase 3**: Update model parameters via loss backpropagation.

The K-layer loop introduced by KPO occurs in Phase 2. However, the actual runtime of Phase 2 is significantly shorter compared to Phase 1 and Phase 3, and thus does not impact the overall runtime of the method. Detailed experimental results supporting this conclusion are provided in Appendix D.1.

5 Experiments

In this section, we aim to answer the following research questions (RQ):

- **RQ1**: How does KPO perform in the recommendation and product search tasks?
- **RQ2**: What are the effects of the key components and hyperparameters?
- **RQ3**: How does KPO perform in terms of sample efficiency and robustness to noisy logits?

5.1 Experimental Setup

We organize experiments on two typical ranking tasks: recommendation and product search.

5.1.1 Datasets

For the recommendation task, we utilize the MovieLens (Harper and Konstan, 2016) and Goodreads (Wan and McAuley, 2018) datasets. The user interaction sequences in each dataset are chronologically sorted and then split into training, validation, and test sets in an 8:1:1 ratio.

Method	MovieLens					Goodreads					Shopping Queries	
	HR@1	HR@5	HR@10	N@5	N@10	HR@1	HR@5	HR@10	N@5	N@10	N@5	N@10
KTO	0.5368	0.8421	0.9474	0.6996	0.7342	0.4875	0.8486	0.9534	0.6808	0.7147	0.7327	0.7525
DPO	0.5263	0.8632	0.9579	0.7052	0.7348	0.4908	0.8569	0.9584	0.6858	0.7216	0.7356	0.7531
SimPO	0.5263	**0.8842**	0.9579	0.7217	0.7448	0.4842	0.8569	0.9551	0.6794	0.7113	0.7392	0.7560
cDPO	0.5158	0.8632	**0.9684**	0.6960	0.7290	0.4509	0.8536	0.9534	0.6651	0.6979	0.7321	0.7503
S-DPO	0.5368	0.8526	0.9474	0.7062	0.7369	0.4842	0.8353	0.9484	0.6712	0.7083	0.7288	0.7480
DPO_{PL}	0.5474	0.8737	0.9474	0.7229	0.7463	0.4859	0.8619	**0.9634**	0.6876	0.7205	0.7363	0.7529
KPO_{CUT}	0.5474	0.8632	**0.9684**	0.7167	0.7493	0.4992	0.8453	0.9468	0.6852	0.7182	0.7347	0.7521
KPO	**0.5579**	**0.8842**	**0.9684**	**0.7361**	**0.7620**	**0.5042**	**0.8719**	0.9584	**0.6994**	**0.7272**	**0.7477**	**0.7631**

Table 1: **Comparison with preference alignment methods.** Bold indicates the best performance.

For the product search task, we used the Shopping Queries dataset (Reddy et al., 2022), which includes queries paired with up to 40 candidate products. Each product is assigned a four-level score ($\{0, 1, 2, 3\}$) representing its relevance to the query, which can serve as the ground truth label. Queries are grouped and randomly split into training, validation, and test sets in an 8:1:1 ratio.

The detailed description of the datasets and their statistical information is provided in Appendix C.1.

5.1.2 Evaluation Setting

We evaluate the model's ability to rank 20 candidate items based on a given query.

For the recommendation task, the ground truth item is the user's most recently interacted item. The candidate list includes this ground truth item and 19 randomly sampled items. The model's performance is evaluated based on its ability to rank the ground truth item higher, using Hit Ratio (HR@1, 5, 10) and Normalized Discounted Cumulative Gain (N@5, 10).

For the product search task, multiple ground truth items have relevance labels, we evaluate the model using N@5 and N@10 to measure its ability to prioritize highly relevant items. Additional results for the setting with a single ground truth item are provided in Appendix D.2.

5.1.3 Implementation Details

Our experiments are conducted on eight NVIDIA A40 GPUs. We use the Llama-3.2-3B-Instruct (Meta, 2024) model as the backbone. In the supervised fine-tuning (SFT) stage, the model is trained for 5 epochs with a learning rate of 1e-4. In the preference alignment stage, the learning rate is reduced to 1e-5, and training is performed over 3 epochs. The global batch size is fixed at 128. Refer to Appendix C.3 for more implementation details.

5.2 Overall Performance (RQ1)

In this section, we compare KPO with other preference alignment methods and non-preference alignment methods to evaluate the effectiveness of KPO.

