# Make Large Language Model a Better Ranker

Anonymous EMNLP submission


## Abstract

Large Language Models (LLMs) demonstrate robust capabilities across various fields, leading to a paradigm shift in LLM-enhanced Recommender System (RS). Research to date focuses on point-wise and pair-wise recommendation paradigms, which are inefficient for LLM-based recommenders due to high computational costs. However, existing list-wise approaches also fall short in ranking tasks due to misalignment between ranking objectives and next-token prediction. Moreover, these LLM-based methods struggle to effectively address the order relation among candidates, particularly given the scale of ratings. To address these challenges, this paper introduces the large language model framework with Aligned Listwise Ranking Objectives (ALRO). ALRO is designed to bridge the gap between the capabilities of LLMs and the nuanced requirements of ranking tasks. Specifically, ALRO employs explicit feedback in a listwise manner by introducing soft lambda loss, a customized adaptation of lambda loss designed for optimizing order relations. This mechanism provides more accurate optimization goals, enhancing the ranking process. Additionally, ALRO incorporates a permutation-sensitive learning mechanism that addresses position bias, a prevalent issue in generative models, without imposing additional computational burdens during inference. Our evaluative studies reveal that ALRO outperforms both existing embedding-based recommendation methods and LLM-based recommendation baselines.


## 1 Introduction

The rapid advancement in Large Language Models (LLMs), known by GPT-4 (OpenAI, 2023), has marked a significant milestone in demonstrating their versatility in zero-shot and few-shot learning across various domains. These models, effectively employed in domains like Question Answering and Information Retrieval, have shown remarkable adaptability and reliability. Their ability to efficiently handle tasks usually requiring extensive domain-specific training has sparked a surge in research aimed at exploring their potential across diverse applications, e.g. Recommender System.

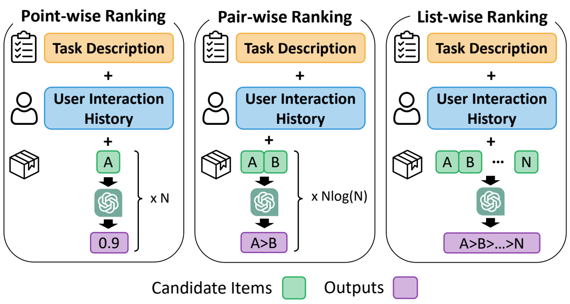



Figure 1: The comparison of point-wise, pair-wise, and list-wise ranking in LLM-based recommendation.

In the context of recommender systems, the application of LLMs has attracted considerable attention. Wu et al. (2023) demonstrates a novel paradigm in using Large Language Models as recommender systems. This approach leverages the natural language processing strengths for context-sensitive recommendations. Concurrently, investigations conducted in Bao et al. (2023) and Li et al. (2023) explore the capability of LLM in point-wise recommendation, revealing how language models can be adapted for suggesting products. Qin et al. (2023) investigate pairwise ranking prompts to enhance recommendation systems. Despite these advancements, as depicted in Figure 1, a significant limitation of these methods is their high computational cost, stemming from the iterative call of LLMs to evaluate each candidate item. Moreover, existing approaches focus on implicit feedback, filtering rating signals with predefined thresholds. This practice fails to effectively address partial order relations inherent in the magnitude of ratings.

In leveraging LLMs for recommendation systems, the list-wise ranking method stands out for its computational efficiency (Yue et al., 2023; Chen, 2023). However, executing list-wise ranking with explicit feedback effectively is fraught with challenges (Dai et al., 2023). The core issue lies in the objective misalignment between LLMs' natural language generation and ranking tasks. Specifically, ranking demands a sophisticated reasoning process to understand partial order relation within the sequence of candidates based on the ratings, which cannot be addressed by supervised fine-tuning with cross-entropy (Dai et al., 2023; Xu et al., 2024). Optimizing Large Language Models to interpret the magnitude of these ratings and the order relation among candidates remains a critical challenge in enhancing the ranking performance. Additionally, the inherent position bias in LLM-generated lists further complicates the matter. This bias indicates that the initial input ordering of the candidates significantly influences the final ranking of potential outputs. Although techniques like bootstrapping, suggested by Hou et al. (2023), offer a solution by iteratively querying the LLM with permuted candidate sequences to obtain unbiased arrangements, this method significantly increases computational demands. Such an increase is particularly problematic given the substantial resources required by Large Language Model operation, thereby highlighting a crucial trade-off between the precision and practicality of employing LLMs as recommender systems.

To overcome the aforementioned challenges, we propose the Large Language Model learning Framework with Aligned Listwise Ranking Objectives (ALRO), which integrates explicit feedback and soft lambda loss and permutation-sensitive learning into the training process to enhance the ranking capabilities of Large Language Models (LLMs). This enhancement is achieved through supervised fine-tuning and Low-Rank Adaptation (LoRA) (Hu et al., 2022). Specifically, ALRO employs a soft lambda loss that effectively bridges the gap between the objectives of ranking and language generation. This transformation emphasizes the significance of item orders within the predicted list, augmenting their impact during the language generation task. Furthermore, we introduce a permutation-sensitive learning framework designed to enhance ranking consistency by evaluating the distance between outputs from permuted candidate lists, thereby ensuring stable ranking outcomes regardless of candidates' input order. This strategy boosts the permutation invariance capability of the model, which is essential for reducing position bias. Through aligning distance metrics across original and permuted lists, our model effectively identifies and mitigates bias, enhancing the robustness and efficacy of the ranking process. The contributions of this paper are:

- We harmonize the goals of language generation and ranking tasks within a listwise framework using a novel soft lambda rank approach that incorporates explicit feedback, ensuring seamless integration of these objectives.
- We introduce a permutation-sensitive learning methodology that addresses position bias efficiently, without adding extra computational load during inference.
- We assess the performance of our model across four extensively used datasets, demonstrating its effectiveness.

