MITIGATING REWARD HACKING IN LLM-BASED RECOMMENDATION: A PREFERENCE OPTIMIZATION APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training adaptation has become the central paradigm for leveraging large language models (LLMs) in recommendation. While recent preference optimization methods, such as Direct Preference Optimization (DPO), enhance pairwise preference discrimination, they remain vulnerable to *reward hacking*: models exploit imperfections in reward signals, leading to inflated training metrics without genuine recommendation gains. We provide a theoretical analysis of this phenomenon from a gradient perspective and formalize the concept of the ε-*insensitive region*, where pairwise updates exert negligible influence on the relative ordering between positives and unsampled negatives. We further show under the Bradley–Terry model that such regions can occupy a substantial portion of the preference distribution, inevitably causing misaligned ranking. To address this issue, we propose **Si**mulated Preference Optimization for **R**eward-hacking m**i**tigation using Pseudo-negatives (**SIRIUS**). Our framework introduces pseudo-negative samples to enrich contrastive signals and reduce the prevalence of ε-insensitive regions. Extensive experiments on three public benchmarks—LastFM, Goodreads, and Steam—demonstrate that SIRIUS consistently improves ranking quality and effectively mitigates reward hacking, providing both theoretical and practical insights for advancing LLM-based recommendation. Our code is available at https://anonymous.4open.science/r/C557-id

1 INTRODUCTION

Post-training adaptation has become the dominant paradigm for modeling user preferences in LLM-based recommendation. Early methods—prompting (Gao et al., 2023b; Geng et al., 2022; Dai et al., 2023) and supervised fine-tuning (SFT) (Bao et al., 2023b; Liao et al., 2024b; Bao et al., 2023a; Lin et al., 2024; Zhang et al., 2023)—cast user histories and item descriptions as text for next-token prediction; despite strong results, they lack explicit negative modeling, limiting fine-grained preference contrasts. Preference optimization addresses this with pairwise objectives: Direct Preference Optimization (DPO) (Rafailov et al., 2023), a lightweight alternative to RLHF (Christiano et al., 2017), is analogous to Bayesian Personalized Ranking (BPR) (Rendle et al., 2012), and recent variants tailored to LLM recommenders report improved discrimination and ranking (Chen et al., 2024b; Liao et al., 2024a; Gao et al., 2024).

However, preference optimization is vulnerable to **reward hacking**: models exploit imperfections in reward signals to inflate training metrics without genuinely improving alignment (Amodei et al., 2016). While related effects are documented in LLM alignment (Rafailov et al., 2023; Rashidinejad & Tian, 2024), their mechanisms in recommendation remain underexplored. In LLM-based recommendation (Figure 1), reward hacking appears as large training-time margins that do not translate into inference-time ranking gains.

This issue is acute in recommendation due to the one-class nature of user behavior data (Pan et al., 2008; Hu et al., 2008; He & McAuley, 2016). Unlike human value alignment with explicit positive/negative labels (Rafailov et al., 2023; Meng et al., 2024), recommenders primarily observe positives; negatives are implicitly sampled from unobserved space (Chen et al., 2020; Ding et al., 2020). This asymmetry induces a mismatch: optimization separates positives from a small set of sampled

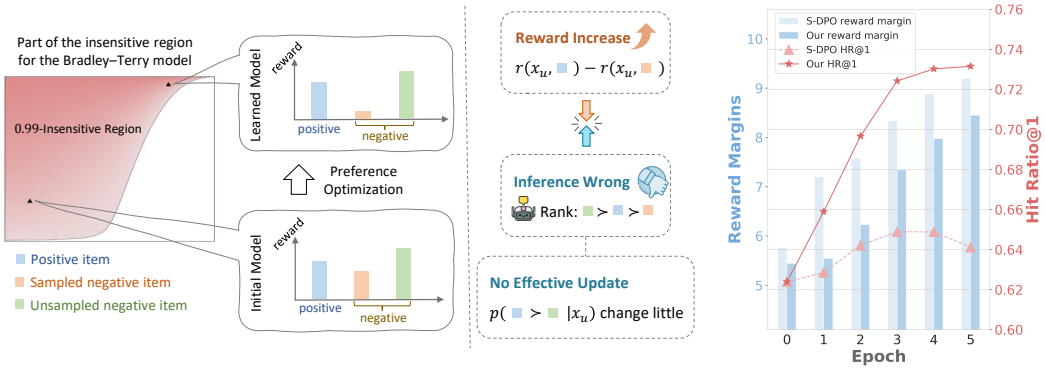


Figure 1: Illustrations of reward hacking in recommendation. (a) Left: An item in the 0.99-insensitive region prevents the unsampled negative (green) from learning from the positive (blue) and sampled negative (orange), inflating training margins while leaving inference rankings incorrect. (b) Right: On the LastFM dataset, S-DPO reward margins grow with epochs, but performance peaks early and then declines, showing reward hacking. In contrast, applying SIRIUS to one preference optimization method maintains stable performance without such signs even after more epochs.

negatives while leaving the vast pool of unsampled items—many of them poor recommendations (Chen et al., 2023; Ma et al., 2024)—largely untouched. Empirically, Bai et al. (2024) observe that many unsampled negatives initially receive higher rewards than positives, yet their rankings barely change during training. We show this persistence arises from an **ε-insensitive region** in which pairwise updates from positive–negative comparisons have negligible influence on the relative order of unsampled items. Theoretically, this region can cover up to 63.8% of item pairs (*e.g.,* under a Bradley–Terry model with $\varepsilon = 0.99$), causing margins on sampled pairs to inflate the objective while leaving the global rankings largely unchanged, as illustrated in Figure 1.

