

Learning Mixtures of Plackett-Luce Models for Multi-Objective Alignment

Dongyue Li

Northeastern University, Boston, MA
li.dongyu@northeastern.edu

Ziniu Zhang

Northeastern University, Boston, MA
zhang.zini@northeastern.edu

Lu Wang

University of Michigan, Ann Arbor, MI
wangluxy@umich.edu

Hongyang R. Zhang

Northeastern University, Boston, MA
ho.zhang@northeastern.edu

Abstract

We consider the problem of learning a mixture of k Plackett-Luce models given multi-way ranking responses from annotators that may represent heterogeneous underlying preferences. This problem has many applications in AI alignment and preference optimization. Prior work has studied mixtures of Bradley-Terry models from pairwise comparisons. However, uncovering mixture models is theoretically unidentifiable when k exceeds $m/2$, where m is the length of a ranking. We propose an efficient implementation to address this limitation, which involves first augmenting the rankings to a larger size by generating new responses from a base language model, followed by a gradient-based estimation to reduce inference cost in the input embedding space. Based on this procedure, we then design an expectation-maximization algorithm with these two steps to fit a mixture of Plackett-Luce models, called MOPLEX. Extensive experiments are conducted to verify this approach. First, we show that the gradient-based approximation estimates true probabilities with less than 5% error on models with up to 34 billion parameters. Second, we show that MOPLEX improves clustering and ranking accuracy by an average of 43.7% and 15.2% over baselines using single ranking and mixtures of Bradley-Terry models, on preference optimization datasets. These results demonstrate the effectiveness of MOPLEX for tackling multi-way rankings from heterogeneous preferences through measuring alignment between gradients.

1 Introduction

We study language model alignment given preference data drawn from several heterogeneous subpopulations. The central challenge is to identify the clustering structures and learn a corresponding mixture of ranking models (Zhao et al., 2016; Awasthi et al., 2014). This problem arises in sce-

narios when preferences are collected from diverse annotator subpopulations or scored along several evaluation criteria (Kirk et al., 2024; Castricato et al., 2025). These different subgroups can rank candidate responses differently, yielding a mixture of rankings that a single ranking model cannot accurately capture (Chakraborty et al., 2024). In this paper, we tackle this problem by designing a new algorithm to learn a mixture of ranking models using a gradient-based estimation procedure.

One natural approach is to extend the Bradley-Terry (BT) model used in RLHF and DPO (Rafailov et al., 2023) to a mixture model. For example, recent work trains a mixture of BT models by regularizing mixture weights to a uniform prior (Shen et al., 2025), or by using an expectation-maximization algorithm to alternate between cluster assignment and model optimization (Chakraborty et al., 2024). However, these methods apply to rankings of two choices (i.e., pairwise comparisons), leaving learning mixtures on longer rankings open. While BT models can be generalized to Plackett-Luce (PL) models (Rafailov et al., 2023), existing methods focus on fitting single models rather than mixtures.

A key problem in this setting is identifiability, where two distinct mixture models can produce the same ranking probability distributions, making it hard to identify the true clusters. For example, consider ranking four responses drawn equally from two true clusters: $A \succ B \succ C \succ D$ and $B \succ A \succ D \succ C$. Observing only rankings of length $m \leq 3$ yields the same marginal distributions as another clustering ($A \succ B \succ D \succ C$ and $B \succ A \succ C \succ D$), as shown in Figure 3. We then formulate a broader pattern that a mixture of k models is unidentifiable when the ranking length m is short. Our synthetic experiments show that a mixture of BT models collapses to random guessing when $m > 2$, whereas increasing the ranking size beyond $2k$ recovers the true clusters.

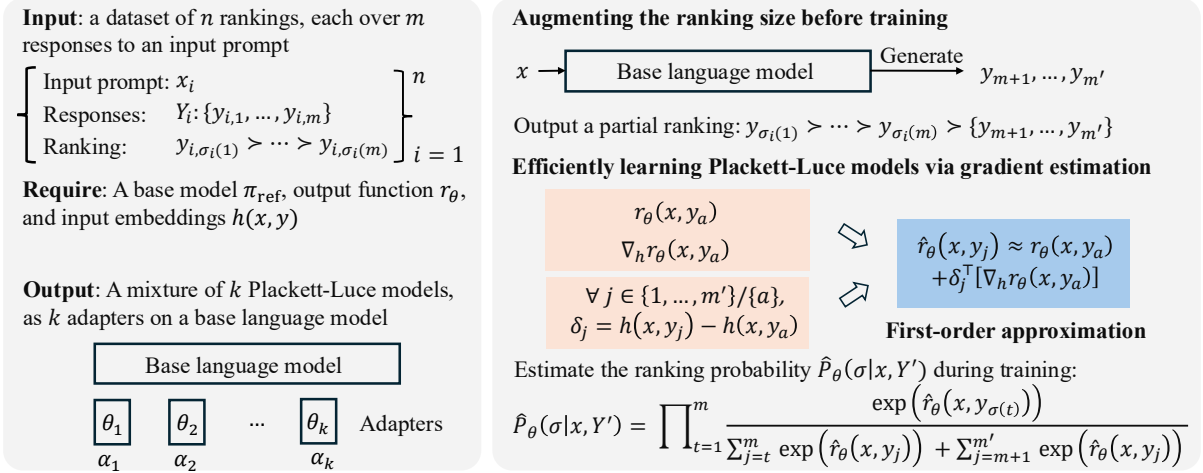


Figure 1: **Left:** Given n rankings over m candidate responses per prompt, we aim to learn a mixture of k Plackett-Luce models, which can be implemented via an ensemble of k adapters on a base language model. We identify an unidentifiability problem: when the ranking length is small (specifically, when $m \leq 2k - 1$), there exist two distinct mixture models that yield the same ranking distribution, making the true clusters unidentifiable. **Right:** We propose an algorithm to address this problem. First, we augment the ranking length by generating additional responses using the base model, outputting a larger partial ranking. Second, to reduce the computational cost, we estimate the model outputs on responses via a first-order approximation in the input embedding space. This only computes the outputs and gradients for a small number of a anchor responses, reducing the runtime and memory cost from $O(km)$ to $O(ka)$. The computational cost of the full procedure is given in Table 2.

This work proposes an algorithm that overcomes this limitation. First, to address insufficient ranking lengths, we augment the ranking length m by generating additional candidate responses using the base language model before training. Second, training the k -mixture ranking models over m candidates requires $O(km)$ forward passes per iteration, which is computationally expensive. We design a gradient-based estimation procedure to reduce the cost to $O(ka)$ with a typically three times smaller than m in practice. Our idea is to first compute exact outputs and gradients for the a anchor responses. Then, we can estimate outputs for the remaining responses using a first-order approximation in the input embedding space, without performing full model computation. Taken together, we design an algorithm named MOPLEX to efficiently scale the learning of mixtures of Plackett-Luce models on top of any base model. See Figure 1 for an illustration of our approach.

We extensively evaluate our approach across a range of preference optimization datasets and language models. First, our gradient estimation method approximates full model outputs with under 5% error on models with up to 34 billion parameters. Second, when applied to datasets involving multiple evaluation criteria and annotator subpopulations, our algorithm improves clustering

and ranking accuracy by **43.7%** and **15.2%**, respectively, compared to baselines that use a single ranking model or a mixture of BT models. Furthermore, our method reduces GPU runtime and memory costs by up to **3 $\times$** compared to full computation of Plackett-Luce models. By increasing ranking lengths three times, our method yields a **4.6%** performance gain over full models on the original short rankings. The code for replicating these empirical findings is available at <https://github.com/VirtuosoResearch/MoPLEx-implementation>.

2 Preliminaries

We study learning a mixture of Plackett-Luce models from a ranking dataset. In the context of text data, we are given rankings over text responses for an input prompt. Our input is a dataset $\mathcal{D} = \{(x_i, Y_i, \sigma_i)\}_{i=1}^n$, where each instance comprises a prompt x , a set of m candidate responses $Y = \{y_1, \dots, y_m\}$, and a multiway ranking σ . This ranking is a permutation over $[m]$ that defines an order $y_{\sigma(1)} \succ y_{\sigma(2)} \succ \dots \succ y_{\sigma(m)}$, where $\sigma(t)$ denotes the index of the response at rank t .

In practice, rankings are often gathered from diverse annotator populations or evaluation criteria. This can yield different rankings over the same candidate responses where a single rank-

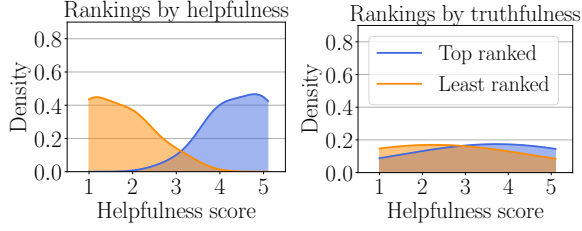


Figure 2: We plot the helpfulness scores for the top and bottom-ranked responses from the UltraFeedback dataset. **Left:** Responses ranked by helpfulness exhibit clear score separation. **Right:** Rankings based on truthfulness are nearly indistinguishable using helpfulness scores. Thus, a single ranking model cannot capture rankings given by various evaluation criteria.

ing model cannot adequately capture (Chakraborty et al., 2024; Shen et al., 2025). We illustrate an example in Figure 2. Thus, we model such ranking datasets by a mixture of k Plackett-Luce ranking models, each corresponding to a cluster, where k is treated as a hyperparameter. Our objective is to identify the cluster assignment for each ranking and to learn the corresponding ranking models.

Mixtures of Plackett-Luce models. Let $z_i \in \{1, \dots, k\}$ denote the cluster assignment for example i , and let α_c be the prior probability of cluster c , where $\sum_{c=1}^k \alpha_c = 1$. For a given cluster c , the likelihood of observing a specific ranking σ_i is given by a ranking model $P_{\theta_c}(\sigma_i | x_i, Y_i, z_i = c)$. The objective is to learn the model parameters $\{\theta_c\}_{c=1}^k$ and mixing weights $\{\alpha_c\}_{c=1}^k$ by maximizing the log-likelihood:

$$\sum_{i=1}^n \log \left(\sum_{c=1}^k \alpha_c \cdot P_{\theta_c}(\sigma_i | x_i, Y_i, z_i = c) \right). \quad (1)$$

We train the mixture model via the expectation-maximization algorithm. After training, each ranking σ_i is assigned to the cluster that yields the highest posterior probability.

The Plackett-Luce (PL) model is commonly used to compute the probability of a ranking. Let $r_{\theta_c}(x, y)$ be the score assigned by the model for cluster c . $P_{\theta_c}(\sigma_i | x_i, Y_i, z_i = c)$ is defined as:

$$\prod_{t=1}^{m-1} \frac{\exp(r_{\theta_c}(x_i, y_{i, \sigma_i(t)}))}{\sum_{l=t}^m \exp(r_{\theta_c}(x_i, y_{i, \sigma_i(l)}))}. \quad (2)$$

Substituting (2) into Equation (1) yields the mixture of PL models. The mixture of BT models is a special case when $m = 2$. Extensions to other ranking models are discussed in Appendix A.5.