## 2 Related Works

### 2.1 Large Language Model for Recommendation

Recent advancements in Large Language Models have showcased their formidable capabilities across a spectrum of tasks, drawing interest towards their potential in recommender systems (Qiu et al., 2021; Bao et al., 2023; Dai et al., 2023; Zhi et al., 2024). A comprehensive survey by Wu et al. (2023) listed the existing works on LLM-based Recommendations, particularly focusing on LLMs as agents that directly generate predictive outcomes. We delineated them into three paradigms, point-wise, pair-wise, and list-wise approaches.

The point-wise paradigm is characterized by the LLM processing each historical and candidate item pair individually. (Sachan et al., 2022; Zhi et al., 2024) For example, Bao et al. (2023) adapted the recommendation template to frame it as a yes-no question, requiring the LLM to evaluate each candidate sequentially. Another significant contribution is by Li et al. (2023) and Yue et al. (2023), who leveraged LLMs to recommend items through an adapter module that computes the probability of each item for recommendation. In the pair-wise paradigm, the LLM determines the preferable option between two candidate items. Qin et al. (2023) introduced a pair-wise prompting strategy employing a sliding window technique to identify the recommended items. Nonetheless, the point-wise and pair-wise approaches are notably inefficient due to the necessity of repeatedly calling the LLM, escalating the time cost as the number of candidates increases (Bao et al., 2023; Li et al., 2023; Kang et al., 2023) . In contrast, the listwise approach offers a more efficient solution by ranking the entire list of candidates in a single inference phase. Although some studies propose a listwise approach (Sun et al., 2023; Dai et al., 2023; Chen, 2023; Ma et al., 2023; Drozdov et al., 2023; Yue et al., 2023), they often address the problem with supervised fine-tuning, while falling short in handling the rating magnitude with metric-oriented LLM-based recommendation.

### 2.2 Learning to Rank

Learning to Rank (LTR) constitutes a fundamental component in information retrieval systems, aimed at ordering entities by their relevance. This domain is categorized into three main methodologies according to the design of the loss function: pointwise, pairwise, and listwise approaches. Pointwise methods focus on predicting the absolute relevance of individual items, typically framed as classification or regression tasks (Li et al., 2007; Crammer and Singer, 2001). Pairwise strategies, in contrast, emphasize the relative importance between item pairs, to accurately determine the more relevant item in a pair (Freund et al., 2003; Burges et al., 2005; Chapelle and Keerthi, 2010). The listwise approaches extend this concept by considering the entire item list as the training unit, aiming to directly optimize the overall item ordering to align with ranking objectives (Xu and Li, 2007; Cao et al., 2007; Taylor et al., 2008; Xia et al., 2008; Burges, 2010). In this paper, we present an innovative adaptation of the lambda loss function (Wang et al., 2018) tailored for natural language generation, leveraging the pairwise approach to enhance the coherence of generated texts. This adaptation underscores the potential of LTR methodologies to extend beyond traditional retrieval tasks.

## 3 Problem Statement

We define the sequential recommendation ranking problem as follows. Let $\mathcal{U}$ represent the set of users and $\mathcal{I}$ denote the set of items. For any given user $u \in \mathcal{U}$, their historical interactions with items are represented by $\mathcal{H}_u = \{h_1, h_2, \dots, h_k\}$, where each $h_i \in \mathcal{I}$ signifies an item that user $u$ has previously interacted with. With this notation in place, the ranking problem is formalized as follows:

**Definition 1** *For a user $u$, consider $\mathcal{C}_u = \{c_1, c_2, \dots, c_m\}$ as the set of candidate items for recommendation, where each $c_i \in \mathcal{I}$ and $m \leq |\mathcal{I}|$. The goal is to devise a ranking function $F : \mathcal{H}_u \times \mathcal{C}_u \rightarrow S_m$ that accurately predicts the permutation $\tau \in S_m$ that best orders the items in $\mathcal{C}_u$. The set $S_m$ is the symmetric group of all $m$-element permutations, encapsulating every possible arrangement of the candidate items.*

## 4 Methodology

In this section, we elucidate the constraints inherent in prevailing prompting paradigms when addressing list-wise recommendation tasks. Our learning framework is developed with four distinct components: Template Design, Supervised Fine-Tuning, Soft Lambda Loss, and Permutation-Sensitive Learning.

### 4.1 Template Design

Before delving into the specifics of our learning module, we delineate the process of transforming the ranking task into a language generation problem. Drawing inspiration from Instruction Tuning (Taori et al., 2023), we employ a natural language prompt template, denoted as $T_{\text{src}}(\mathcal{H}_u, \mathcal{C}_u)$, which transmutes the input user history $\mathcal{H}_u$ and context $\mathcal{C}_u$, inclusive of item attributes such as names, categories, and descriptions and their explicit rating from user, into a structured format. This transformation additionally aids in creating target text templates $T_{\text{tgt}}(\tau)$, representing the permutation that arranges candidate items according to user preferences. The detailed template design and example are provided in Appendix A.1.