Guided by this key insight, we propose **Si**mulate Preference Optimization for **R**eward-hacking m**i**tigation **us**ing Pseudo-negatives (**SIRIUS**). The key idea is to introduce **pseudo-negative samples**—hypothetical items deliberately constructed to act as anchors, ensuring that optimization signals extend beyond sampled pairs and continue to update unsampled pairs. We prove that for any pairwise preference model with bounded rewards, such pseudo-negatives can always be constructed as stable anchors. By coupling positives with pseudo-negatives, optimization signals propagate beyond sampled pairs, reducing the measure of the ε-insensitive region and mitigating reward hacking.

Building on this theory, we analyze three public benchmarks—LastFM (Cantador et al., 2011), Goodreads[1], and Steam (Kang & McAuley, 2018b)—and find that a substantial fraction of data points lies within the ε-insensitive region, indicating a systemic vulnerability. Extensive experiments show that SIRIUS consistently mitigates reward hacking and improves ranking accuracy.

2 Preliminary

In this section, we commence by formally defining the task of sequential recommendation as the alignment of LLMs with user preferences. Then, we present the general framework of current LLM4Rec methods, which employ language modeling objectives to fine-tune LMs. Finally, we provide a detailed exposition of the prevalent training methodologies for aligning LMs with human preferences, encompassing direct preference optimization.

2.1 Task Formulation

Given a chronologically ordered historical interaction sequence $\mathcal{H}_u = \{i^1, i^2, \ldots, i^n\}$ for user u, sequential recommendation aims to predict the next item i^{n+1} that the user will be interested in,

[1] https://www.goodreads.com/

based on the sequence of historical items. This prediction is made from a candidate set $\mathcal{C}_u$, where $\mathcal{C}_u = \{i^j\}_{j \in \mathcal{J}_u}$ and $\mathcal{J}_u$ contains i^{n+1}, is a subset of size N drawn from the total item collection $\mathcal{I}$.

2.2 Supervised Fine-Tuning of LLM-based Recommender

Supervised Fine-Tuning (Ouyang et al., 2022) is commonly adopted in LLM-based recommenders to enhance their performance on recommendation-specific data (Zhao et al., 2024; Wu et al., 2024). A typical SFT procedure consists of two steps: transforming recommendation data into text-based prompts, and fine-tuning language models using these prompts along with the next preferred items as the supervision signal. For the first step, a recommendation task prompt x_u for user u includes a description of the sequential recommendation task, the user's historical interactions $\mathcal{H}_u$, and the candidate item set $\mathcal{C}_u$. The expected answer i^{n+1}, denoted as y_u, is the target of the prompt. Consequently, the SFT dataset $\mathcal{D}_{\text{SFT}} = \{(x_u, y_u)\}$ is generated. For the second step, the constructed prompts and its target (x_u, y_u) are utilized to fine-tune the LLM which is parameterized by θ through language modeling loss. The objective is to maximize the likelihood of the chosen response y_u given the input prompt x_u:

$$\max_{\theta} \mathbb{E}_{(x_u, y_u) \sim \mathcal{D}_{\text{SFT}}} \sum_{t=1}^{|y_u|} \log \left[p_\theta \left(y_u^{(t)} | x_u, y_u^{(1:t-1)} \right) \right]. \tag{1}$$

Here $|y_u|$ denotes the number of tokens in y_u, $y_u^{(t)}$ denotes the t-th token of y_u, and $y_u^{(1:t-1)}$ is the tokens preceding $y_u^{(t)}$.

2.3 Preference Optimization

During the stage of preference alignment in LLMs, preference optimization methods require transferred data, similar as Section 2.2. The preference alignment dataset $\mathcal{D}_{\text{PO}}$ contains triples (x_u, y_u^i, y_u^j), where $(x_u, y_u^i) \in \mathcal{D}_{\text{SFT}}$, and y_u^j is uniformly sampled from $\mathcal{C}_u \setminus \{y_u^i\}$. Formally, to maximize the expected reward of the policy while minimizing deviation from the reference model, RLHF optimizes the following objective for the optimal policy:

$$\max_{\pi_\theta} \mathbb{E}_{x_u \sim \mathcal{D}, y \sim \pi_\theta(y|x_u)} \left[r(x_u, y) \right] - \beta \mathbb{D}_{\text{KL}} \left[\pi_\theta(y|x_u) \,||\, \pi_{\text{ref}}(y|x_u) \right]. \tag{2}$$

Here π_θ represents the likelihood of the policy model parameterized by θ, while π_{ref} denotes the likelihood of a reference model with frozen parameters, typically the model obtained after SFT. The first term encourages the policy to maximize the expected reward $r(x_u, y)$, while the second term, controlled by the coefficient β, penalizes deviations from the reference model by minimizing the Kullback-Leibler (KL) divergence between π_θ and π_{ref}.

Direct Preference Optimization (DPO) theoretically extracts a closed-form optimal policy from RLHF and formulates the reward function as:

$$r(x_u, y) = \beta \log \frac{\pi_\theta(y|x_u)}{\pi_{\text{ref}}(y|x_u)} + \beta \log Z(x_u), \tag{3}$$

where $Z(x_u) = \sum_y \pi_{\text{ref}}(y|x_u) \exp\left(\frac{1}{\beta} r(x_u, y) \right)$ is a function independent of both y and the policy model π_θ. The likelihood of preference $p(y_u^i \succ y_u^j | x_u)$ can be estimated by Bradley-Terry model (Bradley & Terry, 1952) as:

$$p(y_u^i \succ y_u^j \mid x_u) = \sigma(r(x_u, y_u^i) - r(x_u, y_u^j)), \tag{4}$$

where σ refers to the sigmoid function. Then, DPO optimizes the preference model as:

$$\begin{aligned} \mathcal{L}_{\text{DPO}} &= -\mathbb{E}_{(x_u, y_u^i, y_u^j) \sim \mathcal{D}_{\text{PO}}} \log p(y_u^i \succ y_u^j \mid x_u). \\ &= -\mathbb{E}_{(x_u, y_u^i, y_u^j) \sim \mathcal{D}_{\text{PO}}} \log \sigma \left(\beta \log \frac{\pi_\theta(y_u^i \mid x_u)}{\pi_{\text{ref}}(y_u^i \mid x_u)} - \beta \log \frac{\pi_\theta(y_u^j \mid x_u)}{\pi_{\text{ref}}(y_u^j \mid x_u)} \right). \end{aligned} \tag{5}$$