Method	Modeling	Objective
KTO	y	$\lambda_y - v_{\text{KTO}}(x, y)$
DPO	$y_1 \succ y_2$	$-\log \sigma\left(r_1 - r_2\right)$
SimPO	$y_1 \succ y_2$	$-\log\sigma\left(\frac{\beta}{\lvert y_1\rvert}\log\pi_\theta(y_1\lvert x) - \frac{\beta}{\lvert y_2\rvert}\log\pi_\theta(y_2\lvert x) - \gamma\right)$
cDPO	$y_1 \gtrless y_2$	$-(1-\epsilon)\log\sigma\left(r_1 - r_2\right) - \epsilon\log\sigma\left(r_2 - r_1\right)$
S-DPO	$y_1 \succ \{y_2, \ldots, y_M\}$	$\log\sigma\left(-\log\sum_{j=2}^{M}\exp\left(r_j - r_1\right)\right)$
DPO_{PL}	$y_1 \succ y_2 \succ \ldots \succ y_M$	$\sum_{i=1}^{M-1}\log\sigma\left(-\log\sum_{j=i+1}^{M}\exp\left(r_j - r_i\right)\right)$
KPO_{CUT}	$y_1 \succ y_2 \succ \ldots \succ y_{\mathcal{K}(x)}$	$\sum_{i=1}^{\mathcal{K}(x)-1}\log\sigma\left(-\log\sum_{j=i+1}^{\mathcal{K}(x)}\exp\left(r_j - r_i\right)\right)$
KPO	$y_1 \succ y_2 \succ \ldots \succ y_{\mathcal{K}(x)} \succ \{y_{\mathcal{K}(x)+1}, \cdots y_M\}$	$\sum_{i=1}^{\mathcal{K}(x)}\log\sigma\left(-\log\sum_{j=i+1}^{M}\exp\left(r_j - r_i\right)\right)$

Table 2: **Modeling approaches and optimization objectives for preference alignment methods.** For convenience, we define $r_i = \beta\log\frac{\pi_\theta(y_i|x)}{\pi_{\text{ref}}(y_i|x)}$. The detailed definitions of KTO are provided in the Appendix A.4.

5.2.1 Comparison with Preference Alignment Methods

To evaluate the effectiveness of KPO loss, we compare it with various preference alignment methods on the recommendation and product search tasks.

Baselines. We compare KPO to various baselines, including KTO (Ethayarajh et al., 2024), DPO (Rafailov et al., 2023), SimPO (Meng et al., 2024), Conservative DPO (cDPO) (Mitchell, 2023), S-DPO (Chen et al., 2024b), and DPO_{PL} (Rafailov et al., 2023). We also introduce KPO_{CUT}, a KPO variant that cuts off tail-irrelevant items $\{y_{\mathcal{K}(x)+1}, \ldots, y_M\}$ for comparison. Objective formulations are summarized in Table 2, with detailed baseline descriptions in Appendix C.2.1.

Results. The experimental results are summarized in Table 1. To fairly evaluate the loss function's effectiveness, KPO's performance is reported

Method	MovieLens HR@1	HR@5	HR@10	N@5	N@10	Goodreads HR@1	HR@5	HR@10	N@5	N@10
SASRec	0.4043	0.8298	0.9043	0.6356	0.6588	0.3661	0.7654	0.9118	0.5763	0.6238
GRU4Rec	0.4526	0.8316	0.9053	0.6498	0.6738	0.3478	0.7504	0.9251	0.5606	0.6185
Caser	0.3404	0.7979	0.9255	0.5845	0.6259	0.4133	0.8083	0.9283	0.6251	0.6640
MoRec	0.2737	0.6842	0.8211	0.4783	0.5244	0.3111	0.7121	0.8918	0.5240	0.5824
LLaRA	0.4565	0.8370	0.9130	0.6376	0.6630	0.4742	0.8053	0.9235	0.6341	0.6713
SFT	0.5053	0.8526	0.9368	0.6983	0.7255	0.4809	0.8369	0.9468	0.6675	0.7034
KPO_{CL}	**0.5684**	**0.8947**	**0.9684**	**0.7381**	**0.7637**	**0.5158**	**0.8735**	**0.9667**	**0.7024**	**0.7353**

Table 3: **Comparison with other traditional and LLM-based models.** Bold indicates the best performance.

without curriculum learning. The key findings are as follows: (1) KPO consistently outperforms other methods across most metrics, demonstrating its effectiveness. (2) KPO surpasses KPO_{CUT}, highlighting the importance of irrelevant items in helping the model distinguish between relevant and irrelevant ones. (3) Although KPO slightly underperforms DPO_{PL} in HR@10 on the Goodreads dataset, the HR@10 values across all methods are already high. Notably, KPO achieves a higher N@10 than DPO_{PL}, reflecting better overall ranking quality.

5.2.2 Comparison with Non-Preference Alignment Methods

To verify whether KPO outperforms other non-preference alignment methods, this section focuses on the recommendation task and compares KPO with various recommendation models.

Baselines. We thoroughly compare KPO with two categories of models: traditional models (SASRec (Kang and McAuley, 2018), GRU4Rec (Hidasi et al., 2016), Caser (Tang and Wang, 2018)) and LLM-based models (MoRec (Yuan et al., 2023), LLaRA (Liao et al., 2024)). The detail description of the models can be found in Appendix C.2.2.

Results. We evaluate the full KPO method with K-aware curriculum learning (KPO_{CL}) against baseline models, including the SFT model for comparison. As shown in Table 3, KPO_{CL} significantly outperforms baseline models, demonstrating its effectiveness. This improvement likely stems from the fact that baseline models are trained based on single ground truth items, neglecting the ranking relationships among multiple items, a core focus of the KPO method.

5.3 Effectiveness of Key Components (RQ2)

We investigate the effects of the following key components of our method: (1) K-aware curriculum learning, (2) query-adaptive K, (3) β in Eq. (13), and (4) threshold τ in Eq. (14).