In practice, we can fine-tune a language model to parameterize the score $r_{\theta_c}(x, y)$. For example, following DPO (Rafailov et al., 2023), we can define it as the scaled log-probability ratio:

$$r_{\theta_c}(x, y) := \beta \log \frac{\pi_{\theta_c}(y | x)}{\pi_{\text{ref}}(y | x)}, \quad (3)$$

where π_{ref} is a base reference model and β is a hyperparameter controlling the KL-divergence.

Problem statement. Given the ranking dataset $\mathcal{D}$ and a base training method like DPO, we aim to learn a mixture of PL models with the mixing weights and the model parameters $\{\alpha_c, \theta_c\}_{c=1}^k$. This can be implemented as k low-rank adapters on a base language model. We design algorithms to efficiently learn the mixture of PL models through fine-tuning language models.

3 Unidentifiability of Mixture Models

This section presents a limitation that the mixture of Plackett-Luce (PL) models is unidentifiable when the ranking size m is small. Unidentifiability occurs when two distinct sets of mixtures produce the same ranking distributions, making it impossible to recover the true clusters (See Definition A.1). This limits the ability of mixture Bradley-Terry models to identify multiple clusters.

Figure 3 illustrates this limitation using four candidate responses (A, B, C, D). Suppose the true distribution consists of two equal clusters: $A \succ B \succ C \succ D$ and $B \succ A \succ D \succ C$. If we only observe rankings of length $m \leq 3$, this setup yields the same marginal distributions as an alternative clustering ($A \succ B \succ D \succ C$ and $B \succ A \succ C \succ D$). Thus, it is difficult to distinguish the true clusters from alternative ones. In contrast, learning from rankings of length $m = 4$ can identify the clusters correctly.

Building on the insights from Zhao et al. (2016), we note that the unidentifiability of the mixture of PL models depends on the cluster number k and the ranking size m .

Proposition 3.1 (Slightly adapted from Zhao et al. (2016)). *For a mixture of k PL models defined over m -way rankings, when $m \leq 2k - 1$, the model is non-identifiable. That is, there exist two sets of distinct weights and parameters that induce an identical distribution over the space of m -way rankings.*

The result suggests that, for a larger number of clusters, more candidate responses are needed to

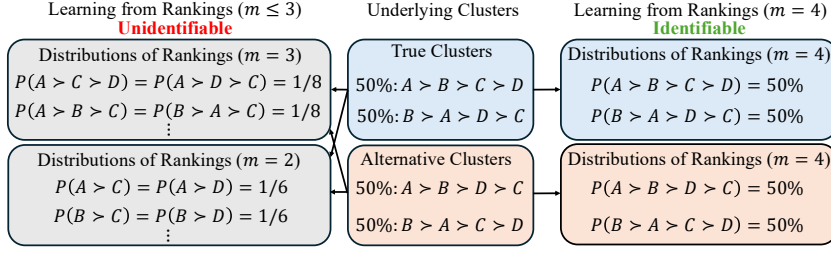


Figure 3: We illustrate the unidentifiability limit of a mixture model with $k = 2$. **Left:** If the given dataset contains rankings of length $m = 3$ (or $m = 2$), two possible cases of underlying clusters can yield the same marginal distributions. The mixture model is mathematically incapable of identifying the true clusters. **Right:** If the given dataset contains rankings of length $m = 4$, the two sets of clusters yield distinct distributions, and the mixture model can identify the correct underlying clusters.

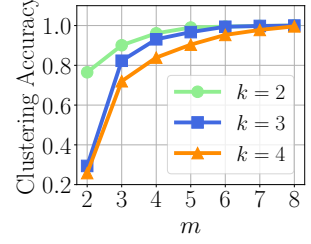


Figure 4: We validate the unidentifiability of the mixture of PL models in a synthetic experiment, with varying numbers of clusters k . We find that for a small m , the underlying clusters cannot be fully recovered.

identify the underlying clusters. The proof is deferred to Appendix A.3.

We now validate the above limitation of the mixture models. We construct a synthetic dataset to simulate various criteria. Each candidate response is represented by a list of k integers, where the c -th integer serves as the score for cluster $c \in \{1, \dots, k\}$. We train a Qwen-3-0.6B model, varying the ranking size m from 2 to 8 and the clusters k from 2 to 4. We then measure the clustering accuracy on a held-out set between the inferred and true cluster assignments. Details are described in Section B.

As shown in Figure 4, the underlying clusters cannot be fully recovered when m is small. Notably, relying on pairwise comparisons (when $m = 2$) collapses to random guessing for $k = 3$ and $k = 4$. Conversely, increasing the ranking size enables the model to identify the true clusters.

4 Our Approach

In this section, we present an efficient algorithm for learning mixture models. Our algorithm involves two components: First, we augment the ranking by generating additional responses. Second, we design an efficient gradient-based implementation for estimating Plackett-Luce mixtures on the expanded rankings.

4.1 Augmenting the ranking

First, we propose to augment the ranking by generating additional responses. For an input prompt x , we expand the response set to size m' by generating $m' - m$ new responses with the base model. Because no annotation is available for the generated responses, we treat them as ranked lower than

existing responses as an approximation. These new responses are appended as unranked candidates at the bottom of the existing ranking, resulting in a partial m' -way ranking: $y_{\sigma(1)} \succ \dots \succ y_{\sigma(m)} \succ \{y_{m+1}, \dots, y_{m'}\}$.

Our motivation is that the augmented responses are sampled from an unaligned base model, so they are likely to be ranked below the original candidates that are curated by annotators. To validate this, we score the generated responses using a ranking model trained on each evaluation criterion of the UltraFeedback dataset. Over 81% of the generated responses score below existing responses, and none is ranked in the top two positions. On the other hand, considering all possible cases of ranking all additional responses is computationally expensive.

Denote $Y' = Y \cup \{y_{m+1}, \dots, y_{m'}\}$. We compute the probability $P_{\theta_c}(\sigma \mid x, Y', z = c)$ as:

$$\prod_{t=1}^m \frac{\exp(r_{\theta_c}(x, y_{\sigma(t)}))}{\sum_{l=t}^m \exp(r_{\theta_c}(x, y_{\sigma(l)})) + Z'(r_{\theta_c})}, \quad (4)$$

where $Z'(r_{\theta_c}) := \sum_{j=m+1}^{m'} \exp(r_{\theta_c}(x, y_j))$. The scores of the augmented responses are included in the denominator. Then, we learn the mixture of PL models by substituting (4) into Equation (1).

Next, we show that with augmented responses, the identifiability can be achieved.

Proposition 4.1. *Let $Y^{\text{aug}} = \{y_{m+1}, \dots, y_{m'}\}$ be a set of additional responses, forming an expanded set $Y' = Y \cup Y^{\text{aug}}$ of size m' . Let $\mathcal{S}_{m',m}$ be the space of partial rankings where each $\sigma \in \mathcal{S}_{m',m}$ orders m responses of Y' and leaves the other $m' - m$ at the end of the ranking:*

$$y_{\sigma(1)} \succ \dots \succ y_{\sigma(m)} \succ \{y_{\sigma(m+1)}, \dots, y_{\sigma(m')}\},$$

so that $|\mathcal{S}_{m',m}| = m'!/(m' - m)!$, and the probability of each $\sigma \in \mathcal{S}_{m',m}$ is given by Equation (4). Assume that the augmented responses have distinct scores in every cluster c , meaning $r_{\theta_c}(x, y_i) \neq r_{\theta_c}(x, y_j)$ (in Equation 3) for all distinct $y_i, y_j \in Y^{\text{aug}}$. When $m' - m \geq k$ and $m \geq 2$, the mixture of k PL models over $\mathcal{S}_{m',m}$ is identifiable in the sense of Definition A.1.

The idea is that conditioned on the assumption of distinct scores for the generated responses, a matrix representing all possible ranking probabilities under the mixture model has sufficiently many independent rows to match its column dimension, thus exhibiting full column rank. The proof is given in Appendix A.4.

We empirically verify the assumption that generated responses have distinct scores. Recall that the score is computed via model log-likelihoods over the responses. Generating non-duplicate responses at a high sampling temperature can produce responses with distinct scores. We evaluate the scores of the generated responses on the UltraFeedback dataset. We find that using a sampling temperature of 2 yields no responses with the same scores under each evaluation criterion. Furthermore, we compute the coefficient of variation for their scores, computed as the standard deviation of the scores divided by their mean. The coefficient of variation is over 73%, indicating substantial variance in the scores.

4.2 Learning mixtures of Plackett-Luce models

Learning the mixture model on expanded rankings is computationally expensive. Evaluating a mixture of k PL models over m responses involves km full model forward passes at each iteration. Next, we introduce a method to reduce the computational cost from $O(km)$ to $O(ka)$ with a much smaller constant a than m .

We estimate $r_\theta(x, y)$ based on a first-order approximation property that has been observed widely on language models (Zhang et al., 2025c). The idea is to estimate the output of the model using a first-order approximation in the input embedding space. Let $h(x, y)$ be the input embedding of the response y for prompt x , and let $r_{\theta_c}(x, y)$ be a function of $h(x, y)$. Given an anchor example (x, y_0) , we examine the first-order approximation

Table 1: Relative approximation error of $\epsilon_{x,y}$, tested with language models of up to 34 billion parameters.

Distance	Qwen-0.6B	Gemma-2B	CodeLlama-34B
0% – 5%	$0.1 \pm 0.0\%$	$0.1 \pm 0.1\%$	$0.1 \pm 0.0\%$
5% – 10%	$0.6 \pm 0.1\%$	$0.5 \pm 0.1\%$	$0.6 \pm 0.0\%$
10% – 15%	$1.7 \pm 0.2\%$	$1.5 \pm 0.1\%$	$1.4 \pm 0.1\%$
15% – 20%	$2.9 \pm 0.1\%$	$3.2 \pm 0.3\%$	$3.3 \pm 0.3\%$
20% – 25%	$4.3 \pm 0.3\%$	$4.8 \pm 0.4\%$	$4.0 \pm 0.3\%$

of the model output $r_\theta(x, y)$ as:

$$r_\theta(x, y) \approx r_\theta(x, y_0) + \langle \nabla_h r_\theta(x, y_0), h(x, y) - h(x, y_0) \rangle + \epsilon_{x,y}, \quad (5)$$

where ∇_h denotes the gradient of r_θ with respect to the input embeddings h , and $\epsilon_{x,y}$ is the approximation error. For responses of different lengths, we apply the approximation up to the length of the shorter response and ignore padding tokens.

We demonstrate empirically that the error $\epsilon_{x,y}$ remains small across various models and datasets. We evaluate $\epsilon_{x,y}$ across three LMs from 0.6 to 34 billion parameters, on the UltraFeedback dataset (Cui et al., 2024). For each example (x, Y, σ) , we randomly select one anchor response and estimate the outputs of the remaining responses. We compute the estimation errors over 50 randomly selected examples. Table 1 reports the average squared relative error, $(\epsilon_{x,y}/r_\theta(x, y))^2$, grouped by the relative distance in their input embeddings, $\|h(x, y) - h(x, y_0)\|/\|h(x, y_0)\|$.