As a variant of DPO, Simple Preference Optimization (SimPO) removes the reference term in the reward function, making it a reference-free approach (Meng et al., 2024; Xu et al., 2024):

$$r(x_u, y) = \frac{\beta}{|y|} \log \pi_\theta(y|x_u), \tag{6}$$

The optimization objective of SimPO is formulated as:

$$\mathcal{L}_{\text{SimPO}} = -\mathbb{E}_{(x_u, y_u^i, y_u^j)\sim\mathcal{D}_{\text{PO}}} \log \sigma \left(\frac{\beta}{|y^i|} \log \pi_\theta(y^i|x) - \frac{\beta}{|y^j|} \log \pi_\theta(y^j|x) - \gamma \right), \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function, and the fixed reward margin γ enforces a minimum separation to distinguish reward differences.

Taken together, preference optimization methods provide principled frameworks to align policy models with user preferences. However, their optimization dynamics in recommendation remain underexplored. In the next section, we develop a theoretical analysis to reveal potential pitfalls, highlighting the mechanism of reward hacking.

3 Theoretical Analysis of Reward Hacking

3.1 General Theoretical Framework

Before presenting the formal theory, we formalize the intuitive gradient analysis of certain preference optimization algorithms (Appendix C) as the following proposition.

Proposition 1. *For preference optimization methods such as DPO and SimPO, the gradient of the pairwise objective with respect to model logits depends only on the sampled positive–negative pair (y_u^i, y_u^j). In particular, the logits of all other items y_u^k $(k \neq i, j)$ receive zero direct gradient signal and thus are not explicitly optimized during training.*

This property explains why sampled negatives dominate training while unobserved negatives remain unsupervised, providing a preliminary explanation of reward hacking. However, to examine whether this insensitivity is algorithm-specific or a structural property of pairwise preference models, we next develop a general theoretical framework. We begin with the following definitions.

Definition 1 (Pairwise preference model)**.** We consider a general pairwise preference model that depends only on reward differences, following Xu & Kankanhalli (2025). Formally, let $r_u^i \in \mathbb{R}$ denote the reward score of item y_u^i given user history x_u. A pairwise preference function p is defined as

$$p_u^{ij} = p(r_u^i - r_u^j), \quad p : \mathbb{R} \to (0, 1),$$

where p is strictly monotone and satisfies mild regularity conditions. In addition, the model satisfies the natural symmetry $p_u^{ij} = 1 - p_u^{ji}$. These assumptions hold for common preference models (*e.g.,* Bradley–Terry). And we use the notation $p_u^i \succ p_u^j$ to denote that the model prefers p_u^i over p_u^j.

Definition 2 (ε-insensitivity region)**.** Let $h(\boldsymbol{x})$ be a differentiable function over $\boldsymbol{x} = (x_1, \ldots, x_n)$. For $\varepsilon > 0$, we say h is ε-insensitive to x_i at x_i' if

$$\left| \frac{\partial h(\boldsymbol{x})}{\partial x_i}\Big|_{x_i = x_i'} \right| < \varepsilon.$$

The ε-insensitive region of h with respect to x_i is then

$$\Omega_\varepsilon(h, x_i) = \left\{ \boldsymbol{x} \in \mathcal{D}(h) \;\middle|\; |\partial h(\boldsymbol{x})/\partial x_i| < \varepsilon \right\}.$$

Intuitively, an ε-insensitive region characterizes "flat zones" of a function, where changes in one variable produce vanishingly small effects on the output. In such regions, optimization signals along that variable become negligible, meaning that training may push strongly on some inputs while having almost no effect on others.

With these definitions in place, we now turn to the relationship between sampled and unsampled preference pairs. To reason about how training pairs influence unsampled pairs, we first establish a decomposition property of general pairwise preference models.

Proposition 2 (Preference decomposition via a third item)**.** *For any user u and items $y_u^i, y_u^j, y_u^k \in \mathcal{I}$, under the pairwise preference model p as definition 1, the following decomposition holds: we have*

$$p_u^{ik} = p\big(p^{-1}(p_u^{ij}) + p^{-1}(p_u^{jk})\big)\,, \tag{8}$$

where $p^{-1} : (0,1) \to \mathbb{R}$ is the inverse of p, and $\mathcal{I}$ denotes the set of all items.

Proposition 2 shows that the preference between (y_u^i, y_u^k) can be expressed through the preferences (y_u^i, y_u^j) and (y_u^j, y_u^k) via decomposition. This property will play a central role in our subsequent analysis of ε-insensitive regions. In particular, it implies that p^{ik} can be regarded as a function of p^{ij}, allowing us to study the sensitivity of p^{ik} with respect to variations in p^{ij}. The proof is deferred to Appendix B.1.

We can show that such ε-insensitive regions exist for all $0 < \varepsilon < 1$, as stated in the following theorem.