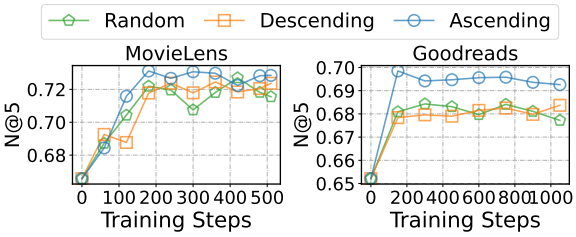


Figure 3: N@5 on MovieLens and Goodreads validation set under different training orders.

5.3.1 K-aware Curriculum Learning

To verify the effectiveness of K-aware curriculum learning, we design three training orders for the datasets: (1) "Random": The dataset is randomly shuffled. (2) "Descending": The dataset is sorted by K in descending order. (3) "Ascending": The dataset is sorted by K in ascending order.

We evaluate the impact of training orders by plotting N@5 curves on the validation set (Fig.(3)). The results show that the "Ascending" order outperforms "Random" and "Descending" orders in both overall performance and stability, underscoring the effectiveness of K-aware curriculum learning. Test set results are provided in Appendix D.3.

K	MovieLens HR@1	HR@5	HR@10	N@5	N@10
1	0.5368	0.8526	0.9474	0.7062	0.7369
3	0.5579	0.8737	0.9684	0.7279	0.7574
5	0.5474	0.8737	0.9684	0.7251	0.7526
7	0.5474	0.8632	0.9579	0.7258	0.7469
10	0.5474	0.8737	0.9474	0.7229	0.7463
query-adaptive	**0.5579**	**0.8842**	**0.9684**	**0.7361**	**0.7620**

Table 4: **Comparison of query-adaptive and fixed K.**

5.3.2 Query-adaptive K

To evaluate the effectiveness of query-adaptive K, we compare the performance of query-adaptive KPO against KPO with fixed K values ($[1, 3, 5, 7, 10]$). As shown in Table 4, query-adaptive K consistently outperforms fixed K, high-

lighting its effectiveness.

5.3.3 β in the Loss Function Eq. (13)

Fixing τ at 24, the β is varied across $[0.1, 0.5, 1.0, 3.0, 5.0]$. Typically, smaller β values indicate stronger influence of preference signals on the LLM, while larger values suggest weaker influence. As shown in Fig. (4), the best performance occurs at $\beta = 1.0$, with higher β values leading to a notable drop in HR@1. This underscores the importance of effectively leveraging preference signals in ranking tasks.

5.3.4 Threshold τ in Eq. (14)

Fixing β at 1.0, the τ is varied across $[18, 20, 22, 24, 26]$. Based on the experimental results in Fig.(4), we can find that $\tau = 24$ is the optimal value. Additionally, the performance of the model is not significantly affected by variations in τ, further indicating that KPO demonstrates a certain level of robustness and stability.

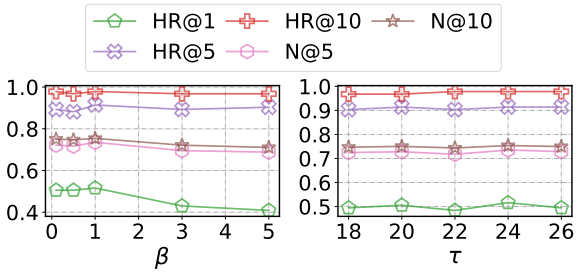


Figure 4: Study of β, τ on MovieLens validation set.

5.4 In-depth Analysis of KPO (RQ3)

In this section, we conduct an in-depth analysis of KPO from two key perspectives: (1) sample efficiency, and (2) robustness to noisy logits.

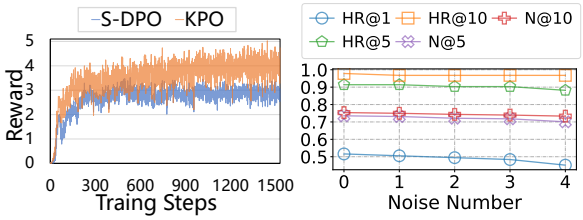
5.4.1 Sample Efficiency

As shown in Section §4.3.4, KPO and S-DPO exhibit similar runtimes. This section highlights how the K-layer loop in KPO improves sample efficiency. Fig.(5a) compares the reward curves of the top-1 item during training for both methods.

Fig.(5a) shows that KPO consistently outperforms S-DPO in reward with the same number of training steps. Moreover, KPO achieves the same reward level as S-DPO in fewer steps, underscoring its superior sample efficiency.

5.4.2 Robustness to Noisy Logits

Since using LLM logits to determine K candidates may introduce inaccuracies, we investigate how such errors impact KPO's training performance. To simulate these inaccuracies, we introduce a noise-adding mechanism that randomly swaps the logits of two items within M candidates, resulting in false top-K selections. We then assess performance as the number of swaps increases. Fig.(5b) presents the experimental results on the MovieLens validation dataset. The experiments demonstrate that KPO maintains relatively stable performance despite increasing noise levels, highlighting its robustness to imperfect logit estimates from the LLM.