We find that the estimation error remains less than 5%. Additionally, the model with more parameters tends to yield a smaller error. These results suggest that Equation 5 can accurately approximate the model output for estimating ranking probabilities. Similar results are observed on other datasets and are described in detail in Table 5 of Appendix B.

We then leverage this approximation to efficiently train the mixture of PL models. Specifically, we apply the first-order approximation to estimate the output for each response in a PL model. At each iteration, we randomly sample a subset of a anchor responses, $Y_a \subseteq Y$. For each anchor $y_i \in Y_a$, we compute the exact output $r_\theta(x, y_i)$ and its gradient $g_i = \nabla_h r_\theta(x, y_i)$. For the remaining $m - a$ candidates, we estimate their outputs $\hat{r}_\theta(x, y_j)$ by applying Equation 5 and averaging the first-order approximations derived from all a

Algorithm 1 Gradient-based Estimation of Ranking Probabilities

Input: Input prompt x , candidate responses $Y = \{y_1, \dots, y_m\}$, ranking σ , current model r_θ

Require: Number of anchors a

```

1:  $Y_a \leftarrow$  Sample  $a$  anchor responses from  $Y$ 
2: for  $y_i \in Y_a$  do ▷ Exact evaluations
3:    $r_\theta(x, y_i) \leftarrow$  Model output
4:    $g_i = \nabla_h r_\theta(x, y_i) \leftarrow$  Gradient over the input embedding space
5: end for
6: for  $y_j \in Y \setminus Y_a$  do ▷ Approximation
7:   for  $y_i \in Y_a$  do
8:      $\delta_{i,j} \leftarrow h(x, y_j) - h(x, y_i)$ 
9:   end for
10:   $\hat{r}_\theta(x, y_j) \leftarrow \frac{1}{a} \sum_{y_i \in Y_a} (r_\theta(x, y_i) + g_i^\top \delta_{i,j})$ 
11: end for
12:  $\hat{P}_{\theta_c} \leftarrow$  Estimate ranking probability in Equation 4 using the estimated scores
13: Return  $\hat{P}_{\theta_c}$ 

```

anchors:

$$\frac{1}{a} \sum_{y_i \in Y_a} \left(r_\theta(x, y_i) + g_i^\top (h(x, y_j) - h(x, y_i)) \right).$$

The procedure is described in Algorithm 1.

Next, we train a mixture of PL models with the expectation-maximization algorithm, which alternates between two steps. First, given $\theta^{(t)}$ and $\alpha^{(t)}$ at step t , we compute the posterior probability that the ranking σ_i was generated by cluster c :

$$\gamma_{i,c} := \frac{\alpha_c^{(t)} \cdot \hat{P}_{\theta^{(t)}}(\sigma_i \mid x_i, Y_i, z_i = c)}{\sum_{l=1}^k \alpha_l^{(t)} \cdot \hat{P}_{\theta^{(t)}}(\sigma_i \mid x_i, Y_i, z_i = l)}. \quad (6)$$

Second, we update the parameters by maximizing the log-likelihood. The prior probabilities are updated as: $\alpha_c^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \gamma_{i,c}$. Then, we update model parameters by minimizing the weighted negative log-likelihood as:

$$- \sum_{c=1}^k \sum_{i=1}^n \gamma_{i,c} \log \hat{P}_{\theta_c^{(t)}}(\sigma_i \mid x_i, Y_i, z_i = c). \quad (7)$$

We do not backpropagate gradients on the outputs estimated by gradients.

Taken together, we summarize the complete procedure in Algorithm 2. We provide a comparison of our approach to related methods in Table 2. We also discuss extensions of our approach to learning a mixture of Mallows models in Appendix A.5.

Algorithm 2 Learning a Mixture of Plackett-Luce Models via Gradient Estimation (MOPLEX)

Input: n rankings $\mathcal{D} = \{(x_i, Y_i, \sigma_i)\}_{i=1}^n$, a base language model π_{ref}

Require: Number of clusters k , ranking size m' , number of anchors a , learning rate η , number of training epochs T

```

1: for  $(x_i, Y_i, \sigma_i) \in \mathcal{D}$  do ▷ Augmentation
2:    $Y_i^g \leftarrow$  Generate  $m' - m$  responses by  $\pi_{\text{ref}}$ 
3:    $Y_i' \leftarrow Y_i \cup Y_i^g$ 
4: end for
1:  $\theta_c^{(0)} \leftarrow$  Initialize from  $\pi_{\text{ref}}$  for  $c \in \{1, \dots, k\}$ 
2:  $\alpha_c^{(0)} \leftarrow 1/k$  for  $c \in \{1, \dots, k\}$ 
3: for  $t = 0, \dots, T - 1$  do
4:   for  $(x_i, Y_i', \sigma_i) \in \mathcal{D}$  do ▷ E-step
5:      $\hat{P}_{\theta_c^{(t)}} \leftarrow$  Apply Algorithm 1 to estimate ranking probability for  $c \in \{1, \dots, k\}$ 
6:      $\gamma_{i,c} \leftarrow$  Compute the posterior probability in Equation (6) for  $c \in \{1, \dots, k\}$ 
7:   end for
8:   for  $c \in \{1, \dots, k\}$  do ▷ M-step
9:      $\alpha_c^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n \gamma_{i,c}$ 
10:     $\hat{P}_{\theta_c^{(t)}} \leftarrow$  Apply Algorithm 1 to estimate ranking probability
11:     $\theta_c^{(t+1)} \leftarrow$  Update model parameters by SGD on the loss in Equation (7)
12:   end for
13: end for
14: Return  $\{\theta_c^{(T)}, \alpha_c^{(T)}\}_{c=1}^k$ 

```

5 Experiments

We evaluate our algorithm by answering the following questions: How accurately does our algorithm identify underlying clusters and rank responses, particularly compared to baselines using mixtures of BT models? To what extent does our algorithm reduce computational and memory overhead? How does each component of our algorithm affect downstream performance?

We extensively evaluate our methods across datasets involving multiple evaluation criteria and diverse user demographics. Our method outperforms baselines using single ranking models and mixtures of BT models by **43.7%** and **15.2%** in clustering and ranking accuracy, respectively. Furthermore, it reduces computational and memory costs by up to **3×**, matching the performance of training PL models with full computation. Lastly, our ablation studies show that every component of

Table 2: A summary of the runtime and memory usage by MOPLEX. Here, k is the number of mixture components, m is the number of responses in rankings, a is the number of anchor responses. T is the runtime of the base training method, and A is the memory of the model on one response. In practice, the runtime and memory overhead (T_g and A_g) in the gradient estimation is much smaller than T and A . a is typically more than three times smaller than m .

Method	Runtime	Memory
Single Ranking Model	mT	mA
Mixtures of Ranking Models	kmT	kmA
MOPLEX	$kaT + T_g$	$kaA + A_g$

our algorithm effectively contributes to modeling rankings from heterogeneous subpopulations.

5.1 Experimental setup

Datasets and models. First, we consider settings with multiple evaluation criteria, each as a distinct cluster. We use the UltraFeedback dataset (Cui et al., 2024), which ranks four candidate responses ($m = 4$) per prompt across four criteria: helpfulness, honesty, instruction-following, and truthfulness ($k = 4$). We use a subset where these criteria have different rankings, yielding 3,612 training, 448 validation, and 452 test rankings.

Second, we consider settings where rankings originate from diverse user demographics, treating each as a cluster. We use the Persona dataset (Castricato et al., 2025), which provides pairwise comparisons ($m = 2$). We select a subset of 12 distinct profiles ($k = 12$) spanning two age ranges, two sexes, and four races, resulting in 1,905 training, 237 validation, and 258 test rankings. We use Qwen-3-0.6B as the base language model. On both datasets, there is no overlap of the input prompts between the training and test datasets, which avoids data leakage at test time.

Baselines. We compare against two types of representative baselines: (1) single ranking models, DPO (Rafailov et al., 2023) and its extensions; and (2) mixtures of BT models (MaxMin-RLHF (Chakraborty et al., 2024), MiCRo (Shen et al., 2025), and EM-DPO (Chidambaram et al., 2025)). We train the same base language model for all baselines.

Implementations. We use DPO as the base method and fine-tune with LoRA adapters. We set k to 4 and 12 for the datasets, respectively. We set the ranking sizes m' between 4 and 32 and the num-

Table 3: We report the average clustering and ranking accuracy (%) on the UltraFeedback and Persona datasets. We compare MOPLEX against baselines that train single ranking models and mixtures of BT models. We run each experiment with three random seeds and report the average results with standard deviations.

Methods	UltraFeedback	Persona
Clustering Accuracy		
MiCRo	26.6 ± 2.1	19.6 ± 0.4
MaxMin-RLHF	27.2 ± 0.4	18.4 ± 0.2
EM-DPO	26.9 ± 2.1	18.3 ± 0.5
MOPLEX (Alg. 2)	70.9 ± 2.3	57.1 ± 0.5
Ranking Accuracy		
DPO	58.2 ± 1.5	57.7 ± 0.2
LiPO	63.3 ± 0.6	57.1 ± 0.8
MiCRo	57.7 ± 0.4	53.2 ± 0.8
MaxMin-RLHF	60.6 ± 0.7	49.8 ± 0.1
EM-DPO	59.3 ± 0.9	49.6 ± 1.5
MOPLEX (Alg. 2)	75.0 ± 2.9	76.4 ± 0.6

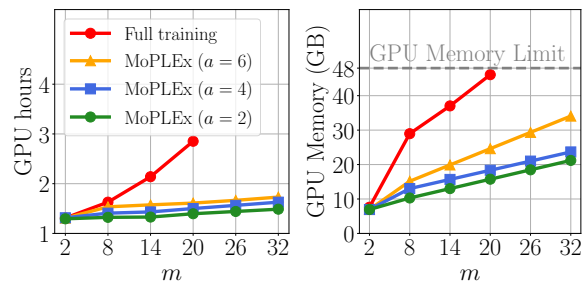


Figure 5: We compare the number of GPU hours and GPU memory between our method and fully training the mixtures of PL models, evaluated on the Persona dataset with $k = 12$ clusters. Our method reduces the runtime and memory cost by up to $3\times$, with comparable performance. We evaluate the training by using a batch size of 2 and a context length of 1024.

ber of anchors a between 1 and 8. The effects of varying k , m' , and a are analyzed in ablation studies.

5.2 Experimental results

Evaluation metrics. We evaluate the mixture models on both clustering and ranking accuracy. For clustering, we assign each ranking to the cluster with the highest posterior probability. Due to permutation, we align inferred labels with ground-truth clusters via optimal matching for computing the accuracy. For ranking, we measure how accurately each model predicts the rankings of every pair of responses within its assigned cluster, reporting the average across all clusters.

Table 3 summarizes the performance on the

Table 4: We report the ranking accuracies (%) on the UltraFeedback dataset across individual evaluation dimensions and their average. Our approach is compared against baselines using a single ranking model and mixtures of BT models. In addition, we report the ablation study results of leaving one component of our algorithm out on the UltraFeedback dataset. Ranking accuracy is measured as the proportion of correctly predicted pairs out of the six possible pairs per four-response ranking. All results report the mean and variance over three random seeds.