Theorem 1 (Existence of ε-insensitive regions)**.** *Let p be any pairwise preference model as in definition 1, and fix $\varepsilon > 0$. For any pair (y^i, y^j) with $0 < p^{ij} < 1$, there exist constants $0 < M_1 < M_2 < 1$ such that for all $p^{ik} \in (0, M_1) \cup (M_2, 1)$ we have*

$$\left|\frac{\partial p^{ik}}{\partial p^{ij}}\right| < \varepsilon.$$

In other words, pairwise preferences inevitably contain ε-insensitive regions: even when a sampled pair (y^i, y^j) receives strong gradient updates, the influence on an unsampled pair (y^i, y^k) decays and eventually vanishes once p^{ik} enters such a region. This is not a flaw of specific algorithms like DPO or SimPO but a structural property of pairwise models. As a result, optimization on sampled pairs cannot reliably alter unsampled comparisons, leading to apparent training improvements that fail to generalize—precisely the essence of reward hacking. The full proof is deferred to Appendix B.2.

In the context of recommendation, this implies that sampled negatives may lie in regions where training updates exert negligible influence on unsampled negatives. Consequently, the model may exhibit apparent improvements—via increased reward margins on sampled pairs—while leaving most unsampled comparisons unaffected, which is precisely the phenomenon of reward hacking.

While Theorem 1 shows that ε-insensitive regions are structurally unavoidable, their practical impact depends on how large these regions are. To make this effect concrete, we next specialize to the Bradley–Terry model, which admits closed-form analysis of gradient insensitivity.

3.2 Specialization to the Bradley–Terry Model

To make the existence theorem more concrete, we specialize the analysis to the widely used Bradley–Terry (BT) model (Bradley & Terry, 1952), which enables closed-form characterization of ε-insensitive regions. For clarity, we omit the user index u and input interaction history x_u, and denote samples as (y^i, y^j, y^k) corresponding to (positive item, sampled negative in training, unsampled negative).

Under this model, the pairwise preference is

$$p^{ij} = p_{\text{BT}}(r^i - r^j) = \frac{1}{1 + e^{-(r^i - r^j)}}.$$

According to Proposition 2, p^{ik} can be written as a function of p^{ij}, p^{kj}:

$$p^{ik} = \frac{1}{1 + \frac{(1-p^{ij})(1-p^{jk})}{p^{ij}p^{jk}}} = \frac{1}{1 + \frac{(1-p^{ij})p^{kj}}{p^{ij}(1-p^{kj})}}. \tag{9}$$

The derivative of p^{ik} with respect to p^{ij} can be derived as

$$\frac{\partial p^{ik}}{\partial p^{ij}} = \frac{p^{kj}(1-p^{kj})}{\left(p^{ij} + p^{kj} - 2p^{ij}p^{kj}\right)^2}. \tag{10}$$

The derivative quantifies how much the preference between (y^i, y^k) changes when the sampled pair (y^i, y^j) is optimized. When this derivative becomes very small, it means that updates on (y^i, y^j) have little effect on (y^i, y^k), directly reflecting ε-insensitivity in the preference model. This closed form enables us to precisely characterize the ε-insensitivity region under the BT model, which identifies areas where changes to the training pair (y^i, y^j) have negligible impact on the relative ranking of (y^i, y^k). Figure 2 plots contour lines of $\left|\partial p^{ik}/\partial p^{ij}\right|$ for different values of ε. These regions correspond to areas where the influence of changes in the sampled pair (y^i, y^j) on the preference of (y^i, y^k) becomes negligible.

While Theorem 1 shows that ε-insensitive regions inevitably exist, it does not tell us how large these regions are in practice. This distinction is crucial: if the insensitive area is vanishingly small, reward hacking would be a theoretical curiosity but not a real obstacle. The Bradley–Terry model provides a tractable case where we derive the exact size of these regions, quantifying their practical significance. In particular, we compute the analytic area of the ε-insensitive region, as stated below.

Proposition 3 (Area of ε-insensitivity region)**.** *The area of* $\Omega_\varepsilon(p^{ik}, p^{ij})$*, where* $0 < \varepsilon < 1$*, is*

$$A\left(\Omega_\varepsilon(p^{ik}, p^{ij})\right) = 1 - \frac{1}{2}\ln\left(\frac{1-\varepsilon}{1+\varepsilon}\right) - \frac{1}{2\sqrt{\varepsilon}}\ln\left(\frac{1+\sqrt{\varepsilon}}{1-\sqrt{\varepsilon}}\right).$$

The proof is deferred to Appendix B.3. This expression shows that the area is a monotone increasing function of ε. For example, when $\varepsilon = 0.5$, the region occupies approximately 30.3% of the (p^{ij}, p^{ik}) space, indicating that the influence of such regions is far from negligible.

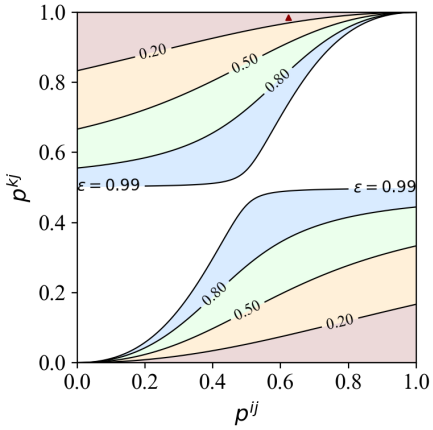


Figure 2: Contour plot of $\left|\partial p^{ik}/\partial p^{ij}\right|$ under the Bradley–Terry model. Darker regions indicate ε-insensitivity, where updates on (y^i, y^j) have negligible influence on (y^i, y^k).

The following example illustrates how entering the ε-insensitive region of (p^{ij}, p^{ik}) during training can cause negative effects.