(a) (b)

Figure 5: (a) Comparison of the reward of the top-1 item between KPO and S-DPO on MovieLens. (b) Study of noise on the MovieLens validation set.

6 Conclusion

In this study, we propose a novel method called KPO, designed to address the limitations of existing approaches that rely on full-order or partial-order ranking but often neglect the significance of top-K ranking. In detail, we introduce the K-order ranking, which prioritizes fine-grained ranking consistency for the top-K items while disregarding less relevant ones. Building on this foundation, we extend the PL model to accommodate top-K ranking and develop the corresponding KPO loss. Additionally, we derive a theoretical formula for the optimal accuracy achievable by KPO, thereby theoretically demonstrating that KPO outperforms S-DPO. Considering the varying number of relevant items across queries, we make KPO query-adaptive, enabling it to dynamically adjust K for each query. To further improve training efficiency and stability, we introduce K-aware curriculum learning, which allows LLMs to progressively learn from simpler to more complex data. Extensive experiments show that KPO significantly outperforms existing preference alignment methods, highlighting not only the effectiveness of top-K ranking but also the critical role of query-adaptive K.

Limitations

In this paper, we propose a query-adaptive KPO framework that dynamically determines the K-order for candidate items based on each query. While our approach has demonstrated effectiveness in experiments, the method for obtaining the query-adaptive K remains heuristic and does not guarantee that the resulting K is optimal.

On one hand, we rely on the logits generated by the LLM to represent the relevance between candidate items and the query. However, these logits may not always provide an accurate measure of relevance. It will be our future work to investigate more precise methods for assessing relevance.

On the other hand, our approach determines K by counting the number of items whose logits exceed a predefined threshold τ. This highlights that K is highly sensitive to the choice of this hyperparameter. In future work, we will explore strategies to derive a more accurate and optimal K.

References

Gediminas Adomavicius and Jingjing Zhang. 2016. Classification, ranking, and top-k stability of recommendation algorithms. *INFORMS J. Comput.*, 28(1):129–147.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

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. 2024a. Preference learning algorithms do not learn preference rankings. *CoRR*, abs/2405.19534.

Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. 2024b. On softmax direct preference optimization for recommendation. *CoRR*, abs/2406.09215.

Paolo Ciaccia and Davide Martinenghi. 2024. Directional queries: Making top-k queries more effective in discovering relevant results. *Proc. ACM Manag. Data*, 2(6):232:1–232:26.

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

Chenhao Fang, Xiaohan Li, Zezhong Fan, Jianpeng Xu, Kaushiki Nag, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2024. Llm-ensemble: Optimal large language model ensemble method for e-commerce product attribute value extraction. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2910–2914. ACM.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

F. Maxwell Harper and Joseph A. Konstan. 2016. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

DANIEL Kai-Ineman and Amos Tversky. 1979. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):363–391.

Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 197–206. IEEE Computer Society.

Wonbin Kweon, SeongKu Kang, Sanghwan Jang, and Hwanjo Yu. 2024. Top-personalized-k recommendation. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 3388–3399. ACM.

Dung D. Le and Hady Wirawan Lauw. 2021. Efficient retrieval of matrix factorization-based top-k recommendations: A survey of recent approaches. *J. Artif. Intell. Res.*, 70:1441–1479.

Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. 2023. Rethinking the role of token retrieval in multi-vector retrieval. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 1785–1795. ACM.

Sichun Luo, Bowei He, Haohan Zhao, Wei Shao, Yanlin Qi, Yinya Huang, Aojun Zhou, Yuxuan Yao, Zongpeng Li, Yuanzhang Xiao, et al. 2024. Recranker: Instruction tuning large language model as ranker for top-k recommendation. *ACM Transactions on Information Systems*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2421–2425. ACM.

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

Meta. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Eric Mitchell. 2023. A note on dpo with noisy preferences and relationship to ipo.

OpenAI. 2022. Introducing chatgpt. `https://openai.com/blog/chatgpt`.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeffrey 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 Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*, abs/2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *CoRR*, abs/2312.02724.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 1504–1518. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping queries dataset: A large-scale ESCI benchmark for improving product search. *CoRR*, abs/2206.06588.

Sofia Eleni Spatharioti, David M. Rothschild, Daniel G. Goldstein, and Jake M. Hofman. 2023. Comparing traditional and llm-based search for consumer choice: A randomized experiment. *CoRR*, abs/2307.03744.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 565–573. ACM.

Amos Tversky and Daniel Kahneman. 1992. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5:297–323.

Mengting Wan and Julian J. McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 86–94. ACM.

Zhengyi Yang, Xiangnan He, Jizhi Zhang, Jiancan Wu, Xin Xin, Jiawei Chen, and Xiang Wang. 2023. A

generic learning framework for sequential recommendation with distribution shifts. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 331–340. ACM.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? ID- vs. modality-based recommender models revisited. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2639–2649. ACM.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. Llamarec: Two-stage recommendation using large language models for ranking. *CoRR*, abs/2311.02089.