Ranking accuracy (%)	Helpfulness	Honesty	Instruction-following	Truthfulness	Average
DPO	56.4 $\pm$ 3.5	53.1 $\pm$ 0.2	60.0 $\pm$ 1.6	72.2 $\pm$ 0.0	58.2 $\pm$ 1.5
LiPO	63.8 $\pm$ 0.9	58.8 $\pm$ 0.9	56.2 $\pm$ 0.5	74.5 $\pm$ 0.4	63.3 $\pm$ 0.6
MiCRo	58.8 $\pm$ 1.4	52.1 $\pm$ 2.9	57.2 $\pm$ 3.9	70.8 $\pm$ 2.0	57.7 $\pm$ 0.4
MaxMin-RLHF	73.0 $\pm$ 0.6	51.1 $\pm$ 0.1	57.8 $\pm$ 1.6	48.6 $\pm$ 1.9	60.6 $\pm$ 0.7
EM-DPO	65.7 $\pm$ 8.3	54.2 $\pm$ 7.4	58.3 $\pm$ 0.8	61.9 $\pm$ 2.8	59.3 $\pm$ 0.9
MOPLEX (Alg. 2)	79.4 $\pm$ 3.9	73.0 $\pm$ 1.3	76.9 $\pm$ 2.6	70.5 $\pm$ 1.3	75.0 $\pm$ 2.9
w/o Plackett-Luce models	75.7 $\pm$ 2.1	71.3 $\pm$ 3.7	70.4 $\pm$ 2.3	69.2 $\pm$ 2.3	71.6 $\pm$ 2.7
w/o generated responses	75.0 $\pm$ 2.9	67.2 $\pm$ 2.2	72.2 $\pm$ 2.6	68.3 $\pm$ 2.7	70.7 $\pm$ 2.6
w/o mixture models	71.2 $\pm$ 2.8	63.8 $\pm$ 1.4	66.4 $\pm$ 0.8	60.6 $\pm$ 2.1	65.5 $\pm$ 1.8
w/o gradient estimation	80.4 $\pm$ 2.7	72.2 $\pm$ 1.7	76.8 $\pm$ 2.8	72.8 $\pm$ 2.7	75.6 $\pm$ 2.3

UltraFeedback and Persona datasets. MOPLEX yields a **43.7%** improvement in clustering accuracy over mixture-of-BT baselines (including MiCRo, MaxMin-RLHF, and EM-DPO). Furthermore, MOPLEX achieves a **15.2%** relative improvement in ranking accuracy over these baselines on average. Results for each cluster on the UltraFeedback dataset are described in Table 4, and the rest is described in Appendix B.

We further evaluate the efficiency by measuring the training runtime in GPU hours and GPU memory usage in GB. Figure 5 compares our method against a mixture of PL models with full computation on the Persona dataset, varying the ranking size m from 2 to 32 and the number of anchors a between 2, 4, and 6. Notably, fully training the PL model exceeds the 48GB memory limit whenever $m > 20$. By using $a = 2$ anchors, our method maintains comparable performance while reducing runtime and memory overhead by $2\times$ and $3\times$, respectively. Furthermore, with $a = 6$ anchors, our approach scales to rankings of $m = 32$ and outperforms the best fully trained PL models (at $m = 20$) by 4.6%, while using $1.6\times$ less runtime and $1.9\times$ less memory.

5.3 Ablation studies

Effect of each component in MOPLEX. First, we conduct an ablation study to isolate the effect of each component in our algorithm. We compare our approach against four variants: (1) replacing Plackett-Luce models with Bradley-Terry models by transforming multi-way rankings into pairwise comparisons (w/o PL models); (2) training the PL mixture model without generating new responses (w/o generated response); (3) training a single PL

model instead of a mixture of PL models (w/o mixture models); and (4) using exact computation for all responses instead of gradient estimation (w/o gradient estimation). We report the results in Table 4 on the UltraFeedback dataset. The results show that our algorithm outperforms the first three variants. Furthermore, it matches the ranking accuracy of training a mixture of PL models with full computation while reducing GPU memory costs by half.

Effect of generated responses diversity. Next, we evaluate how the diversity of generated responses affects the performance of our algorithm. To control the diversity, we train our algorithm on the UltraFeedback dataset, varying the sampling temperature during generation between 0.5, 1.0, 2.0, and 4.0. The higher temperature tends to yield more diverse responses. We show the ranking accuracy for the four criteria and their average in Table 8 of Appendix B. The results show that a high sampling temperature (around 2.0) yields diverse responses and benefits ranking accuracy. Higher temperatures degrade the quality of the generated responses and reduce accuracy.

Choice of hyperparameters. We conduct ablation studies to analyze hyperparameters on the Persona dataset. Increasing k beyond 12 yields no significant gains in clustering or ranking accuracy. Second, expanding the augmented ranking size m' from 2 to 32 improves performance. These gains plateau after $m' = 26$. Finally, although more anchors improve approximation, increasing a beyond 6 provides no obvious gain to overall performance. We report the full results in Appendix B.

6 Related Work

Efficient learning from ranking data is an important problem in areas like social choice and language model alignment (Conitzer et al., 2024). To capture more complex rankings, prior works have modeled intransitive rankings (Makhijani and Ugander, 2019), partial rankings (Awadelkarim and Ugander, 2024), and ranking distributions centered around multiple orderings (Seshadri et al., 2020). Most relevant to our study is learning mixture models for ranking. Prior work has established identifiability conditions for Plackett-Luce (PL) models and designed a generalized method of moments to learn two-component PL models (Zhao et al., 2016). Polynomial-time algorithms are developed to provably learn mixtures of two Mallows models via tensor decomposition (Awasthi et al., 2014) and learn mixtures with any constant number of components (Liu and Moitra, 2018). While prior works focus on learning from rankings such as voting data, our work studies learning mixtures of ranking models on text datasets and designs efficient algorithms for language models.

Several studies have designed ranking models beyond BT models in preference optimization. GPO (Zhang et al., 2025a) embeds responses into a latent space to capture intransitive pairwise rankings. Recent work has proposed learning a single multiway ranking model with surrogate losses to approximate ranking metrics (Zhao et al., 2025). For datasets with multiple evaluation criteria, prior works have trained regression models on evaluation scores (Wang et al., 2024b,a). EM-DPO (Chidambaram et al., 2025) learns mixtures of BT models from partial rankings. In contrast, our work focuses on learning mixtures of multiway ranking models. While another line of research optimizes a single language model across subpopulations of rankings via group distributional optimization (Yao et al., 2025), we learn a mixture of multiple ranking models.

Another line of work involves computing the influence of training data on model predictions. A commonly used technique is influence functions, shown effective for tracking the impact of training segments in sequence tagging (Jain et al., 2022). Prior work proposed function vectors to represent input-output functions using a set of attention heads in language models (Todd et al., 2024). To model task relationships, existing meth-

ods compute higher-order task affinity aggregated from multitask training outcomes across task subsets (Li et al., 2023a) or fit linear surrogate models to approximate these outcomes (Li et al., 2023b). Gradients have been used as features for efficiently computing model influence functions for modeling task relationships (Li et al., 2024a,b; Yang et al., 2025). Recent methods leverage these gradient features to group tasks and train weighted ensembles of low-rank adapters (Li et al., 2025b). Such gradient-based approaches have also been extended to model task relationships in multi-objective reinforcement learning (Zhang et al., 2025b) and multitask algorithmic reasoning (Li et al., 2025a).

There is a vast body of work on learning from a mixture of data distributions. For example, previous studies have applied kernel methods to learn predictors for a test distribution using labeled samples from several related distributions (Blanchard et al., 2011). In a multitask learning setting for predicting rewards across multiple arms in contextual bandits, kernel mean embeddings have been used to measure task similarities by characterizing each arm based on its contextual probability distribution (Deshmukh et al., 2017). While traditional semantic class learning and label propagation rely on graph structural linkages (Kozareva et al., 2011), our algorithm resolves structural unidentifiability in rank aggregation through generative ranking augmentation. Mixture proportion estimation has been studied to estimate the proportion in a mixture of clean and corrupted data, for training classifiers from noisy labels (Scott, 2015). Expectation-maximization algorithms are designed to jointly fit truncated multivariate Gaussian distributions to truncated and censored data (Lee and Scott, 2012). In comparison, our work designs an algorithm to learn a mixture of Plackett-Luce models from a text dataset with multiple preference criteria.

7 Conclusion

This paper investigates learning from datasets with multiple distinct ranking distributions. We identify the unidentifiability limit of mixture ranking models. To address this, we propose an efficient algorithm that resolves unidentifiability and scales to large models. Extensive evaluations demonstrate our method’s performance and efficiency advantages over existing approaches.

Limitations and Future Works

First, our algorithm treats generated responses ranked lower than existing responses in the dataset as an approximation. While empirical evidence supports this approximation, exceptions where generated responses rank higher can occur in practice. One alternative is to compute the probability of all possible rankings for new responses. While computing all possible cases is computationally expensive, designing an efficient algorithm to estimate these rankings is an interesting direction for future work. Second, while our current approach learns mixtures of ranking models with a fixed number of components, dynamically adapting this number in continual learning settings remains a promising open problem. Additionally, our method evaluates single-turn conversational data. Extending this approach to multi-turn conversations where user preferences dynamically evolve presents another valuable direction for future research. Our method provides a foundation to scale up the learning of ranking models for such extensions.

Potential Risks

We discuss a few potential risks in our work. Our work focuses on learning a mixture of Plackett-Luce models to identify clusters of rankings. Because the algorithm groups preferences into distinct clusters, there is a chance it might learn spurious correlations between the preferences and certain specific demographics. One future direction is to examine the misclassification errors in each group and design group distributionally robust optimization methods into the training.

Acknowledgment

We are grateful to Byron Wallace for several discussions related to this paper. Thanks to Zhen-shuo Zhang and Youran Ye for their feedback. The work of D. Li, Z. Zhang, and H. Zhang is in part supported by NSF award IIS-2412008.

AI assistants have been used to enhance the overall clarity and presentation of the work. Specifically, the tools were used for proofreading, identifying typographical or grammatical mistakes, and polishing writing. Furthermore, AI assistants have been used in formatting and verifying the mathematical notations and derivations of this paper.