Example 1. We take the following reward tuple: $(r^i, r^j, r^k) = (-6, -6.5, -4)$. This particular choice of values is illustrative rather than essential; it simply provides a concrete point for visualization in Figure 2. In this case, the preference between the positive item and the sampled negative is moderate ($p^{ij} \approx 0.62$), while the preference against the unsampled negative is extremely low ($p^{ik} \approx 0.04$). Moreover, the derivative $\left|\partial p^{ik}/\partial p^{ij}\right|$ is only 0.19, placing the point well inside the 0.2-insensitive region. During training, optimization may enlarge the reward margin $r^i - r^j$ from 0.5 to 4, yet p^{ik} remains almost unchanged. This demonstrates that even with a substantially enlarged reward margin on the training pair, the relative preference against unsampled negatives stays nearly unaffected, creating the illusion of progress.

The specialization to the Bradley–Terry model demonstrates that ε-insensitive regions are not only a theoretical possibility but can in fact occupy a substantial portion of the preference space. Within these regions, enlarging the reward margin on sampled pairs (y^i, y^j) exerts negligible influence on unsampled pairs (y^i, y^k), which are critical for determining recommendation quality.

In summary, the core issue is not insufficient margins but the lack of gradient propagation to unsampled negatives, leaving them trapped in ε-insensitive regions. This structural gap drives reward hacking, creating the illusion of progress without real improvement. Motivated by this insight, we next introduce a principle for mitigation: providing more informative contrastive signals to shrink the ε-insensitive region.

4 Method

Building on the theoretical insights from Section 3, we now design a practical framework to mitigate reward hacking. Our theoretical analysis revealed that structural *ε-insensitive regions* are intrinsic to

pairwise preference models. Within these regions, optimization signals from sampled pairs (y^i, y^j) fail to propagate to unsampled pairs (y^i, y^k), causing reward hacking: the apparent enlargement of reward margins does not improve the global preference ordering. To mitigate this issue, a natural principle emerges: we must introduce more informative contrastive signals that reduce the likelihood of unsampled comparisons being trapped in ε-insensitive regions.

Guided by this principle, we propose **Si**mulate Preference Optimization for **R**eward-hacking m**i**tigation **us**ing Pseudo-negatives (**SIRIUS**). The key idea is to augment the training objective with a *pseudo-negative sample*, a virtual item that does not belong to the observed item set $\mathcal{I}$ but is deliberately constructed to promote more effective gradient flow toward unsampled pairs. Unlike standard negative sampling, which draws from unobserved real items, pseudo-negative acts as *anchors* that guarantee non-vanishing influence on unsampled pairs. Although methods such as S-DPO use multiple sampled negatives and alleviate this issue to some extent (Appendix D.1), the structural insensitivity remains and the computational cost is high.

4.1 Existence of Pseudo-Negative

Theorem 2 (Existence of Pseudo Negatives)**.** *Given a user history x_u with positive item y_u^i, and assuming an upper bound $N \in \mathbb{R}$ on item rewards. For any pairwise preference model as in definition 1, there always exists a pseudo negative item $y_u^0 \notin \mathcal{I}$ with fixed reward $r_u^0 \in \mathbb{R}$ such that, for any negative item $y_u^k \succ y_u^i$, the corresponding preference p_u^{ik} does not lie within any ε-insensitive region defined along p_u^{i0}.*

A formal proof is provided in Appendix B.4. This theorem guarantees that pseudo-negative always exists, serving as universal anchors that fundamentally reshape the optimization landscape.

Given this existence result, we augment the standard preference optimization objective as:

$$\max_{x_u \sim \mathcal{D}} \sum_{y_u^n \in \mathcal{N}_u} \log p(y_u^i \succ y_u^n \mid x_u), \tag{11}$$

where $\mathcal{N}_u = \{y_u^j, y_u^0\}$ denotes the augmented negative set with a sampled negative y_u^j and a pseudo-negative y_u^0. In practice, the pseudo-negative is not sampled from the data distribution but introduced as a virtual item to reduces the risk of vanishing gradients for unsampled pairs.

4.2 Specialization to the Bradley-Terry Model

To further illustrate the effect of the pseudo-negative, we instantiate our framework under the Bradley-Terry (BT) model. By introducing a pseudo-negative y_u^0 with input-dependent reward r_u^0, the probability p_u^{ik} for an unsampled negative y_u^k becomes jointly influenced by both p_u^{ij} and p_u^{i0}. This ensures that even when the gradient from p_u^{ij} vanishes due to ε-insensitivity, the gradient from p_u^{i0} continues to propagate, thereby exerting meaningful influence on p_u^{ik}. As a result, misordered pairs $(y_u^k \succ y_u^i)$ are guaranteed to receive effective corrective gradients. This mechanism eliminates the pathological case where unsampled comparisons are trapped in ε-insensitive regions, ensuring robust optimization signals. A detailed derivation and geometric analysis of this property under the BT model are provided in Appendix D.2.

4.3 Practical Selection of the Pseudo-Negative Reward r_u^0

According to Theorem 2, the anchor reward r_u^0 must strictly exceed the upper bound of the rewards assigned to unsampled negatives, ensuring that the positive–pseudo-negative pair lies outside the ε-insensitive region for all positive–unsampled-negative comparisons. Existing BT-based preference optimization frameworks can be broadly categorized into two types of implicit reward structures: (i) *reference-based* implicit rewards and (ii) *reference-free* implicit rewards. DPO is a canonical example of the former, while SimPO represents the latter. We detail below how r_u^0 should be constructed under each formulation.