### 4.2 Supervised Fine-Tuning

With the language generation problem that given $T_{\text{src}}(\mathcal{H}_u, \mathcal{C}_u)$ that aims to predict $T_{\text{tgt}}(\tau)$, we implement a supervised fine-tuning paradigm that leverages the Low-Rank Adaptation (LoRA) approach, as introduced by Hu et al. (2022). The core idea behind LoRA is to adapt pre-trained models in a parameter-efficient manner, enabling effective fine-tuning on downstream tasks with minimal modifications to the original model parameters. The fine-tuning process is formulated by the following loss function:

$$\mathcal{L}_{\text{sft}} = -\sum_{t=1}^{|y|} \log \left( P_\theta(y_t | x, y_{<t}) \right), \qquad (1)$$

where $\mathcal{L}_{\text{sft}}$ denotes the supervised fine-tuning loss, and $P_\theta(y_t|x, y_{<t})$ represents the conditional probability of predicting the token $y_t$ given the input tokens $x$ and the preceding tokens $y_{<t}$. In this context, $x$ and $y$ correspond to the tokenized representations of $T_{\text{src}}(\mathcal{H}_u, \mathcal{C}_u)$ and $T_{\text{tgt}}(\tau)$, respectively. This supervised fine-tuning process utilizes target tokens that correspond to the correctly ranked list of candidate answers, which are subsequently adjusted to reflect user preferences.

### 4.3 Soft Lambda Loss (SLL)

The widely adopted cross-entropy loss in language generation, derived from next-token prediction during supervised fine-tuning, faces a fundamental misalignment with the objectives of ranking. Such a discrepancy undermines the efficacy of cross-entropy loss when applied to the specific demands of ranking, leading to suboptimal performance in these contexts. To empower the Language Model with the capability to identify partial order relations, learning to rank (LTR) objectives serves as an effective supervised signal. Unlike the existing LTR framework (Wang et al., 2018), this is not straightforward to directly optimize on Normalized Discounted Cumulative Gain (NDCG) when dealing with language models that generate ranked token probabilities incrementally. Traditional ranking losses, such as Lambda loss (Wang et al., 2018) or SoftRank (Taylor et al., 2008), are not directly applicable. The Lambda loss, is defined as:

$$\mathcal{L}_{\text{rank}} = \sum_{i=1}^{|\tau|} \sum_{j:\tau_j<\tau_i} \delta_{i,j} |G_i - G_j| \cdot \log_2\left(1 + e^{-\sigma(s_i - s_j)}\right), \tag{2}$$

where

$$\delta_{ij} = \left| \frac{1}{D_{|i-j|}} - \frac{1}{D_{|i-j|+1}} \right|, \tag{3}$$

with $G_i$ and $D_i$ following the definitions from NDCG, and $s_i$ representing the model-derived prediction score. In large language models, the ranking order is typically determined by using the $argmax$ function on the output probabilities of tokens, which is non-differentiable and thus unsuitable for the training process.

To overcome this, we propose a method that combines the soft-argmax function with Lambda loss to calculate the deviation of predicted probabilities from the ideal ranking order. We define a differentiable ranking score for the generative model by substituting the traditional $argmax$ function in $s_i$ with the soft-argmax, expressed as:

$$s_i = \max_j \frac{e^{\gamma y_{j,i}}}{\sum_k e^{\gamma y_{j,k}}} \cdot j, \tag{4}$$

where $y_{i,j}$ denotes the output probability of the language model for the $j$th position and token $i$, and $\gamma$ represents the scaled value that adjusts the distribution of softmax. By making the computation of $s_i$ differentiable with the soft-argmax method, we align the objectives of language generation with those of the ranking task. Overall, Soft Lambda Loss follows the Equation 2, which is derived from Wang et al. (2018), by replacing $s_i$ with Equation 4 to get a differentiable objective.

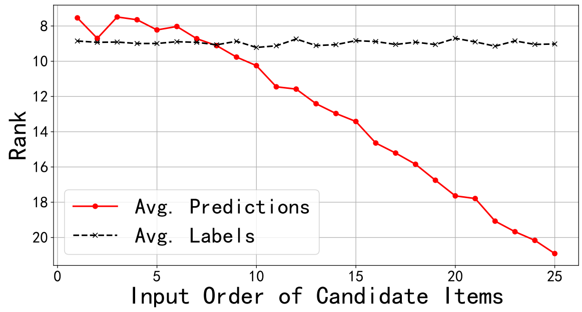



Figure 2: Demonstration of position bias. The figure shows how the placement of candidate items in the input sequence can significantly alter the ranking results produced by a Language Model.

### 4.4 Permutation-Sensitive Loss (PSL)

In list-wise recommendation tasks with Large Language Models, position bias emerges as a formidable challenge, with the order of the candidate input sequence notably swaying the ranking outcomes. As depicted in Figure 2, language models exhibit a propensity to assign higher rankings to candidates positioned at the beginning of the list. This tendency highlights the significant influence of candidate positioning on model evaluations, underscoring the imperative of developing methodologies to counteract these biases.