References

- Amel Awadelkarim and Johan Ugander. 2024. Statistical models of top-k partial orders. In *SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
- Pranjal Awasthi, Avrim Blum, Or Sheffet, and Aravindan Vijayaraghavan. 2014. Learning mixtures of ranking models. *Advances in Neural Information Processing Systems (NeurIPS)*.
- Gilles Blanchard, Gyemin Lee, and Clayton Scott. 2011. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in Neural Information Processing Systems (NeurIPS)*.
- Louis Castricato, Nathan Lile, Rafael Rafailov, Jan-Philipp Fränken, and Chelsea Finn. 2025. Persona: A reproducible testbed for pluralistic alignment. In *International Conference on Computational Linguistics (COLING)*.
- Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Dinesh Manocha, Furong Huang, Amrit Singh Bedi, and Mengdi Wang. 2024. Maxmin-rlhf: alignment with diverse human preferences. In *ICML*.
- Keertana Chidambaram, Karthik Vinary Seetharaman, and Vasilis Syrgkanis. 2025. Direct preference optimization with unobserved preference heterogeneity: The necessity of ternary preferences. *AISTATS*.
- Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H Holliday, Bob M Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailley Schoelkopf, and William S. Zwicker. 2024. Position: social choice should guide ai alignment in dealing with diverse human feedback. In *International Conference on Machine Learning (ICML)*, pages 9346–9360.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: Boosting language models with high-quality feedback. In *International Conference on Machine Learning (ICML)*.
- Aniket Anand Deshmukh, Urun Dogan, and Clayton Scott. 2017. Multi-task learning for contextual bandits. *Advances in Neural Information Processing Systems (NeurIPS)*.
- Sarthak Jain, Varun Manjunatha, Byron C Wallace, and Ani Nenkova. 2022. Influence functions for sequence tagging models. In *Findings of EMNLP 2022*.
- Hannah Rose Kirk, Alexander Whitefield, Paul Rottger, Andrew M Bean, Katerina Margatina, Rafael Mosquera-Gomez, Juan Ciro, Max Bartolo, Adina Williams, He He, Bertie Vidgen, and Scott A Hale. 2024. The prism alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. *NeurIPS*.

- Zornitsa Kozareva, Konstantin Voevodski, and Shanghua Teng. 2011. Class label enhancement via related instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 118–128.
- Gyemin Lee and Clayton Scott. 2012. Em algorithms for multivariate gaussian mixture models with truncated and censored data. *Computational Statistics & Data Analysis*.
- Dongyue Li, Haotian Ju, Aneesh Sharma, and Hongyang R Zhang. 2023a. Boosting multitask learning on graphs through higher-order task affinities. In *SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
- Dongyue Li, Huy Nguyen, and Hongyang R. Zhang. 2023b. Identification of negative transfers in multitask learning using surrogate models. *Transactions on Machine Learning Research (TMLR)*.
- Dongyue Li, Aneesh Sharma, and Hongyang R. Zhang. 2024a. Scalable multitask learning using gradient-based estimation of task affinity. In *SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
- Dongyue Li, Zhenshuo Zhang, Minxuan Duan, Edgar Dobriban, and Hongyang R. Zhang. 2025a. Efficiently learning branching networks for multitask algorithmic reasoning. *arXiv preprint arXiv:2512.01113*.
- Dongyue Li, Ziniu Zhang, Lu Wang, and Hongyang R. Zhang. 2024b. Scalable fine-tuning from multiple data sources: A first-order approximation approach. In *Findings of EMNLP 2024*.
- Dongyue Li, Ziniu Zhang, Lu Wang, and Hongyang R. Zhang. 2025b. Efficient ensemble for fine-tuning language models on multiple datasets. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Allen Liu and Ankur Moitra. 2018. Efficiently learning mixtures of mallows models. In *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE.
- Rahul Makhijani and Johan Ugander. 2019. Parametric models for intransitivity in pairwise rankings. In *The World Wide Web Conference (WWW)*.
- 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. *Advances in Neural Information Processing Systems (NeurIPS)*.
- Clayton Scott. 2015. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Artificial Intelligence and Statistics (AISTATS)*. PMLR.
- Arjun Seshadri, Stephen Ragain, and Johan Ugander. 2020. Learning rich rankings. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9435–9446.
- Jingyan Shen, Jiarui Yao, Rui Yang, Yifan Sun, Feng Luo, Rui Pan, Tong Zhang, and Han Zhao. 2025. Micro: Mixture modeling and context-aware routing for personalized preference learning. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. 2024. Function vectors in large language models. In *International Conference on Learning Representations (ICLR)*, volume 2024.
- Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. 2024a. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. In *ACL*.
- Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024b. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Findings of EMNLP*.
- Fan Yang, Hongyang R. Zhang, Sen Wu, Christopher Ré, and Weijie J. Su. 2025. Precise high-dimensional asymptotics for quantifying heterogeneous transfers. *Journal of Machine Learning Research*, 26(113):1–88.
- Binwei Yao, Zefan Cai, Yun-Shiuan Chuang, Shanglin Yang, Ming Jiang, Diyi Yang, and Junjie Hu. 2025. No preference left behind: Group distributional preference optimization. In *International Conference on Learning Representations (ICLR)*.
- Yifan Zhang, Ge Zhang, Yue Wu, Kangping Xu, and Quanquan Gu. 2025a. Beyond bradley-terry models: A general preference model for language model alignment. In *International Conference on Machine Learning (ICML)*.
- Zhenshuo Zhang, Minxuan Duan, Youran Ye, and Hongyang R. Zhang. 2025b. Scalable multi-objective and meta reinforcement learning via gradient estimation. *arXiv preprint arXiv:2511.12779*.
- Ziniu Zhang, Zhenshuo Zhang, Dongyue Li, Lu Wang, Jennifer Dy, and Hongyang R. Zhang. 2025c. Linear-time demonstration selection for in-context learning via gradient estimation. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- Yang Zhao, Yixin Wang, and Mingzhang Yin. 2025. Permutative preference alignment from listwise ranking of human judgments. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- Zhibing Zhao, Peter Piech, and Lirong Xia. 2016. Learning mixtures of plackett-luce models. In *International Conference on Machine Learning (ICML)*. PMLR.

A Omitted Technical Materials

A.1 Notations

We describe the notation used throughout the paper. The dataset $\mathcal{D} = \{(x_i, Y_i, \sigma_i)\}_{i=1}^n$ consists of n ranking instances. Each instance pairs an input prompt x_i with a set $Y_i = \{y_1, \dots, y_m\}$ of m candidate responses and a ranking σ_i , a permutation of the candidate indices recording the order in which the responses are preferred. m' denotes the ranking length once the generated responses have been appended. a denotes how many responses are evaluated exactly as anchors for gradient-based estimation. $h(x, y)$ is the input embedding of response y under prompt x .

k is the number of clusters, each corresponding to one ranking distribution. z_i is the unobserved cluster assignment of instance i . The mixing weight α_c is the prior probability of cluster c , and $\gamma_{i,c}$ is the corresponding posterior, that is, the responsibility of cluster c for instance i . For a cluster c , the ranking probability is written as P_{θ_c} and is determined by a scalar score $r_{\theta_c}(x, y)$ assigned to each response. The score is given by a language model π_{θ_c} and a reference model π_{ref} , with β controlling the strength of the KL-divergence penalty between them.

A.2 Preliminaries

To precisely describe the concepts used in this paper, we first define the identifiability of a mixture of k Plackett-Luce models, following the terminology from [Zhao et al. \(2016\)](#).

A Plackett-Luce (PL) model defines the probability distribution over rankings of m alternatives. In our context, these alternatives are candidate responses $\{y_1, \dots, y_m\}$ to a prompt x . A ranking σ specifies the order $y_{\sigma(1)} \succ y_{\sigma(2)} \succ \dots \succ y_{\sigma(m)}$, where $\sigma(t)$ denotes the response index at rank t . Given a score vector r in a space $\{\theta | \theta_i \in [0, 1], \text{ for } i = 1, \dots, m, \text{ and } \sum_{i=1}^m \theta_i = 1\}$, with a score assigned to each alternative, the probability of observing ranking σ is:

$$P(\sigma|r) = \prod_{i=1}^{m-1} \frac{r_{\sigma(i)}}{\sum_{j=i}^m r_{\sigma(j)}}.$$

Recall that we compute these scores using a language model parameterized by θ , denoted as $r_{\theta}(x, y)$, using DPO. In this section, we focus on analyzing the identifiability of PL models in terms of the score vector r following [Zhao et al. \(2016\)](#).

We do not discuss the identifiability of the network parameters θ , while recognizing that distinct network weights could yield identical scores for the same input.

For a PL model with score vectors $r, v \in \mathbb{R}^m$, the model is strictly identifiable if $P(\cdot|r) = P(\cdot|v)$ implies $r = v$. For mixture models, we state the definition to account for label switching, where permuting the indices of the mixture components does not change the marginal distribution.

Definition A.1 (Identifiability of k -PL). A mixture of k Plackett-Luce models (k -PL) over m alternatives is identifiable if there do not exist two distinct mixture formulations that produce the same distribution over all rankings. Specifically, the model is identifiable if there do not exist:

1. Two integers $k_1, k_2 \leq k$,
2. A set of pairwise different score vectors $r^{(1)}, \dots, r^{(k_1)}$ and $v^{(1)}, \dots, v^{(k_2)}$ (no two vectors are exactly the same),
3. Strictly positive mixing weights $\alpha^{(1)} = (\alpha_1^{(1)}, \dots, \alpha_{k_1}^{(1)})$ and $\alpha^{(2)} = (\alpha_1^{(2)}, \dots, \alpha_{k_2}^{(2)})$ where $\sum_{r=1}^{k_1} \alpha_r^{(1)} = 1$ and $\sum_{r=1}^{k_2} \alpha_r^{(2)} = 1$,

such that for every possible ranking σ , the following equality holds:

$$\sum_{c=1}^{k_1} \alpha_c^{(1)} P(\sigma|r^{(c)}) = \sum_{c=1}^{k_2} \alpha_c^{(2)} P(\sigma|v^{(c)}). \quad (8)$$

To prove the identifiability of k -PL models, we analyze the rank of a probability matrix involving the probabilities over $m!$ possible rankings over m alternatives. For a PL model with a score vector r , we represent a distribution over all $m!$ rankings as an $m! \times 1$ column vector, denoted as $f_m(r)$, where each entry is the probability of a specific ranking σ . For example, when $m = 3$, we have:

$$f_3(r) = \begin{pmatrix} \Pr(y_1 \succ y_2 \succ y_3|r) \\ \Pr(y_1 \succ y_3 \succ y_2|r) \\ \Pr(y_2 \succ y_1 \succ y_3|r) \\ \Pr(y_2 \succ y_3 \succ y_1|r) \\ \Pr(y_3 \succ y_1 \succ y_2|r) \\ \Pr(y_3 \succ y_2 \succ y_1|r) \end{pmatrix} = \begin{pmatrix} \frac{r_1 r_2}{1-r_1} \\ \frac{r_1 r_3}{1-r_1} \\ \frac{r_1 r_2}{1-r_2} \\ \frac{r_2 r_3}{1-r_2} \\ \frac{r_1 r_3}{1-r_3} \\ \frac{r_2 r_3}{1-r_3} \end{pmatrix}.$$

Then, we define a probability matrix F as an $m! \times 2k$ matrix formed by concatenating the

column vectors of these $2k$ PL models. Given $r^{(1)}, \dots, r^{(2k)}$, we define

$$F = [f_m(r^{(1)}), f_m(r^{(2)}), \dots, f_m(r^{(2k)})]. \quad (9)$$

The entry $F_{\sigma,c}$ corresponds to the probability of a ranking σ in the c -th PL model. Note that F is a function of the $2k$ score vectors.