A Mathematical Derivation

A.1 Preference Modeling Derivation

In this section, we prove that the preference $y_1 \succ \cdots \succ y_K \succ \{y_{K+1}, \ldots, y_M\}$ can be expressed as:

$$\begin{aligned}&\hat{p}(y_1 \succ \cdots \succ y_K \succ \{y_{K+1}, \ldots, y_M\} \mid x)\\ &= \prod_{i=1}^{K} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))}.\end{aligned} \tag{15}$$

Specifically, based on Eq. (4), we can derive step by step as follows:

$$\begin{aligned}&\hat{p}(y_1 \succ \cdots \succ y_K \succ y_{K+1}, y_{K+2}, \ldots, y_M \mid x)\\ &= \sum_{\text{Per}(y_{K+1}, \ldots, y_M)} \prod_{i=1}^{M-1} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))}\\ &= \prod_{i=1}^{K} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))} \times\\ &\quad \sum_{\text{Per}(y_{K+1}, \ldots, y_M)} \prod_{i=K+1}^{M-1} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))}\\ &= \prod_{i=1}^{K} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))} \times\\ &\quad \sum_{\text{Per}(y_{K+1}, \ldots, y_M)} p(y_{K+1} \succ \cdots \succ y_M \mid x)\\ &= \prod_{i=1}^{K} \frac{\exp(r(x, y_i))}{\sum_{j=i}^{M} \exp(r(x, y_j))},\end{aligned} \tag{16}$$

where $\text{Per}(y_{K+1}, \ldots, y_M)$ denotes the set of all permutations of $y_{K+1}, \ldots, y_M$.

A.2 Proof of the Ranking Accuracy Theorem

In this section, we provide the proof of the theorem presented in Section §4.2, building on the method outlined in (Chen et al., 2024a).

Theorem 1. *Let π^* be the optimal policy that maximizes the KPO objective. Given a dataset of aggregated preferences $\mathcal{D}_p = \{(x, y_1 \succ \cdots \succ y_K \succ \{y_{K+1}, \ldots, y_M\}\}$. Assume $\mathcal{D}_p$ contains ground-truth ranking probabilitie following the PL model. Specifically, for any item y_i and the subset of remaining items $\{y_{i+1}, \ldots, y_M\}$, the ranking probability is defined as follows:*

$$\alpha(x, y_i, y_{>i}) = \mathbb{P}(y_i \succ \{y_{i+1}, \cdots, y_M\}) \tag{17}$$

The top-K ranking accuracy of π^* is given by:

$$\begin{aligned}&\mathcal{R}^*_{KPO}(\mathcal{D}_p, \pi_{\text{ref}})\\ &= \mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}_p} \left[\prod_{l=1}^{K} \prod_{k=l+1}^{M} \mathbb{I}\left[\frac{w_l \pi_{\text{ref}}(y_l \mid x)}{w_k \pi_{\text{ref}}(y_k \mid x)} > 1 \right] \right],\end{aligned} \tag{18}$$

where $\frac{w_l}{w_k}$ is defined as:

$$\frac{w_l}{w_k} = \left(\frac{\alpha(x, y_l, y_{>l})}{\alpha(x, y_k, y_{>k})} \right)^{1/\beta} \cdot \prod_{i=l}^{k-1} (1 - \alpha(x, y_i, y_{>i}))^{-1/\beta}. \tag{19}$$

Proof. Firstly, under the PL model, we have:

$$\mathbb{P}^*(y_i \succ \{y_{i+1} \cdots y_M\}) = \frac{\exp(r^*(x, y_i))}{\sum_{n=i}^{M} \exp(r^*(x, y_n))}. \tag{20}$$

Following DPO (Rafailov et al., 2023), we can express the ground-truth reward through its corresponding optimal policy:

$$r^*(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x). \tag{21}$$

We argue that, after thorough optimization, the optimal ranking probability $P^*(y_i \succ \{y_{i+1} \cdots y_M\})$ derived from the optimal strategy equals the ground-truth ranking probability $\alpha(x, y_i, y_{>i})$ defined in the dataset. Then we can derive that:

$$\alpha(x, y_i, y_{>i}) = \frac{\exp\left(\beta \log \frac{\pi^*(y_i|x)}{\pi_{\text{ref}}(y_i|x)}\right)}{\sum_{n=i}^{M} \exp\left(\beta \log \frac{\pi^*(y_n|x)}{\pi_{\text{ref}}(y_n|x)}\right)}. \tag{22}$$

Rearranging, we have:

$$\begin{aligned}&\frac{\alpha(x, y_l, y_{>l})}{\alpha(x, y_k, y_{>k})}\\ &= \frac{\exp\left(\beta \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)}{\exp\left(\beta \log \frac{\pi^*(y_k|x)}{\pi_{\text{ref}}(y_k|x)}\right)} \cdot \frac{\sum_{n=k}^{M} \exp\left(\beta \log \frac{\pi^*(y_n|x)}{\pi_{\text{ref}}(y_n|x)}\right)}{\sum_{n=l}^{M} \exp\left(\beta \log \frac{\pi^*(y_n|x)}{\pi_{\text{ref}}(y_n|x)}\right)}\\ &= \frac{\exp\left(\beta \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)}{\exp\left(\beta \log \frac{\pi^*(y_k|x)}{\pi_{\text{ref}}(y_k|x)}\right)} \cdot \prod_{n=l}^{k-1} (1 - \alpha(x, y_n, y_{>n})).\end{aligned} \tag{23}$$