It is worth noting that the observed phenomenon depends exclusively on natural language generation tasks with the sequence of input candidates. This contrasts with embedding-based recommendation systems, where the order of inputs does not influence outcomes by calculating the score of the user and item pair separately. The effect of permutation on the output is described by the inequation:

$$F(T(\mathcal{H}_u, \mathcal{C}_u)) \neq F(T(\mathcal{H}_u, \mathcal{C}'_u)), \quad (5)$$

where $F(\cdot)$ denotes the logits output by large language model, and $\mathcal{C}'_u = \{c_{\pi(0)}\, c_{\pi(1)}, \cdots, c_{\pi(m)}\}$ represents a permuted candidate list from the original candidate list $\mathcal{C}_u$, with $\pi(\cdot)$ as the random permutation function that rearranges the candidates. This equation highlights the dependency of the model on the sequence in which inputs are provided, distinguishing it from conventional recommendation approaches which are order invariant.

Although Hou et al. (2023) proposed the bootstrapping method, which shuffles the candidate items multiple times and takes average scores as the final ranking result, it is inefficient as it repetitively calls language models in the inference stage to get average ranking. To alleviate this issue without burdening the inference in the recommendation, we propose a permutation-sensitive loss that aims to minimize the output distribution distance between the original candidate list $\mathcal{C}_u$ and the random permutated candidate list $\mathcal{C}'_u$ within the fine-tuning stage. . By adopting Kullback–Leibler divergence that minimizes the distance between two distributions, we empower the model with permutation invariant capability. The loss function could be formulated as:

$$\mathcal{L}_{\text{perm}} = \sum_t \text{KL}\left(P_\theta(y_t|x, y_{<t}) \| P_\theta(y'_t|x', y'_{<t})\right), \quad (6)$$

where $x$ and $x'$ are the prompt derived from $T(\mathcal{H}_u, \mathcal{C}_u)$ and $T(\mathcal{H}_u, \mathcal{C}'_u)$ respectively, and $y$ and $y'$ are the labels for the given prompts. The details of $\mathcal{C}'_u$, $y'_t$ and corresponding $P_\theta(y'_t|x', y'_{<t})$ are provided in Appendix A.2.

### 4.5 Training Objective

Overall, we provide the soft lambda loss $\mathcal{L}_{\text{rank}}$ with permutation-sensitive framework $\mathcal{L}_{\text{perm}}$ to address the issues mentioned above, which goes beyond the naive supervised fine-tuning. The objective function is reformulated as:

$$\mathcal{L} = \mathcal{L}_{\text{sft}} + \alpha\mathcal{L}_{\text{rank}} + \beta\mathcal{L}_{\text{perm}}, \quad (7)$$

where $\alpha, \beta$ are hyperparameters that adjust the importance of each loss.

## 5 Experiment

In our study, we conducted a comprehensive evaluation of our model across two real-world datasets. This was complemented by an ablation study, robustness tests, and efficiency evaluations. Our experiment was directed by the following pivotal research questions:

- **(RQ1)** Does the proposed framework surpass existing baselines in both embedding-based and LLM-based recommendation models?
- **(RQ2)** What extent does supervised fine-tuning on recommendation-specific corpus enhance Large Language Model performance?
- **(RQ3)** How crucial is the involvement of our proposed module for metrics improvement?
- **(RQ4)** How does permutation-sensitive learning compare to bootstrapping methods in terms of performance and efficiency?
- **(RQ5)** How does the ALRO framework improve performance across different parameter sizes of the backbone language model compared to traditional supervised fine-tuning?

Through these explorations, we aim to elucidate the contributions of domain-specific fine-tuning with our novel modules to the advancements in LLM-based recommendation systems.

### 5.1 Dataset

We selected four widely adopted open-source datasets to evaluate the effectiveness of our framework: *Movie* (MovieLens-1M[1]), *Music* (the "CDs & Vinyl" subset), *Books* (the "Books" subset), and *Games* (the "Toys and Games" subset) from the Amazon product reviews dataset. The Amazon product reviews datasets encompass reviews from 1996 to 2023 (Hou et al., 2024a) with 5-core. Detailed information about these datasets is presented in Table 1. To evaluate the model's capability of ranking explicit feedback, we sampled the most recent 25 user-interacted items as candidates $\mathcal{C}_u$, each with a rating $r_{c_i} \in [1 \,..\, 5]$. The output permutation $\tau$ is sorted from the candidate ratings $r_{c_i}$ provided by the user. The length of historical sequence $|\mathcal{H}_u|$ is set to 20. Followed by Kang and McAuley (2018), we split the user interaction sequence into three-part, 1) the most recent 25 actions for testing 2) the most recent 25 to 50 actions for validation 3) all remaining actions for training.

### 5.2 Baselines and Evaluation Metrices

To evaluate the effectiveness of our framework, we select several state-of-the-art baselines, which could be categorized into Non-Sequential Recom-

[1]https://grouplens.org/datasets/movielens/1m/

Table 1: Statistics of datasets.

| Dataset | Movie | Books | Games | Music |
|---|---|---|---|---|
| Users | 6040 | 54440 | 12182 | 9612 |
| Items | 3952 | 446987 | 114601 | 83937 |
| Actions | 1.0M | 9.27M | 1.53M | 1.58M |
| Density (%) | 4.19 | 0.0381 | 0.11 | 0.197 |
| Tokens/Item | 20.76 | 45.53 | 56.92 | 24.10 |

mendation, Sequential Recommendation, Ranking Methods, and Large Language Model-based Recommendation. We introduce the BERT-based model as the backbone to extract the textual information of items in both Non-Sequential Recommendation and Sequential Recommendation.