We prove the identifiability or non-identifiability of k -PL by analyzing the rank of F . If the rank of F is $2k$ for all pairwise different vectors $(r^{(1)}, \dots, r^{(2k)})$, then the k -PL model is identifiable. Conversely, if the rank of F is less than $2k$, the k -PL model is non-identifiable.

A.3 Proof for Proposition 3.1

Now we present the proof of Proposition 3.1 for the non-identifiability of mixture ranking models. The key idea is to show that the row space of the probability matrix F can exhibit a reduced dimensionality under a construction. Under this construction, the row space of the probabilities $m!$ possible rankings can be represented by m row vectors, thus proving that the rank F is less than $2k$. We then prove that this dimensionality collapse leads to two k -PL models that yield the same distributions.

Proof of Proposition 3.1. Our proof is based on analyzing the $m! \times 2k$ probability matrix F . To demonstrate unidentifiability, we aim to prove that a non-zero coefficient vector $\beta \in \mathbb{R}^{2k}$ exists such that $F\beta = 0$.

To prove the unidentifiability, we construct a specific symmetric structure for $r^{(c)}$, and show that under such a construction, the mixture of the ranking models is unidentifiable. We construct a combined pool of $2k$ distinct latent clusters. Let $r^{(c)}$ denote the score vector for cluster $c \in \{1, \dots, 2k\}$. For a given set of m alternatives $Y = \{y_1, y_2, \dots, y_m\}$, we assign the following scores under each cluster c :

- We assign a distinct score $e_c \in (0, 1)$ to alternative y_1 . We ensure $e_i \neq e_j$ for all $i \neq j$.
- We assign an identical score $b_c = \frac{1-e_c}{m-1}$ to all remaining alternatives $y_2, \dots, y_m$.

Now, we evaluate the entries of $F_{\sigma,c}$ under this specific construction. Let $t \in \{1, \dots, m\}$ denote the rank position of alternative y_1 in the permutation σ . In the Plackett-Luce model, the probability of selecting the alternatives in the order specified

by σ is computed as a sequence of $m-1$ choices. This sequential product simplifies by grouping the choices into three cases:

- First, before rank t (steps 1 to $t-1$), an alternative with score b_c is chosen. Assuming the scores sum to 1 ($e_c + (m-1)b_c = 1$), the remaining sum of scores at step j is $1 - (j-1)b_c$. The probability for these steps is $\prod_{j=1}^{t-1} \frac{b_c}{1-(j-1)b_c}$.
- Second, at rank t , alternative y_1 with score e_c is chosen. The remaining sum of scores is $1 - (t-1)b_c$. The probability is $\frac{e_c}{1-(t-1)b_c}$.
- Third, after rank t (steps $t+1$ to m), all remaining $m-t$ alternatives have the identical score b_c . By symmetry, the probability of any specific ordering of these remaining identical alternatives is simply $\frac{1}{(m-t)!}$.

Therefore, the likelihood entry simplifies to a function that depends strictly on y_1 's rank position t and the cluster c . Let us denote this unique probability value as $v_{t,c}$:

$$v_{t,c} = \frac{1}{(m-t)!} \frac{e_c b_c^{t-1}}{\prod_{j=0}^{t-1} (1 - j b_c)}. \quad (10)$$

Because the probability $F_{\sigma,c}$ evaluates exactly to $v_{t,c}$ for any ranking σ where y_1 is at rank t , the $m!$ rows of the likelihood matrix F consist of only m unique types of rows. We can formally express this by factorizing the $m! \times 2k$ matrix F into the product of two smaller matrices, M and V :

$$F = MV, \quad (11)$$

where we explicitly construct M and V as follows:

- V is an $m \times 2k$ matrix capturing the unique probability values across all clusters. Its (t, c) -th entry is exactly $V_{t,c} = v_{t,c}$, representing the sequence probability in cluster c when y_1 is chosen at step t .
- M is an $m! \times m$ binary indicator matrix. Its rows represent the $m!$ possible rankings σ , and its columns represent the m possible rank positions t . The entry $M_{\sigma,t} = 1$ if alternative y_1 is exactly at position t in ranking σ , and 0 otherwise.

Since y_1 can occupy exactly one position in any given ranking, each row of M contains exactly one

1 and $m - 1$ zeros. Multiplying row σ of M by column c of V simply selects the correct value $v_{t,c}$ to form $F_{\sigma,c}$.

Since the inner dimension of both matrices is m , we establish the exact dimensionality bound for F :

$$\text{rank}(F) = \min\{m, 2k\}.$$

Thus, the dimension of the null space of F is larger than 1, under the proposition's condition $m \leq 2k - 1$. Therefore, the null space of F is strictly non-trivial, guaranteeing the existence of a non-zero vector β such that $F\beta = \mathbf{0}$.

Furthermore, because the rows of F represent probabilities over an exhaustive sample space, $\sum_{\sigma} F_{\sigma,c} = 1$ for all c . Summing the rows of $F\beta = \mathbf{0}$ yields:

$$\begin{aligned} \sum_{\sigma} \sum_{c=1}^{2k} F_{\sigma,c} \beta_c &= 0 \\ &= \sum_{c=1}^{2k} \beta_c \left(\sum_{\sigma} F_{\sigma,c} \right) = \sum_{c=1}^{2k} \beta_c = 0. \end{aligned}$$

Since β is a non-zero vector whose elements sum to zero, it must contain both strictly positive and strictly negative elements. Let $I_+ = \{c \mid \beta_c > 0\}$ and $I_- = \{c \mid \beta_c < 0\}$. We define a normalization constant $W = \sum_{c \in I_+} \beta_c = \sum_{c \in I_-} -\beta_c$.

We now construct the two mixtures. We define two disjoint subsets of cluster parameters $r = \{r^{(c)}\}_{c \in I_+}$ and $r' = \{r^{(c)}\}_{c \in I_-}$. We construct their corresponding mixing weights as $\alpha_c = \frac{\beta_c}{W}$ for $c \in I_+$, and $\alpha'_c = \frac{-\beta_c}{W}$ for $c \in I_-$.

Because $F\beta = \mathbf{0}$, we have:

$$\sum_{c \in I_+} \alpha_c F_{\sigma,c} = \sum_{c \in I_-} \alpha'_c F_{\sigma,c} \quad \text{for all } \sigma.$$

This confirms that the mixture model defined by (r, α) and the disjoint mixture model defined by (r', α') produce the same marginal probability distribution over the m -way rankings, proving the model is non-identifiable.

A.4 Proof for Proposition 4.1

Next, we present the proof for analyzing the rank of the probability matrix of partial rankings over the augmented set of responses. We assume that the generated responses exhibit distinct scores. We show that, under this assumption, a probability

matrix (representing all possible ranking probabilities under the mixture model) has sufficiently many independent rows to match its column dimension, thus exhibiting a full column rank.

Proof of Proposition 4.1. Recall that we augment the original m responses with additional generated responses and expand the candidate set to Y' with a size of m' . In this proof, we examine the probability over the space of partial rankings over m' alternatives, formed by selecting a ranked list of m items and leaving the remaining $m' - m$ items unranked at the bottom. The total number of partial rankings in the expanded space is $\frac{m'!}{(m' - m)!}$.

For an augmented set of m' alternatives, we denote such a partial ranking σ as $y_{\sigma(1)} \succ y_{\sigma(2)} \succ \dots \succ y_{\sigma(m)} \succ \{y_{\sigma(m+1)}, \dots, y_{\sigma(m')}\}$. Given a score vector r of dimension m' , the probability of observing the partial ranking σ in a PL model is:

$$P(\sigma|r) = \prod_{i=1}^{m-1} \frac{r_{\sigma(i)}}{\sum_{j=i}^m r_{\sigma(j)} + Z(\sigma)},$$

where $Z(\sigma) = \sum_{t=m+1}^{m'} r_{\sigma(t)}$ represents the sum of scores of alternatives ranked at the bottom.

Given $2k$ score vectors $r^{(1)}, \dots, r^{(2k)}$, we then define the probability matrix over the augmented set of alternatives as F^{aug} , which is a $\frac{m'!}{(m' - m)!} \times 2k$ matrix, where each row corresponds to one of the possible partial rankings and each column corresponds to one of the $2k$ PL components.

In the conditions of this proposition, we assume that the newly generated augmented responses exhibit distinct scores within each PL model. For any distinct additional alternatives $i, j \in \{m + 1, \dots, m'\}$, we assume $r_i^{(c)} \neq r_j^{(c)}$ for each PL model c . As these scores are distinct, the denominator term $Z(\sigma)$ depends on the specific combinatorial subset of items left unselected in the bottom σ . Consequently, for distinct partial rankings $\sigma_1 \neq \sigma_2$ that leave different subsets of items unranked, the denominators differ: $Z(\sigma_1) \neq Z(\sigma_2)$. This makes it impossible to factor F^{aug} into the rank-deficient form.

We now show the identifiability of F^{aug} by examining its rank. We will show that for all possible pairwise different vectors, the rank of F^{aug} is equal to $2k$. We first obtain a $2k \times 2k$ matrix $\hat{F} = T \times F^{\text{aug}}$ by linearly combining some row vectors of F^{aug} via a linear transformation T . Then, we show that $\text{rank}(\hat{F}) = 2k$, which implies that $\text{rank}(F^{\text{aug}}) = 2k$.

Under the conditions of the proposition, we have at least k alternatives in the additional augmented set ($m' - m \geq k$). Without loss of generality, we select exactly k items from the augmented set, denoted as $y_1, y_2, \dots, y_k$. Let $r_j^{(c)}$ denote the score of item y_j in the c -th Plackett-Luce model, which is also the marginal probability that item y_j is ranked at the top. Denote a row vector $\omega^{(j)} = [r_j^{(1)}, r_j^{(2)}, \dots, r_j^{(2k)}]$ as these marginal probabilities across all $2k$ components.

We obtain the $2k \times 2k$ matrix $\hat{F}$ by summing over the corresponding rows in F^{aug} to obtain the following probabilities. Here, the matrix T is a 0-1 matrix indicating the selection of rows:

- The sum of all rows in F^{aug} , yielding the vector $\mathbf{1} = [1, 1, \dots, 1]$.
- The sum of all rows that rank y_j at the top for $j \in \{2, \dots, k\}$, yielding the row vector $\omega^{(j)}$.
- The sum of all rows that rank y_j at the top and y_1 as the second, for $j \in \{2, \dots, k\}$. The c -th element of this row vector is $\frac{r_j^{(c)} r_1^{(c)}}{1 - r_j^{(c)}}$.

Then, the matrix $\hat{F}$ takes the following form:

$$\hat{F} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ r_1^{(1)} & r_1^{(2)} & \dots & r_1^{(2k)} \\ r_2^{(1)} & r_2^{(2)} & \dots & r_2^{(2k)} \\ \vdots & \vdots & \ddots & \vdots \\ r_k^{(1)} & r_k^{(2)} & \dots & r_k^{(2k)} \\ \frac{r_2^{(1)} r_1^{(1)}}{1 - r_2^{(1)}} & \frac{r_2^{(2)} r_1^{(2)}}{1 - r_2^{(2)}} & \dots & \frac{r_2^{(2k)} r_1^{(2k)}}{1 - r_2^{(2k)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{r_k^{(1)} r_1^{(1)}}{1 - r_k^{(1)}} & \frac{r_k^{(2)} r_1^{(2)}}{1 - r_k^{(2)}} & \dots & \frac{r_k^{(2k)} r_1^{(2k)}}{1 - r_k^{(2k)}} \end{bmatrix}.$$

Because the parameters are non-degenerate, at least one vector in $\{\omega^{(1)}, \dots, \omega^{(k)}\}$ is linearly independent of $\mathbf{1}$. Without loss of generality, suppose $\omega^{(1)}$ is linearly independent of $\mathbf{1}$.