Then we have:

$$\frac{\pi^*(y_l|x)}{\pi^*(y_k|x)} = \frac{w_l}{w_k} \frac{\pi_{\text{ref}}(y_l|x)}{\pi_{\text{ref}}(y_k|x)}, \tag{24}$$

where

$$\frac{w_l}{w_k} = \left(\frac{\alpha(x, y_l, y_{>l})}{\alpha(x, y_k, y_{>k})}\right)^{1/\beta} \cdot \prod_{i=l}^{k-1} (1 - \alpha(x, y_i, y_{>i}))^{-1/\beta}. \tag{25}$$

If we define for each $k = 1, \ldots, M$,

$$E_k = \left\{\pi^*(y_k \mid x) > \pi^*(y_j \mid x) \text{ for all } j = k+1, \ldots, K\right\}, \tag{26}$$

then the top-K ranking accuracy of π^* is given by:

$$\mathcal{R}^*_{\text{KPO}} = P\Big(\bigcap_{k=1}^{K} E_k\Big). \tag{27}$$

Finally, we can calculate the ranking accuracy as follows:

$$\begin{aligned}
&\mathcal{R}^*_{\text{KPO}}(\mathcal{D}_p, \pi_{\text{ref}}) \\
&= \mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}_p} \left[\prod_{l=1}^{K} \prod_{k=l+1}^{M} \mathbb{I}\left[\frac{\pi^*(y_l \mid x)}{\pi^*(y_k \mid x)} > 1\right]\right] \\
&= \mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}_p} \left[\prod_{l=1}^{K} \prod_{k=l+1}^{M} \mathbb{I}\left[\frac{w_l \pi_{\text{ref}}(y_l \mid x)}{w_k \pi_{\text{ref}}(y_k \mid x)} > 1\right]\right].
\end{aligned} \tag{28}$$

This complete the proof. □

A.3 Proof that KPO Outperforms S-DPO

Based on Theorem 1, we demonstrate in this section that KPO achieves a higher optimal ranking accuracy compared to S-DPO.

In detail, S-DPO models each data point as: $y_1 \succ \{y_2, \ldots, y_M\}$, which is a special case of KPO when $K = 1$. Thus, similar to the proof of Theorem 1 in Appendix A.2, we express $\frac{\pi^*(y_l|x)}{\pi^*(y_k|x)}$ as:

$$\frac{\pi^*(y_l|x)}{\pi^*(y_k|x)} = \frac{w'_l}{w'_k} \frac{\pi_{\text{ref}}(y_l|x)}{\pi_{\text{ref}}(y_k|x)}, \tag{29}$$

where

$$\begin{aligned}
\frac{w'_l}{w'_k} = &\left(\frac{\alpha(x, y_l, y_{\geq l})}{\alpha(x, y_k, y_{>k})}\right)^{1/\beta} \cdot \prod_{i=l}^{k-1} (1 - \alpha(x, y_i, y_{>i}))^{-1/\beta} \\
&\cdot \mathbb{I}[l = 1] + \mathbb{I}[l \neq 1].
\end{aligned} \tag{30}$$

As a result, the optimal ranking accuracy of S-DPO is:

$$\begin{aligned}
&\mathcal{R}^*_{\text{S-DPO}}(\mathcal{D}_p, \pi_{\text{ref}}) \\
&= \mathbb{E}_{(x, y_1, \ldots, y_M) \sim \mathcal{D}_p} \left[\prod_{l=1}^{K} \prod_{k=l+1}^{M} \mathbb{I}\left[\frac{w'_l \pi_{\text{ref}}(y_l \mid x)}{w'_k \pi_{\text{ref}}(y_k \mid x)} > 1\right]\right].
\end{aligned} \tag{31}$$

Next, we aim to prove that $\frac{w_l}{w_k} > \frac{w'_l}{w'_k}$ for all $l \in \{2, \ldots, K\}$ and $k \in \{l+1, \ldots, M\}$.

Since the ranking probabilities $\alpha(x, y_i, y_{>i})$ are provided by the dataset $\mathcal{D}_p$, this implies that

$$r^*(x, y_l) > r^*(x, y_k), \forall l < k. \tag{32}$$

Hence, we can derive that:

$$\left(\frac{\alpha(x, y_l, y_{>l})}{\alpha(x, y_k, y_{>k})}\right)^{1/\beta} \cdot \prod_{i=l}^{k-1} (1 - \alpha(x, y_i, y_{>i}))^{-1/\beta} > 1. \tag{33}$$

Therefore, we conclude that $\frac{w_l}{w_k} > \frac{w'_l}{w'_k}$ for all $l \in \{2, \ldots, K\}$ and $k \in \{l+1, \ldots, M\}$.