- **Non-Sequential Recommendation: NCF** (He et al., 2017) adopts a neural network with collaborative filtering for recommendations. **DIN** (Kang and McAuley, 2018) involves user interest modeling based on user behavior with an attention mechanism.
- **Sequential Recommendation: GRU4Rec** (Hidasi et al., 2016) is a session-based recommendation system utilizing a GRU-based recurrent network. **SASRec** (Hidasi et al., 2016) employs a self-attention network with positional embeddings to capture the user's sequential behavior information. **CORE** (Hou et al., 2022) uses a representation-consistent framework to unify the session and item representation spaces. **NARM** (Li et al., 2017) decomposes user behavior into global and local forms using attention networks for sequential recommendation.
- **Ranking Methods: Seq2Slate** (Bello et al., 2018) adopts RNN modules with a pointer network that maps candidate items to ranking positions in an end-to-end manner. **PRM** (Pei et al., 2019) utilizes a transformer-based network to re-rank lists by assigning scores to each candidate in a list-wise form.
- **Large Language Model-based Recommendation:** For **Zero-shot LLM** and **Few-shot LLM**, we follow list-wise setting in Hou et al. (2024b), which provides instructions and examples. **TallRec** (Bao et al., 2023) fine-tunes LLMs with instruction tuning for point-wise recommendation. **ES4Rec** (Li et al., 2023) introduces pre-trained item embeddings as prompts with an adapter to fine-tune the LLM. **LlamaRec** (Yue et al., 2023) employs a two-stage re-ranking framework for recommendation. We adopted the LLM re-ranking module in LlamaRec for ranking the candidates. We use Llama2-7b as the base model for all LLM-based baselines. It is worth noting that **ES4Rec** and **TallRec** require negative sampling data to maintain the performance of learning user embeddings, which imposes an additional burden on LLM training.

To assess the performance of various models in ranking tasks for explicit feedback, we employ Normalized Discounted Cumulative Gain (NDCG) at different cutoffs as our evaluation metric, specifically NDCG@k with $k$ values of 3, 5, 10, and 25.

### 5.3 Implementation Details

Our experiments were conducted on a cluster of 12 Linux servers, each equipped with 8 A800 GPUs. For the backbone model, we utilized the Llama2-7b [2] with BF16 precision, available on Huggingface. The supervised fine-tuning step was implemented using the PyTorch framework and peft library, applying the LoRA technique with a rank setting of 16. We used the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of 5e-5 and batch size as 128 for SFT, complemented by 2 gradient accumulation steps with a total of 10 training epochs. We also used Deepspeed (Rasley et al., 2020) with ZeRO stage as 2 for distributed training. For our loss function, we fine-tuned the hyperparameters, setting $\alpha$ equal to 1, $\beta$ equal to 2, and $\gamma$ equal to 2.

### 5.4 Overall Performance (RQ1)

To validate the performance of our proposed framework, ALRO, we executed comparative analyses against established baseline methods, with the results presented in Table 2. The following observations were made:

- ALRO consistently outperformed the baselines across various metrics and datasets, unequivocally demonstrating its superiority in ranking tasks within recommender systems.
- Large Language Models (LLMs) without fine-tuning fell short against traditional methods, highlighting the crucial role of supervised fine-tuning for LLMs in recommendation contexts.
- TALLRec achieves comparable performance but faces efficiency challenges.

These insights confirm the significance of our

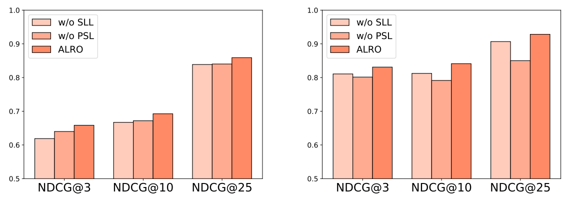
[2]https://huggingface.co/meta-llama/Llama-2-7b-hf

Table 2: Performance Comparison. Optimal outcomes across all models are emphasized in bold, while second-best performances are distinguished by underlining. Evaluation metrics include NDCG at ranks 3, 10, and 25.