DPO. DPO adopts the reference-based implicit reward

$$r_{\text{DPO}}(x_u, y) = \beta \log \frac{\pi_\theta(y \mid x_u)}{\pi_{\text{ref}}(y \mid x_u)} + \beta \log Z(x_u),$$

where the presence of π_{ref} makes it difficult to estimate a universal upper bound for r_{DPO}. To obtain a practical bound, we focus on unsampled negatives satisfying

$$\pi_{\text{ref}}(y_u^k \mid x_u) \geq \pi_{\text{ref}}(y_u^i \mid x_u),$$

as these are precisely the items mis-ranked by the model and are the dominant contributors to reward hacking. For such k, we have

$$\begin{aligned} r_u^k &= \beta \log \frac{\pi_\theta(y_u^k \mid x_u)}{\pi_{\text{ref}}(y_u^k \mid x_u)} + \beta \log Z(x_u) \\ &\leq \beta \log \pi_\theta(y_u^k \mid x_u) - \beta \log \pi_{\text{ref}}(y_u^i \mid x_u) + \beta \log Z(x_u) \\ &< -\beta \log \pi_{\text{ref}}(y_u^i \mid x_u) + \beta \log Z(x_u), \end{aligned}$$

which provides an approximate upper bound for these problematic negatives. Accordingly, we set

$$r_u^0 = -\beta \log \pi_{\text{ref}}(y_u^i \mid x_u) + \beta \log Z(x_u),$$

yielding an x_u-dependent anchor that safely dominates the unsampled negatives under DPO's reward structure.

SimPO. In contrast, SimPO uses the reference-free implicit reward

$$r_{\text{SimPO}}(x_u, y) = \beta \log \pi_\theta(y \mid x_u),$$

which enjoys favorable mathematical properties due to its independence from any reference model. Since $\log \pi_\theta(y \mid x_u) \leq 0$, this reward is always negative and is naturally bounded above. Thus, choosing a constant anchor

$$r_u^0 = 0$$

satisfies the requirement $r_u^0 > \max_{k \in \text{unsampled}} r_u^k$ without affecting optimization stability.

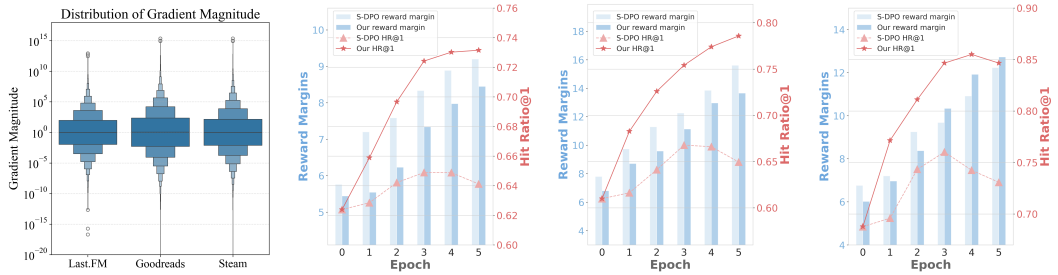
Because SimPO uses a reference-free and inherently bounded reward, a fixed pseudo-negative reward suffices. In contrast, DPO relies on a reference-based reward, which requires an anchor that depends on the input x_u and is computed from the reference model. Despite this difference, both constructions satisfy the theoretical requirement that pseudo-negatives must dominate unsampled negatives, thereby preventing optimization from drifting into the ε-insensitive region. The corresponding pseudocode is provided in Appendix D.3.

5 Experiment

Our experiments aim to empirically validate the existence of ε-insensitive regions, examine their role in reward hacking, and assess whether SIRIUS effectively mitigates this phenomenon while improving recommendation quality. We compare against a broad spectrum of baselines on three benchmarks, including: (i) sequential recommenders such as GRU4Rec (Hidasi et al., 2016), Caser (Tang & Wang, 2018), and SASRec (Kang & McAuley, 2018a); (ii) LLM-based recommenders such as LLaMA2 (Touvron et al., 2023), ChatRec (Gao et al., 2023b), MoRec (Yuan et al., 2023), TallRec (Bao et al., 2023b), and LLaRA (Liao et al., 2024b); and (iii) preference optimization methods including DPO (Rafailov et al., 2023), SimPO (Meng et al., 2024), S-DPO (Chen et al., 2024b), and SimPO without length normalization (SimPO w/o LN). The latter variant is included because, in recommendation tasks, model outputs correspond to item identifiers rather than natural language sentences, making length normalization unnecessary. Dataset details, baseline descriptions, and training/evaluation settings are deferred to Appendix E.1–E.3.

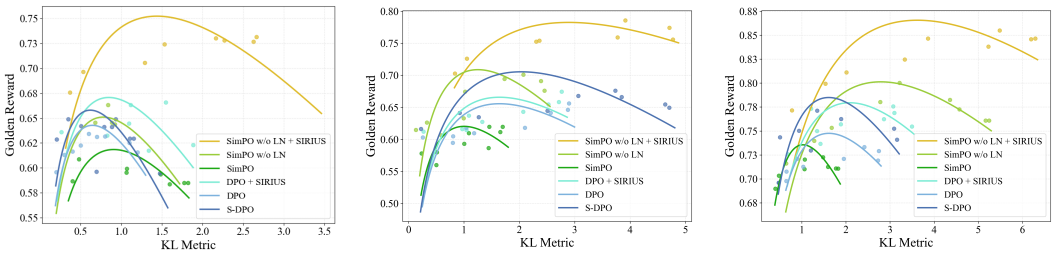
5.1 Reward Hacking Analysis and Mitigation

Portion of ε-Insensitive Regions. Our theoretical analysis in Section 3 suggests that ε-insensitive regions occupy a significant portion of the data distribution, which plays a crucial role in the emergence of reward hacking. To verify this, we empirically measure the proportion of ε-insensitive regions across three benchmark datasets. Figure 3a visualizes the gradient distribution and shows that a substantial portion of samples reside in flat regions where gradients vanish. Across all three datasets, more than 40% of samples fall inside the 0.1-insensitive region. This empirical pattern aligns with our theoretical prediction that reward hacking fundamentally stems from the prevalence of ε-insensitive regions.