Subsequently, for $l \neq 1$, we have:

$$\mathbb{I}\left[\frac{w_l \pi_{\text{ref}}(y_l \mid x)}{w_k \pi_{\text{ref}}(y_k \mid x)} > 1\right] > \mathbb{I}\left[\frac{w'_l \pi_{\text{ref}}(y_l \mid x)}{w'_k \pi_{\text{ref}}(y_k \mid x)} > 1\right]. \tag{34}$$

Therefore, we conclude that $\mathcal{R}^*_{\text{KPO}}(\mathcal{D}_p, \pi_{\text{ref}}) > \mathcal{R}^*_{\text{S-DPO}}(\mathcal{D}_p, \pi_{\text{ref}})$.

A.4 Optimization Objective of KTO

In this section, we provide a detailed introduction to the optimization objectives of KTO (Ethayarajh et al., 2024).

Given that λ_D and λ_U are hyperparameters for desirable and undesirable outputs respectively, the KTO loss is defined as:

$$\mathcal{L}_{\text{KTO}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{x, y \sim \mathcal{D}}[\lambda_y - v_{\text{KTO}}(x, y)], \tag{35}$$

where

$$\begin{aligned}
r_\theta(x, y) &= \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \\
z_0 &= \text{KL}(\pi_\theta(y'|x) \| \pi_{\text{ref}}(y'|x)) \\
v_{\text{KTO}}(x, y) &= \begin{cases} \lambda_D \sigma(\beta(r_\theta(x, y) - z_0)) \text{ if } y \sim y_{\text{desirable}}|x \\ \lambda_U \sigma(\beta(z_0 - r_\theta(x, y))) \text{ if } y \sim y_{\text{undesirable}}|x \end{cases}
\end{aligned} \tag{36}$$

B Ground Truth Label

As mentioned in Section §4.3.3, we need to use the ground truth labels in the dataset to re-rank the top-K items. In practice, ground truth relevance labels are derived as follows:

- *Product Search Tasks*: Each candidate item is assigned a relevance score with respect to the query, typically from a discrete set such as $\{0, 1, 2, 3\}$.
- *Recommendation Tasks*: Only the item most recently interacted with by the user is typically considered relevant, while all other items are treated as irrelevant. This scenario can be seen as a special case where one item's relevance score is "1", and all others are assigned a score of "0".

C Experimental Settings

C.1 Datasets

In this section, we provide a detailed description of three datasets, as outlined below. The statistical information is presented in Table 5.

- *MovieLens*: This is a widely used dataset for movie recommendation tasks, containing user ratings for various movies and offering subsets of different sizes. Given the substantial computational demands of LLMs, we chose the MovieLens100K dataset for our experiments.
- *Goodreads*: This dataset comprises user ratings and reviews of books. To manage the dataset size, we filtered out users with fewer than 20 interactions on Goodreads.
- *Shopping Queries*: This dataset features a collection of challenging Amazon search queries and corresponding results. To limit its size, we excluded products associated with fewer than 5 queries.

Dataset	#Query	#Item	#Interaction
MovieLens	943	1,682	100,000
Goodreads	6,031	4,500	220,100
Shopping Queries	21,852	12,882	96,788

Table 5: **Statistics of datasets.**

C.2 Baselines

C.2.1 Preference Alignment Methods

We compare KPO with various preference alignment methods, including KTO (Ethayarajh et al., 2024), DPO (Rafailov et al., 2023), SimPO (Meng et al., 2024), Conservative DPO (cDPO) (Mitchell, 2023), S-DPO (Chen et al., 2024b), and DPO_{PL} (Rafailov et al., 2023). Detailed descriptions of these methods are provided below:

- *KTO*: Inspired by Kahneman and Tversky's prospect theory (Kai-Ineman and Tversky, 1979; Tversky and Kahneman, 1992), this method relies solely on binary labels, classifying samples as either "good" or "bad," which can be considered a point-wise approach.
- *DPO*: Provides a closed-form solution for the reward model in RLHF (Ouyang et al., 2022) and enables offline optimization of the pair-wise preference model.
- *SimPO*: Proposes a simplified optimization algorithm compared to DPO, eliminating the need for a reference model.
- *Conservative DPO (cDPO)*: Introduces a hyperparameter ϵ to account for the flip rate of noisy labels.
- *S-DPO*: Incorporates multiple negative samples in user preference data and develops an alternative DPO loss formulation tailored for LM-based recommenders, linked to softmax sampling strategies.
- DPO_{PL}: Extends DPO's Bradley-Terry modeling to the list-wise Plackett-Luce modeling.

C.2.2 Recommendation Models

We compare KPO with various recommendation models, which can be broadly classified into two categories: traditional models and LLM-based models.

The traditional recommendation models include:

- *SASRec*: An attention-based sequential recommendation model designed to effectively capture long-range semantic dependencies in user behavior sequences.
- *GRU4Rec*: A recurrent neural network (RNN)-based model known for its simplicity and efficiency in recommendation tasks.
- *Caser*: A convolutional neural network (CNN)-based model that interprets a user's historical behavior sequence as an "image" and leverages CNN operations to extract meaningful patterns.

The LLM-based recommendation models include:

- *MoRec*: A model that enhances traditional recommendation models by integrating modality-specific features of items.
- *LLaRA*: A hybrid model that combines LLM with the traditional models' embeddings through hybrid item representations.