Next, we prove the proposition by examining the following two cases.

Case 1. $\omega^{(2)}, \dots, \omega^{(k)}$ are all linear combinations of $\mathbf{1}$ and $\omega^{(1)}$. For all $j \in \{2, \dots, k\}$, we can rewrite $r_j^{(c)} = p_j r_1^{(c)} + q_j$ for some constants p_j, q_j . In this case, the entries of $\hat{F}$ are rational functions of $r_1^{(c)}$. Because the scores of the augmented items are distinct, $r_1^{(c)}$ takes on pairwise different values across the $2k$ components.

Then, we prove that the rank of F^{aug} is equal to $2k$ by contradiction. Suppose that $\text{rank}(\hat{F}) < 2k$. Then, there exists a non-zero vector $t = [t_1, \dots, t_{2k}]^\top$ spanning the null space, such that $t^\top \hat{F} = \mathbf{0}$. For the dot product of t with the c -th column of $\hat{F}$ yields:

$$t_1 + t_2(r_1^{(c)}) + \sum_{j=2}^k t_{j+1}(r_j^{(c)}) + \sum_{j=2}^k t_{k+j} \left(\frac{r_j^{(c)} r_1^{(c)}}{1 - r_j^{(c)}} \right) = 0.$$

By substituting $r_j^{(c)} = p_j r_1^{(c)} + q_j$ into the above equation and letting the variable $x = r_1^{(c)}$, we can define a function $f(x)$ such that $f(x) = 0$ for every component c .

Then, we can convert this function into a polynomial $g(x)$ by the product of all its denominators. The highest degree terms in $g(x)$ have a maximum possible degree of k . Thus, $g(x)$ is a polynomial of degree at most k . Because $g(x) = 0$ must hold across $2k$ PL-models, the polynomial x must have $2k$ distinct roots. However, a non-zero polynomial of degree k can have at most k roots. Because $k < 2k$, this creates a contradiction. Therefore, $\text{rank}(\hat{F})$ must equal $2k$.

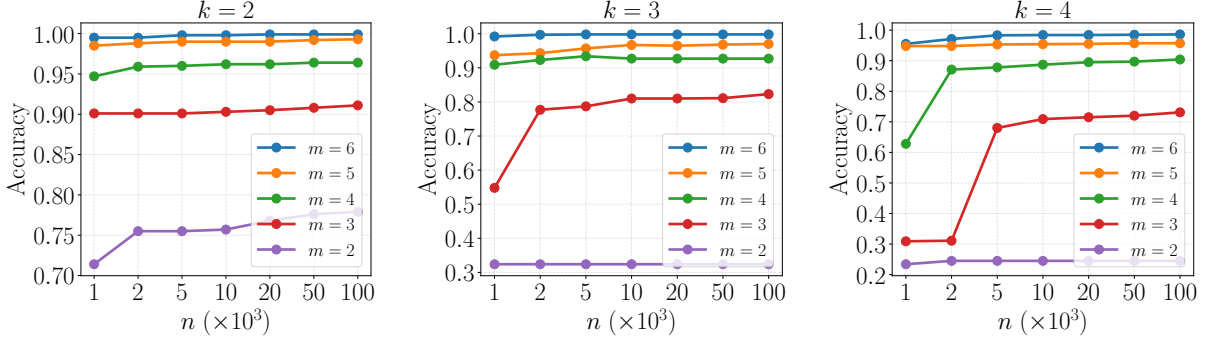
Case 2. There exists an $\omega^{(i)}$ (where $i \in \{2, \dots, k\}$) that is linearly independent of $\mathbf{1}$ and $\omega^{(1)}$. Because $\omega^{(i)}$ cannot be expressed as a linear combination of $\mathbf{1}$ and $\omega^{(1)}$, inserting it into $\hat{F}$ introduces an additional orthogonal row. Following the similar logic of Case 1, we can prove that $\text{rank}(\hat{F})$ equals $2k$.

Since the rank of $\hat{F}$ is equal to $2k$ and $\hat{F} = T F^{\text{aug}}$, thus the rank of F^{aug} is larger than $2k$. As the column dimension of F^{aug} is $2k$, thus, we have the rank of F^{aug} is $2k$.

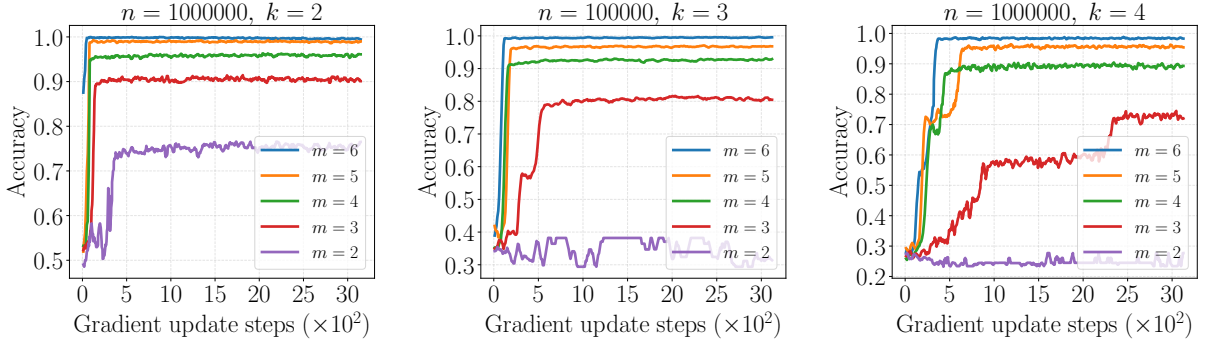
A.5 Extensions

We present our method using the Mallows model that defines the probability of a ranking based on its distance from an ideal ranking σ_c . Let $d_c(\cdot, \cdot)$ be a distance metric and $\phi_c(x_i) \in (0, 1)$ be a dispersion parameter dictating annotator variance. The probability of the ranking is:

$$\frac{\phi_c(x_i)^{d_c(\sigma_i, \sigma_c)}}{\prod_{j=1}^m \sum_{l=0}^{j-1} \phi_c(x_i)^l}. \quad (12)$$



(a) Varying the number of training rankings n between 1,000 and 100,000. Using rankings of $m = 2$ cannot recover the clusters, even for a large n .



(b) Varying the ranking size m between 2, 3, 4, 5 and 6, at $n = 100,000$ rankings. Larger m converges to higher accuracy.

Figure 6: Validation accuracy of the inferred cluster assignments on the synthetic data, for $k = 2, 3, 4$ clusters from left to right in each row. The two rows vary the two quantities that govern recovery: the amount of data in **a**, and the length of each ranking in **b**.

We use the Reverse Major Index (RMJ) as the distance metric, which penalizes adjacent inversions, weighted by their rank position, where $d_c(\sigma_i, \sigma_c) = \sum_{t=1}^{m-1} (m-t) \cdot \mathbb{1}\{r_{\theta_c}(x_i, y_{i, \sigma_i(t)}) < r_{\theta_c}(x_i, y_{i, \sigma_i(t+1)})\}$. The $(m-t)$ weighting imposes a larger penalty for misrankings at the top of the list. Substituting this probability into Equation (1) forms the Mixture of Mallows models.

B Omitted Experiments

B.1 Synthetic data

We now describe the details of the synthetic experiment to show the unidentifiability of the mixture of ranking models. We construct the dataset as follows: each response y is represented as a list of k integers, where k is the total number of clusters. The ground-truth utility of y for an annotator in cluster $c \in \{1, \dots, k\}$ is the c -th integer in the list. In other words, cluster c ranks responses by its c -th integer. For example, consider three responses: “[1, 2, 3]”, “[2, 3, 1]”, “[3, 1, 2]”, the rankings from clusters 1 and 2 are given as:

Cluster 1: [3, 1, 2] $\succ$ [2, 3, 1] $\succ$ [1, 2, 3]

Cluster 2: [2, 3, 1] $\succ$ [1, 2, 3] $\succ$ [3, 1, 2]

We generate a dataset with n annotators and k clusters. We conduct experiments in three settings with k between 2, 3, and 4. We vary the slate size $m \in \{2, 3, 4, 5, 6, 7, 8\}$. We train a Qwen-3-0.6B using the EM algorithm and evaluate the cluster assignment accuracy on a held-out validation set.

We show the validation accuracy of the cluster assignments when applying the algorithm to identify the underlying clusters with a ranking of different sizes m . In Figure 6a, we vary the number of rankings for training n between 1,000 and 100,000. We found that when using ranking over $m = 2$ items, the algorithm cannot fully recover the clusters, even when increasing the rankings n . For three (or four) underlying clusters, using $m = 2$ achieves nearly the accuracy of random assignment (i.e., $1/k$). In contrast, a larger value of m leads to a higher accuracy.

We further show the convergence of the algorithm in Figure 6b. In addition, when comparing the number of samples n across settings with

Table 5: Relative approximation error of $\epsilon_{x,y}$, tested with five language models of up to 34 billion parameters on the HelpSteer2 dataset.

Distance	Qwen-0.6B	Gemma-2B	DeepSeek-7B	Llama-13B	CodeLlama-34B
0% – 5%	$0.1 \pm 0.0\%$	$0.1 \pm 0.0\%$	$0.1 \pm 0.0\%$	$0.1 \pm 0.0\%$	$0.1 \pm 0.0\%$
5% – 10%	$0.7 \pm 0.3\%$	$0.6 \pm 0.3\%$	$0.6 \pm 0.0\%$	$0.5 \pm 0.2\%$	$0.5 \pm 0.0\%$
10% – 15%	$1.6 \pm 0.1\%$	$1.5 \pm 0.2\%$	$1.7 \pm 0.1\%$	$1.6 \pm 0.3\%$	$1.5 \pm 0.4\%$
15% – 20%	$3.0 \pm 0.3\%$	$3.1 \pm 0.1\%$	$3.4 \pm 0.3\%$	$3.0 \pm 0.5\%$	$3.1 \pm 0.9\%$
20% – 25%	$5.4 \pm 0.4\%$	$5.4 \pm 0.1\%$	$5.0 \pm 0.3\%$	$5.1 \pm 0.7\%$	$5.3 \pm 0.2\%$

different numbers of clusters, we found that it requires more training samples with a large number of underlying clusters.

B.2 Evaluation of approximation errors

Table 5 shows the relative approximation error of the gradient estimation method across five different language models when tested on the HelpSteer2 dataset. The models evaluated range from 0.6 billion up to 34 billion parameters, specifically including Qwen-0.6B, Gemma-2B, DeepSeek-7B, Llama-13B, and CodeLlama-34B. The error remains low across all model sizes, starting at 0.1% for the closest distances and remaining between 5.0% and 5.4% for the largest distance bucket.