| **Dataset** | **Movie** | | | **Books** | | | **Games** | | | **Music** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **NDCG** | @3 | @10 | @25 | @3 | @10 | @25 | @3 | @10 | @25 | @3 | @10 | @25 |
| **NCF** | 0.5804 | 0.6452 | 0.8336 | 0.7482 | 0.7692 | 0.9040 | 0.8245 | 0.8346 | 0.9356 | 0.7733 | 0.7918 | 0.9140 |
| **DIN** | 0.6067 | 0.6674 | 0.8437 | 0.7495 | 0.7711 | 0.9048 | 0.8240 | 0.8337 | 0.9351 | 0.7804 | 0.7946 | 0.9153 |
| **GRU4Rec** | 0.5545 | 0.6268 | 0.8241 | 0.7461 | 0.7679 | 0.9034 | 0.8258 | 0.8344 | 0.9355 | 0.7705 | 0.7858 | 0.9117 |
| **DIEN** | 0.5890 | 0.6580 | 0.8385 | 0.7508 | 0.7716 | 0.9051 | 0.8331 | 0.8376 | 0.9372 | 0.7728 | 0.7919 | 0.9133 |
| **SASRec** | 0.6436 | <u>0.6891</u> | 0.8427 | 0.7796 | 0.7961 | 0.9153 | <u>0.8501</u> | 0.8533 | <u>0.9432</u> | 0.7979 | 0.8135 | 0.9229 |
| **COREave** | 0.6236 | 0.6455 | 0.8299 | 0.7740 | 0.7899 | 0.9128 | 0.8498 | 0.8503 | 0.9409 | 0.7876 | 0.7957 | 0.9146 |
| **NARM** | 0.5281 | 0.6059 | 0.8145 | 0.7285 | 0.7555 | 0.8979 | 0.8282 | 0.8373 | 0.9366 | 0.7679 | 0.7863 | 0.9118 |
| **Seq2Slate** | 0.5320 | 0.6034 | 0.8178 | 0.7850 | 0.7961 | <u>0.9160</u> | 0.8292 | 0.8345 | 0.9357 | 0.7850 | 0.7961 | 0.9160 |
| **PRM** | 0.6088 | 0.6496 | 0.8384 | 0.7668 | 0.7858 | 0.9116 | 0.8235 | 0.8315 | 0.9344 | 0.7668 | 0.7858 | 0.9116 |
| **Zero-shot** | 0.5118 | 0.5954 | 0.8089 | 0.7322 | 0.7576 | 0.8989 | 0.8190 | 0.8317 | 0.9339 | 0.7639 | 0.7841 | 0.9106 |
| **Few-shot** | 0.5149 | 0.5958 | 0.8097 | 0.7337 | 0.7595 | 0.8995 | 0.8288 | 0.8358 | 0.9362 | 0.7746 | 0.7874 | 0.9128 |
| **TALLRec** | <u>0.6512</u> | 0.6835 | <u>0.8494</u> | <u>0.7895</u> | **0.8153** | 0.9141 | 0.8492 | 0.8469 | 0.9397 | <u>0.8221</u> | <u>0.8269</u> | **0.9295** |
| **E4SRec** | 0.5697 | 0.6210 | 0.8373 | 0.7620 | 0.7878 | 0.9089 | 0.8477 | <u>0.8545</u> | 0.9354 | 0.7795 | 0.7951 | 0.9203 |
| **LlamaRec** | 0.5360 | 0.6164 | 0.8193 | 0.7439 | 0.7687 | 0.9092 | 0.8402 | 0.8458 | 0.9366 | 0.7831 | 0.7945 | 0.9154 |
| **ALRO** | **0.6584** | **0.6925** | **0.8590** | **0.7903** | <u>0.7981</u> | **0.9190** | **0.8555** | **0.8582** | **0.9472** | **0.8310** | **0.8411** | <u>0.9283</u> |

Table 3: Comparison of Zero-shot, Few-shot and Supervised Fine-Tuning with Llama2-7b backbone.

| **Dataset** | **Movie** | | |
|---|---|---|---|
| **NDCG** | @3 | @10 | @25 |
| **Zero-shot** | 0.5118 | 0.5954 | 0.8089 |
| **Few-shot** | 0.5149 | 0.5958 | 0.8097 |
| **SFT** | **0.5712** | **0.6413** | **0.8258** |




(a) Movie dataset.

(b) Music dataset.

Figure 3: Ablation study on multiple datasets.

ALRO framework in enhancing the efficacy of ranking in recommendation systems and underscore the necessity for appropriate fine-tuning of LLMs to fully leverage their potential recommendation.

### 5.5 Effect of Supervised Fine-Tuning (RQ2)

Prompting techniques have showcased the profound ability of language models to interpret and execute tasks with remarkable precision. (Liu et al., 2023) However, the efficacy of these techniques is challenged when applied to specialized domains such as recommendation systems, particularly due to the potential misalignment between the pre-training corpus and the intricate requirements of ranking tasks. As depicted in Table 3, this discrepancy is notably pronounced in medium-sized language models like Llama-7b, where simple prompting may not suffice to activate the model's ranking capabilities effectively.

To address this gap, our study delves into the impact of supervised fine-tuning on the performance of language models in recommendation-related tasks. Through a comparative analysis encompassing zero-shot, few-shot, and supervised fine-tuning approaches, we unveil a substantial improvement in model performance by supervised fine-tuning, with metrics enhancing by over 10%. This improvement is attributed to the fine-tuning process, which effectively adjusts the model's outputs to better align with specific task requirements. This approach overcomes the shortcomings of conventional prompting techniques that often yield non-parsable outputs, thereby enhancing the model's ability to rank information more accurately.

### 5.6 Ablation Study (RQ3)

In our research, we conducted an ablation study to distinguish the contributions of distinct components within our proposed framework, systematically omitting each module for comparative analysis against the complete model. This involved evaluating two key variants: Exclusion of soft lambda loss (w/o SLL) and Exclusion of permutation-sensitive learning (w/o PSL). Figure 3 shows that

Table 4: Comparative analysis of bootstrapping and permutation-sensitive learning. 'p@i' denotes the number of permutations applied in bootstrapping. The original permutation represented by p@1 is consistent with the prompt in ALRO. TPD represents the average inference time per data sample, measured in seconds.

| **Dataset** | **Movie** | | | |
|---|---|---|---|---|
| **NDCG** | @3 | @10 | @25 | TPD |
| **p@1** | 0,6004 | 0.6203 | 0.8156 | 0.2546 |
| **p@3** | 0.6217 | 0.6842 | 0.8554 | 0.7654 |
| **p@5** | 0.6472 | **0.7032** | **0.8646** | 1.3756 |
| **ALRO** | **0.6584** | 0.6925 | 0.8590 | 0.2546 |

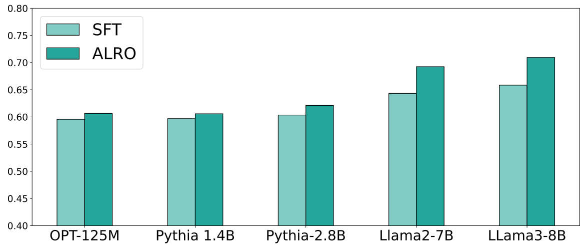



Figure 4: Enhancements achieved by ALRO across various model sizes on Movie dataset, measured using NDCG@10 metric.

both components significantly enhance the system's candidate ranking ability. The reduction in NDCG is attributed to the exclusion of the soft lambda loss, highlighting the importance of objective alignment in enhancing language models as recommender systems. Additionally, the performance drop from removing Permutation-Sensitive Learning underscores the impact of position bias on ranking performance.