C.3 Implementation Details

Our experiments are conducted on eight NVIDIA A40 GPUs. For the KPO method, we use the LLama-3.2-3B-Instruct (Meta, 2024) model as the backbone and apply LoRA (Hu et al., 2022) for fine-tuning. Specifically, the LoRA rank is set to 32, and the LoRA alpha is configured to 64. During the supervised fine-tuning (SFT) stage, the model is trained for 5 epochs with a learning rate of 1e-4. In the preference alignment stage, the learning rate is reduced to 1e-5, and training is performed over 3 epochs. The global batch size is fixed at 128. To ensure optimal performance, we select the model checkpoint that achieves the best results on the validation set. Additionally, a warm-up strategy is employed, where the learning rate is initialized to $\frac{1}{100}$ of its maximum value and gradually increased using a cosine scheduler. For traditional models, we adopt the settings outlined in (Yang et al., 2023), using a learning rate of 0.001, an embedding dimension of 64, and a batch size of

Method	MovieLens					Goodreads				
	HR@1	HR@5	HR@10	N@5	N@10	HR@1	HR@5	HR@10	N@5	N@10
SFT	0.5053	0.8526	0.9368	0.6983	0.7255	0.4809	0.8369	0.9468	0.6675	0.7034
Random	0.5579	0.8842	**0.9684**	0.7361	0.7620	0.5042	0.8719	0.9584	0.6994	0.7272
Descending	0.5474	0.8737	**0.9684**	0.7233	0.7532	0.4942	0.8686	0.9584	0.6949	0.7239
Ascending	**0.5684**	**0.8947**	**0.9684**	**0.7381**	**0.7637**	**0.5158**	**0.8735**	**0.9667**	**0.7024**	**0.7353**

Table 6: **Comparison of different training data orders.** Bold indicates the best performance.

256. To determine the optimal L2 regularization coefficient, we conduct a grid search over the values $[1e-3, 1e-4, 1e-5, 1e-6, 1e-7]$. For other LLM-based models, we follow the training protocol described in LLaRA (Liao et al., 2024), training the models for up to 5 epochs with a batch size of 128.

D Additional Experiments

D.1 Analysis of Time Complexity

As mentioned in Section §4.3.4, the additional K-layer loop introduced by KPO, compared to S-DPO, does not significantly increase the actual runtime. To support this claim, we conducted experiments to compare the runtime performance of KPO and S-DPO in practice.

We measured the average runtime of each phase during optimization on the MovieLens dataset using an NVIDIA A40 GPU with a batch size of 4. As shown in Table 7, KPO's total runtime is only 2% longer than S-DPO. This slight increase arises from Phases 1 and 3 dominating the computation, while the added complexity in Phase 2 has minimal impact. Therefore, KPO achieves runtime efficiency comparable to S-DPO despite its higher theoretical complexity.

Method	Complexity		Runtime			
	Phase 1	Phase 2	Phase 1	Phase 2	Phase 3	Total
S-DPO	$\Theta(M)$	$\Theta(M)$	2.82	0.03	8.04	10.89
KPO	$\Theta(M)$	$\Theta(K \cdot M)$	2.82	0.24	8.04	11.10

Table 7: **Results of time complexity and actual runtime.** "Complexity" refers to the number of iterations per phase, with execution time measured in seconds.

D.2 Shopping Queries Dataset with One Ground Truth Item

We also align the experimental setup of the Shopping Queries dataset with that of the recommendation dataset: a candidate item list is composed of one ground truth item and 19 randomly sampled items. The experimental results are presented in Table 8.

Based on the experimental results, we can conclude that KPO outperforms other preference alignment methods, which demonstrates the effectiveness of KPO.

Method	Shopping Queries				
	HR@1	HR@5	HR@10	N@5	N@10
KTO	0.5120	0.8430	0.9510	0.6891	0.7243
DPO	0.5210	0.8560	0.9550	0.6968	0.7288
SimPO	0.5210	0.8580	0.9630	0.6991	0.7331
cDPO	0.5240	0.8520	0.9540	0.6972	0.7304
S-DPO	0.5270	0.8420	0.9510	0.6936	0.7293
DPO_{PL}	0.5230	0.8560	0.9530	0.6997	0.7315
KPO_{CUT}	0.5220	0.8410	0.9480	0.6929	0.7279
KPO	**0.5330**	**0.8670**	**0.9670**	**0.7087**	**0.7414**

Table 8: **Comparison for optimization objectives on the Shopping Queries dataset.** Bold indicates the best performance.

D.3 K-aware Curriculum Learning

To demonstrate the effectiveness of K-aware curriculum learning, we present the performance of three training data orders—random, descending, and ascending—on the MovieLens and Goodreads test set. For better comparison, we also present the performance of the SFT model. The experimental results are summarized in Table 6.

From these results, we have drawn the following findings and conclusions: The model trained on "Ascending" data consistently outperforms those trained on "Random" and "Descending" data. This indicates that starting with simpler data and gradually progressing to more complex data is beneficial for improving model performance.