(a) Distribution of Gradient. (b) LastFM. (c) Goodreads. (d) Steam.

Figure 3: (a) Distribution of ε-insensitive regions across datasets, confirming their prevalence. (b–d) Reward margin and recommendation quality over training epochs on three datasets, showing earlier reward hacking in SimPO w/o LN + SIRIUS compared to S-DPO.



(a) LastFM. (b) Goodreads. (c) Steam.

Figure 4: Fitted curves modeling the relationship between golden reward and divergence-based distance for DPO variants with and without SIRIUS across datasets. Incorporating SIRIUS consistently raises the peak golden reward, indicating delayed onset of reward hacking. Although S-DPO benefits from multiple negatives, it still lags behind SIRIUS.

Mitigation Effectiveness. We next evaluate whether adding SIRIUS can mitigate reward hacking in practice. Figures 3b–3d plot the evolution of reward margin and recommendation performance as training progresses. A clear pattern emerges: while reward margins consistently increase with more epochs, the performance of S-DPO peaks early (around epoch 3) and then declines across all datasets, a hallmark of reward hacking. In contrast, adding SIRIUS continues to improve beyond this point, demonstrating its ability to delay and reduce the severity of reward hacking.

To further quantify this effect, we follow the over-optimization analysis of Gao et al. (2023a), which models the relationship between the golden reward (here measured by HitRatio@1) and the divergence-based distance

$$d := \sqrt{D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})},$$

where π_θ is the current policy model and π_{ref} is the initial model. The fitted function takes the form

$$R(d) = d(\alpha - \beta \log d),$$

where α and β are dataset-specific coefficients. Intuitively, a lower peak of the golden reward indicates that reward hacking occurs more severely.

As shown in Figures 4a–4c, incorporating SIRIUS consistently raises the peak golden reward for each preference optimization baseline, confirming that reward hacking only occurs after reaching higher levels of true performance. This aligns with our theoretical claim that pseudo-negative provides stronger supervision signals. In addition, we observe that S-DPO consistently achieves higher fitted rewards than vanilla DPO across all divergence levels, owing to the use of multiple negative samples; however, it still lags behind SIRIUS.

Table 1: Performance comparison of SIRIUS with sequential recommenders, LLM-based methods, and DPO-based methods on three benchmark datasets. All DPO-based methods, are implemented with the LLaMA-2-7B backbone for fair comparison. **Bold** numbers indicate the best overall performance, and underlined numbers denote the best among baseline methods. Imp.% reports the relative improvement of SimPO w/o LN + SIRIUS over baselines.

Model	LastFM			Goodreads			Steam		
	HitRatio@1	ValidRatio	Imp.%	HitRatio@1	ValidRatio	Imp.%	HitRatio@1	ValidRatio	Imp.%
GRU4Rec	0.2666	1.0000	172.69%	0.3867	1.0000	103.18%	0.4121	1.0000	107.47%
Caser	0.2410	1.0000	201.66%	0.4174	1.0000	88.24%	0.4288	1.0000	99.39%
SASRec	0.2492	1.0000	191.73%	0.3581	1.0000	119.41%	0.4037	1.0000	111.79%
Llama2	0.0277	0.3910	2524.55%	0.0399	0.6196	1869.17%	0.0430	0.5447	1888.37%
CharRec	0.3770	1.0000	92.84%	0.3306	1.0000	137.66%	0.3626	0.9798	135.80%
MoRec	0.1652	1.0000	340.07%	0.2877	1.0000	173.10%	0.3911	1.0000	118.61%
TALLRec	0.4180	0.9836	73.92%	0.4983	0.9573	57.68%	0.4637	0.9840	84.39%
LLaRA	0.4750	0.9920	53.05%	0.5292	0.9950	48.47%	0.4927	0.9975	73.53%
DPO	0.6393	0.9984	13.72%	0.6462	0.9950	21.59%	0.7333	0.9966	16.60%
SimPO	0.5928	0.9804	22.64%	0.6130	0.9452	28.17%	0.7234	0.9907	18.20%
SimPO w/o LN	0.6633	0.9972	9.60%	0.7010	0.9884	12.08%	0.8002	0.9983	6.85%
S-DPO	0.6493	0.9984	11.97%	0.6744	0.9950	16.50%	0.7712	0.9992	10.87%
+ SIRIUS									
DPO	0.6593	0.9984	✓	0.6744	0.9900	✓	0.7757	0.9983	✓
SimPO w/o LN	**0.7315**	0.9984	✓	**0.7857**	0.9900	✓	**0.8550**	0.9983	✓

5.2 Overall Performance

From the results in Table 1, we observe that incorporating SIRIUS into the LLaMA-2-7B backbone consistently outperforms all baselines across the three datasets. In particular, when applied to SimPO w/o LN, SIRIUS achieves superior performance over all other baselines. When applied to DPO, SIRIUS not only surpasses vanilla DPO but also outperforms its stronger variant S-DPO with three negative samples, demonstrating both the effectiveness and extensibility of our approach. We attribute these improvements to the introduction of pseudo-negative samples, which reduce the likelihood of unsampled pairs falling into the ε-insensitive region and thereby provide more effective supervision signals, ultimately mitigating reward hacking. Additional experimental results and analyses are provided in Appendix E.4.

6 Limitations

Our method builds on gradient-based observations to explain and mitigate reward hacking. While it currently does not exploit multimodal information (*e.g.,* visual or audio signals of items), exploring multimodal embedding to construct richer pseudo-negative samples and enhance contrastive learning effectiveness remains an interesting direction for future research.