### 5.7 Comparison of Bootstrapping and Permutation-Sensitive Learning (RQ4)

Our research introduces a permutation-sensitive learning approach designed to address position bias, which affects the outcomes based on the order of candidate lists. While the bootstrapping method (Hou et al., 2023) , offers a solution to this bias, it significantly increases inference time. We evaluated the effectiveness of permutation-sensitive learning compared to bootstrapping, aiming to reduce position bias without burdening the inference stage. Our comparisons included the original model without modifications, and bootstrapping with permutations executed 3 and 5 times. As demonstrated in Table 4, our method achieves comparable outcomes to bootstrapping while reducing inference times by approximately 5-fold. This indicates that our approach effectively mitigates the inference time issue through well-designed learning objectives.

### 5.8 Effect of Model parameter size (RQ5)

In this section of our research paper, we delve into the adaptability and efficacy of our learning framework across several LLM-based recommender systems, spanning various model sizes. Specifically, we selected four distinct models for our analysis: OPT-125M, Pythia 1.4B, Pythia-2.7B, Llama2-7B, Llama3-8B. By applying our framework to these models, we aim to showcase the consistent and significant performance enhancements it offers compared to traditional supervised fine-tuning approaches. As depicted in Figure 4, there is a clear correlation between model parameter size and performance, which serves to emphasize the capacity of our learning framework to augment the effectiveness of recommender systems across a spectrum of language model sizes. Notably, the enhancements provided by our framework are more significant in larger models than in smaller ones, this may be attributed to the innate reasoning capability of language models. Overall, the experiment highlights the versatility and broad applicability of our framework in improving system performance.

## 6 Conclusion

In this research, we tackled the intricacies of employing large language models as ranking agents in recommender systems with explicit feedback, focusing on refining list-wise ranking methods to manage the order relation. We proposed a cutting-edge framework that integrates soft lambda loss and permutation-sensitive learning. The integration of soft lambda loss is important as it bridges the objective between LLM's natural language generation and the specific demands of ranking tasks. It enhances the performance of ranking by optimizing the order relation within the magnitude of ratings. Furthermore, permutation-sensitive learning approaches effectively address the issue of position bias, providing an improvement over traditional bootstrapping methods without imposing additional computational demands during inference. Our comprehensive evaluation across various datasets confirms the success of our method, advancing LLMs as recommendation agents.

## 7 Limitation

While our framework adeptly aligns the objectives of ranking and language generation, it falls short in fully harnessing the explainability potential inherent in language models. The supervised fine-tuning process, augmented by joint loss optimization, effectively enhances the model's performance in listwise ranking tasks, particularly in recommendation systems. However, this process inadvertently undermines the model's proficiency in tasks beyond recommendation, limiting its versatility. Furthermore, although our method demonstrates efficacy in ranking a set of 25 items, scalability becomes a concern as the number of candidates increases significantly. This limitation arises due to constraints such as context limits or the propensity for forgetting in Large Language Models, compromising the model's ability to maintain performance consistency across varying candidate sizes. Typically, when dealing with large candidate sets, methods such as sliding windows (Sun et al., 2023) or retrieve-and-rank two-stage approaches (Yue et al., 2023) are employed to address scalability issues.

## References


Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*, pages 1007–1014. ACM.

Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Huai-hsin Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2slate: Reranking and slate optimization with rnns. *CoRR*, abs/1810.02019.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Olivier Chapelle and S. Sathiya Keerthi. 2010. Efficient algorithms for ranking with svms. *Inf. Retr.*, 13(3):201–215.

Zheng Chen. 2023. PALR: personalization aware llms for recommendation. *CoRR*, abs/2305.07622.

Koby Crammer and Yoram Singer. 2001. Pranking with ranking. pages 641–647.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*, pages 1126–1132. ACM.

Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald Metzler, and Kai Hui. 2023. Parade: Passage ranking using demonstrations with llms. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14242–14252. Association for Computational Linguistics.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 173–182. ACM.

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*.

Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. CORE: simple and effective session-based recommendation within consistent representation space. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 1796–1801. ACM.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian J. McAuley. 2024a. Bridging language and items for retrieval and recommendation. *CoRR*, abs/2403.03952.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian J. McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *CoRR*, abs/2305.08845.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian J. McAuley, and Wayne Xin Zhao. 2024b. Large language models are zero-shot rankers for recommender systems. In *Advances in Information Retrieval - 46th European Conference* on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part II*, volume 14609 of *Lecture Notes in Computer Science*, pages 364–381. Springer.

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.

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.

Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed H. Chi, and Derek Zhiyuan Cheng. 2023. Do llms understand user preferences? evaluating llms on user rating prediction. *CoRR*, abs/2305.06474.