B.3 Comparing fine-tuned model generation

We evaluate the generation of our fine-tuned models on the UltraFeedback dataset. We compute a win rate against the given response ranked at the top, given a reference response in the dataset. For each test prompt, we generate responses from both models and score them using a reward model trained solely on the ground-truth rankings of that corresponding cluster. The win rate is the percentage of times the fine-tuned model’s response achieves a higher reward score than the baseline’s response.

We find that our method achieves an average win rate of 58%, outperforming baselines using single ranking models by 7% and mixture-of-BT baselines by 4.3% across all evaluated clusters. This indicates that our method aligns the language model’s generations with the distinct ranking distributions of each cluster. In contrast, because pairwise mixture models struggle to accurately identify the clusters, they fail to tailor their generations to the specific target cluster.

B.4 Additional computation and parameters

Our algorithm generates additional responses prior to training, accounting for only a small fraction of

the total runtime. Evaluated on a single A6000 GPU, response generation takes 1.1 GPU hours for the UltraFeedback dataset and 1.3 GPU hours for the Persona dataset, whereas model training requires 7.3 and 7.2 GPU hours, respectively. Overall, response generation consumes roughly 16% of the total runtime.

We use LoRA adapters to implement the mixture of PL models. Each PL model corresponds to a distinct set of LoRA adapters. Thus, a mixture of k models requires k sets of adapters on top of the base model. In our experiments, setting $k = 4$ for the UltraFeedback dataset introduces 13.5% additional parameters relative to the base model. For the Persona dataset, setting $k = 12$ results in 40.5% additional parameters.

B.5 Complete comparison

We provide the complete results corresponding to the experiments and ablation studies discussed in the main text. Table 6 presents the clustering and ranking accuracies broken down by individual evaluation dimensions and their averages.

Table 7 reports the clustering and ranking accuracies across all twelve demographic profiles on the Persona dataset.

Table 8 reports the impact of varying the response generation sampling temperature on ranking accuracy across the evaluation criteria.

Table 9 reports the results of our hyperparameter ablation study of varying the number of clusters, the augmented ranking size, and the number of anchors.

B.6 Discussions

Datasets, model sizes, and compute cost. We evaluate our approach across several datasets that exhibit multi-dimensional evaluations and diverse user demographics. The details of the datasets, including their categories, ranking properties, and sources, are summarized in Table 10.

Table 6: We report the clustering accuracies (%) on the UltraFeedback dataset for each evaluation dimension and their average. Our approach is compared against baselines utilizing a single ranking model and mixtures of BT models. During evaluation, we infer the cluster membership of each test instance via the learned mixture model and apply the corresponding cluster-specific head to predict the final ranking. All results report the mean and variance over three random seeds.

Clustering accuracy (%)	Helpfulness	Honesty	Instruction following	Truthfulness	Average
MiCRo	33.4 ± 7.8	28.7 ± 4.5	22.8 ± 0.3	21.7 ± 5.9	26.6 ± 2.1
MaxMin-RLHF	27.5 ± 4.3	14.4 ± 2.8	35.3 ± 5.6	27.1 ± 0.5	27.2 ± 0.4
EM-DPO	25.1 ± 1.9	22.7 ± 2.7	33.2 ± 5.4	23.2 ± 2.9	26.9 ± 2.1
MoPLEX (Alg. 2)	68.4 ± 2.3	73.8 ± 0.4	68.4 ± 1.3	72.9 ± 3.1	70.9 ± 2.3

Table 7: We report the clustering and ranking accuracies (%) on the Persona dataset across each cluster. Our approach is compared against baselines utilizing a single ranking model and mixtures of BT models. During evaluation, we infer the cluster membership of each test instance via the learned mixture model and apply the corresponding cluster-specific head to predict the final ranking. Ranking accuracy is measured as the proportion of correctly predicted pairs over all pairs. All results report the mean and variance over three random seeds.

	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
Clustering accuracy (%)						
MiCRo	11.7 ± 0.6	10.5 ± 0.3	21.7 ± 0.4	16.6 ± 1.7	15.0 ± 0.0	47.8 ± 1.3
MaxMin-RLHF	35.2 ± 1.9	15.7 ± 0.9	8.0 ± 0.7	8.3 ± 0.3	2.0 ± 0.0	30.4 ± 1.3
EM-DPO	58.8 ± 0.2	26.3 ± 1.2	8.7 ± 0.0	8.3 ± 0.3	25.0 ± 0.0	39.1 ± 1.3
MoPLEX (Alg. 2)	61.1 ± 1.1	60.0 ± 0.4	58.3 ± 0.3	50.0 ± 0.0	60.0 ± 0.5	58.3 ± 1.3
Ranking accuracy (%)						
DPO	52.9 ± 2.4	53.7 ± 0.8	53.3 ± 0.6	57.1 ± 0.4	54.2 ± 3.0	53.3 ± 1.6
LiPO	44.4 ± 0.4	60.0 ± 0.0	54.1 ± 0.6	58.3 ± 1.3	65.0 ± 0.1	58.3 ± 0.3
MiCRo	64.7 ± 1.1	52.6 ± 0.3	65.2 ± 1.2	50.0 ± 0.0	30.0 ± 0.0	56.5 ± 1.2
MaxMin-RLHF	52.9 ± 0.4	63.1 ± 1.6	56.5 ± 1.2	58.3 ± 1.3	35.0 ± 1.0	56.5 ± 2.0
EM-DPO	70.5 ± 0.9	52.6 ± 1.3	39.1 ± 2.3	58.3 ± 2.3	40.0 ± 2.0	56.5 ± 2.0
MoPLEX (Alg. 2)	83.3 ± 1.3	83.3 ± 2.3	80.0 ± 2.0	58.3 ± 0.3	80.0 ± 1.0	70.8 ± 0.3
	Persona 7	Persona 8	Persona 9	Persona 10	Persona 11	Persona 12
Clustering accuracy (%)						
MiCRo	33.3 ± 0.2	14.2 ± 0.9	8.3 ± 0.3	25.0 ± 0.0	13.6 ± 0.4	16.6 ± 1.7
MaxMin-RLHF	11.1 ± 0.1	9.5 ± 0.2	20.8 ± 0.3	10.0 ± 0.0	22.7 ± 0.3	33.3 ± 1.3
EM-DPO	14.2 ± 0.6	2.4 ± 0.4	29.1 ± 0.7	20.0 ± 0.0	13.6 ± 0.4	27.7 ± 1.8
MoPLEX (Alg. 2)	50.0 ± 0.0	54.5 ± 0.4	57.6 ± 1.9	63.6 ± 0.3	66.6 ± 1.7	61.1 ± 1.1
Ranking accuracy (%)						
DPO	53.3 ± 3.6	52.9 ± 0.4	54.6 ± 1.2	54.6 ± 0.2	56.3 ± 2.0	57.5 ± 0.6
LiPO	55.5 ± 0.5	50.0 ± 0.7	42.3 ± 0.0	45.4 ± 0.5	66.6 ± 1.6	77.7 ± 0.7
MiCRo	38.8 ± 0.9	71.4 ± 1.3	62.5 ± 2.0	35.0 ± 0.6	45.4 ± 0.5	50.0 ± 2.0
MaxMin-RLHF	50.0 ± 1.0	52.3 ± 1.8	58.3 ± 0.3	40.0 ± 2.0	36.3 ± 1.6	38.8 ± 0.9
EM-DPO	61.1 ± 1.1	42.8 ± 1.6	45.8 ± 1.3	35.0 ± 0.5	36.3 ± 1.6	50.0 ± 2.0
MoPLEX (Alg. 2)	100.0 ± 0.0	63.6 ± 1.3	80.7 ± 1.6	63.6 ± 1.3	79.1 ± 1.6	66.6 ± 1.6

Table 8: We report the ablation study results of varying the sampling temperature during response generation on the UltraFeedback dataset. We compare the ranking accuracy across four criteria and their average across temperatures of 0.5, 1.0, 2.0, and 4.0. Ranking accuracy is measured as the proportion of correctly predicted pairs out of the six possible pairs per four-response ranking. All results report the mean and variance over three random seeds.

Sampling Temperature	Helpfulness	Honesty	Instruction-following	Truthfulness	Average
0.5	76.6 ± 2.5	69.7 ± 1.6	71.5 ± 1.7	75.8 ± 1.8	70.9 ± 1.9
1.0	77.2 ± 1.8	72.8 ± 2.9	73.2 ± 2.5	68.9 ± 1.3	72.8 ± 2.1
2.0	79.4 ± 3.9	73.0 ± 1.3	76.9 ± 2.6	70.5 ± 1.3	75.0 ± 2.9
4.0	79.5 ± 1.9	72.9 ± 2.1	76.0 ± 2.7	71.7 ± 1.8	74.7 ± 2.3

Table 9: We report the average ranking accuracy on the Persona dataset across 12 demographic profiles when varying key hyperparameters: the number of clusters (k), the augmented ranking size (m'), and the number of anchors (a). We vary one parameter at a time while keeping the others fixed at their default values ($k = 12, m' = 26, a = 6$). All results report the mean and variance over three random seeds.

Varying Number of Clusters (Fixed: $m' = 26, a = 6$)				
Parameter	$k = 4$	$k = 8$	$k = 12$	$k = 16$
Accuracy	65.2 ± 1.6	69.6 ± 0.9	76.4 ± 0.6	76.7 ± 0.4
Varying Augmented Ranking Size (Fixed: $k = 12, a = 6$)				
Parameter	$m' = 12$	$m' = 20$	$m' = 26$	$m' = 32$
Accuracy	59.6 ± 2.6	65.1 ± 1.2	76.4 ± 0.6	76.8 ± 0.8
Varying Number of Anchors (Fixed: $k = 12, m' = 26$)				
Parameter	$a = 2$	$a = 4$	$a = 6$	$a = 8$
Accuracy	63.8 ± 1.6	72.4 ± 1.8	76.4 ± 0.6	76.4 ± 0.9

Table 10: Datasets used in our experiments. Recall that m is the number of options and k is the number of groups.

Dataset	m	k	Sources
UltraFeedback	4	4	openbmb/UltraFeedback
Persona	2	12	SynthLabsAI/PERSONA
HelpSteer2	2	5	nvidia/HelpSteer2

Table 11: Base language models used in our experiments.

Models	Sources
Qwen-3-0.6B	Qwen/Qwen3-0.6B
Gemma-2B	google/gemma-2-2b-it
DeepSeek-7B	deepseek-ai/deepseek-llm-7b-base
Llama-13B	meta-llama/Llama-2-13b-chat-hf
CodeLlama-34B	meta-llama/CodeLlama-34b-Instruct-hf

We conduct experiments using a range of base language models with varying parameter sizes to validate our gradient estimation approach. The models and their parameter counts are detailed in Table 10.

The total computational cost is described in Section 5 of this paper. We evaluate the runtime and memory cost on a machine with a single RTX-A6000 GPU and 48 CPUs.