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1419–1428. ACM.

Ping Li, Christopher J. C. Burges, and Qiang Wu. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 897–904. Curran Associates, Inc.

Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *CoRR*, abs/2312.02443.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *CoRR*, abs/2305.02156.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 3–11. ACM.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large language models are effective text rankers with pairwise ranking prompting. *CoRR*, abs/2306.17563.

Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-BERT: pre-training user representations for improved recommendation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4320–4327. AAAI Press.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3781–3797. Association for Computational Linguistics.

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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1313–1322.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860*.

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

Cong Xu, Zhangchi Zhu, Jun Wang, Jianyong Wang, and Wei Zhang. 2024. Understanding the role of cross-entropy loss in fairly evaluating large language model-based recommendation. *CoRR*, abs/2402.06216.

Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 391–398. 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.

Zheng Zhi, Chao Wenshuo, Qiu Zhaopeng, Zhu Hengshu, and Xiong Hui. 2024. Harnessing large language model in text-rich sequential recommendation. In *Proceedings of the ACM Web Conference 2024, WWW 2024*. ACM.

# A Appendix

## A.1 Template Design

We followed the template design from existing works (Bao et al., 2023; Yue et al., 2023) and refined the prompt to rank the items in a list-wise manner and alleviate position bias, as shown in Table 5. Specifically, the ranking results are sorted based on the rating $r_{c_i} \in [1 \dots 5]$. For candidates with equal ratings, we further sort them alphabetically. It is worth noting that while the equal rating results affect the supervised fine-tuning loss, they do not impact the soft lambda loss suggested in our framework, as the cumulative gain assigned in DCG for items with the same rating remains consistent.

## A.2 Permutation Sensitive Loss

We generate the candidate list $\mathcal{C}'_u = \{c_{\pi(0)}, c_{\pi(1)}, \cdots, c_{\pi(m)}\}$, which represents a permuted version of the original candidate list $\mathcal{C}_u$, where $\pi(\cdot)$ is a random permutation function. When the order of the candidate list is permuted, the corresponding target answer $T_{\text{tgt}}(\tau')$ also noted as $y'_t$ is adjusted to match the new order. For example, referring to Table 5, if we permute the candidates "Starman" and "Jumanji," the corresponding ranking result will change from "B A C ..." to "A B C ...". This permutation ensures that the model learns to rank based on the content rather than the position of the items in the list.

Regarding the probability distribution $P_\theta(y'_t|x', y'_{<t})$, our objective is to minimize the distance of the output distribution after permutation. Let $k_{\text{id}}$ represent the set of all token IDs and $k_\alpha$ represent the set of alphabetic tokens. When the targeted alphabetic token ID $k_\alpha$ changes according to the permutation function $\pi(\cdot)$, we apply the same permutation function to adjust the token categories in the target distribution. Mathematically, we aim to minimize the following loss:

$$\mathcal{L}_{\text{perm}} = \sum_t \text{KL}\left(P_\theta(y_t|x, y_{<t}) \| P_\theta(y'_t|x', y'_{<t})\right), \tag{8}$$

where $\text{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence, measuring the difference between the original output distribution and the permuted output distribution.

After applying the permutation $\pi(\cdot)$ on the candidate, the output tokens ID $k'_\alpha$ are changed. The permuted token ID set is $k'_\alpha = \pi(k_\alpha)$. Consequently, the target distribution must be adjusted to reflect the new order:

$$P_\theta(y'_{t,k'_\alpha}|x', y'_{<t,k'_\alpha}) = P_\theta(y'_{t,\pi^{-1}(k_\alpha)}|x', y'_{<t,\pi^{-1}(k_\alpha)}). \tag{9}$$

This means that the output distribution should accurately reflect the new order imposed by the permutation. The objective is to ensure that the output distribution of the permuted prompt $P_\theta(y'_t|x', y'_{<t})$ closely matches the original distribution $P_\theta(y_t|x, y_{<t})$, thereby maintaining the integrity of the ranking despite the permutation.

Table 5: Instruction Template and Example

| **Prompt Template** |
|---|
| **### Instruction:**<br>Given the user's interaction history, which reveals their items preferences, generate a preference-based ranking of the provided candidate items. Your task is to rank a list of new candidate movies.<br>Your ranking should include all the candidate movies provided, and it should be based solely on the user's preferences, without regard to the initial order of the candidates.<br>**### Input:**<br>**[User Interaction History]:**<br><User Interaction History><br>**[Candidate Items]:**<br><Candidate Items><br>**### Response:**<br>Given the historical interaction, the ranking result is:<br><Ranking Result> |
| **Example** |
| **### Instruction:**<br>Given the user's interaction history, which reveals their items preferences, generate a preference-based ranking of the provided candidate items. Your task is to rank a list of new candidate movies.<br>Your ranking should include all the candidate movies provided, and it should be based solely on the user's preferences, without regard to the initial order of the candidates.<br>**### Input:**<br>**[User Interaction History]:**<br>title: Independence Day genres: Action\|SciFi\|War rating: 3<br>title: Close Encounters of the Third Kind (1977) genres: Drama\|Sci-Fi rating: 4 . . .<br>**[Candidate Items]:**<br>(A) title: Starman genres: Adventure\|Drama\|Romance<br>(B) title: Jumanji (1995) genres: Adventure\|Children's \|Fantasy . . .<br>**### Response:**<br>Given the historical interaction, the ranking result is:<br>B A C . . . |