7 Conclusion

We studied the challenge of reward hacking in LLM-based recommendation and showed that ε-insensitive regions are a structural property of pairwise models. This insight explains why enlarging reward margins on sampled pairs may not improve global ranking. To mitigate this, we proposed SIRIUS, which introduces pseudo-negatives as virtual anchors to provide stronger contrastive signals. Both theoretical analysis and experiments on three benchmarks confirm that SIRIUS reduces reward hacking and improves recommendation quality.

Statement

Ethics Statement This work proposes a preference optimization method designed to mitigate reward hacking and improve global ranking. We do not anticipate any negative social impacts or violations of the ICLR Code of Ethics.

Reproducibility Statement All results in this paper are fully reproducible. The assumptions used for the reward hacking analysis are clearly explained in the main text. Proofs of theorems, propositions, and additional methodological details are provided in Appendix B–D. Implementation details for experiments are included in Appendix E.1–E.3.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Zhuoxi Bai, Ning Wu, Fengyu Cai, Xinyi Zhu, and Yun Xiong. Aligning large language model with direct multi-preference optimization for recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, pp. 76–86, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679611. URL https://doi.org/10.1145/3627673.3679611.

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnan He, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. *CoRR*, abs/2308.08434, 2023a.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, pp. 1007–1014. ACM, 2023b.

Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. Decoding matters: Addressing amplification bias and homogeneity issue for llm-based recommendation. *EMNLP*, 2024.

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

Shihao Cai, Jizhi Zhang, Keqin Bao, Chongming Gao, and Fuli Feng. Flow: A feedback loop framework for simultaneously enhancing recommendation and user agents. *arXiv preprint arXiv:2410.20027*, 2024.

Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik (eds.). *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '11, Chicago, Illinois, USA, October 27, 2011*, 2011. ACM.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Tong Wang, Samuel Marks, Charbel-Raphaël Ségerie, Micah Carroll, Andi Peng, Phillip J. K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca D. Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=bx24KpJ4Eb.

Chong Chen, Min Zhang, Yongfeng Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. Efficient heterogeneous collaborative filtering without negative sampling for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 19–26, 2020.

Chong Chen, Weizhi Ma, Min Zhang, Chenyang Wang, Yiqun Liu, and Shaoping Ma. Revisiting negative sampling vs. non-sampling in implicit recommendation. *ACM Trans. Inf. Syst.*, 41(1), February 2023. ISSN 1046-8188. doi: 10.1145/3522672. URL https://doi.org/10.1145/3522672.

Jiaju Chen, Chongming Gao, Shuai Yuan, Shuchang Liu, Qingpeng Cai, and Peng Jiang. Dlcrec: A novel approach for managing diversity in llm-based recommender systems. *The 18th ACM International Conference on Web Search and Data Mining (WSDM '25)*, 2025.

Lichang Chen, Chen Zhu, Jiuhai Chen, Davit Soselia, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. Odin: disentangled reward mitigates hacking in rlhf. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

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

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NIPS*, pp. 4299–4307, 2017.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. Uncovering chatgpt's capabilities in recommender systems. In *RecSys*, pp. 1126–1132. ACM, 2023.

Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. Simplify and robustify negative sampling for implicit collaborative filtering. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1094–1105. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0c7119e3a6a2209da6a5b90e5b5b75bd-Paper.pdf.

Chongming Gao, Ruijun Chen, Shuai Yuan, Kexin Huang, Yuanqing Yu, and Xiangnan He. Sprec: Leveraging self-play to debias preference alignment for large language model-based recommendations. *arXiv preprint arXiv:2412.09243*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023a. URL https://proceedings.mlr.press/v202/gao23h.html.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *CoRR*, abs/2303.14524, 2023b.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*, 2023c.

Binzong Geng, Zhaoxin Huan, Xiaolu Zhang, Yong He, Liang Zhang, Fajie Yuan, Jun Zhou, and Linjian Mo. Breaking the length barrier: Llm-enhanced ctr prediction in long textual user behaviors. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 2311–2315, 2024. ISBN 9798400704314.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *RecSys*, pp. 299–315. ACM, 2022.

Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR (Poster)*, 2016.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 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*, pp. 364–381, 2024. ISBN 978-3-031-56059-0.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 263–272, 2008. doi: 10.1109/ICDM.2008.22.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206. IEEE Computer Society, 2018a.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206. IEEE Computer Society, 2018b.

Jiayi Liao, Xiangnan He, Ruobing Xie, Jiancan Wu, Yancheng Yuan, Xingwu Sun, Zhanhui Kang, and Xiang Wang. Rosepo: Aligning llm-based recommenders with human values. *arXiv preprint arXiv:2410.12519*, 2024a.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. Llara: Large language-recommendation assistant. In *SIGIR*, pp. 1785–1795. ACM, 2024b.

Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Bridging items and language: A transition paradigm for large language model-based recommendation. In *KDD*, pp. 1816–1826. ACM, 2024.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yejing Wang, Zijian Zhang, Feng Tian, and Yefeng Zheng. Large language models enhanced sequential recommendation for long-tail user and item. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Haokai Ma, Ruobing Xie, Lei Meng, Fuli Feng, Xiaoyu Du, Xingwu Sun, Zhanhui Kang, and Xiangxu Meng. Negative sampling in recommendation: A survey and future directions. *CoRR*, abs/2409.07237, 2024. URL https://doi.org/10.48550/arXiv.2409.07237.

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

Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Yuchun Miao, Sen Zhang, Liang Ding, Yuqi Zhang, Lefei Zhang, and Dacheng Tao. The energy loss phenomenon in rlhf: A new perspective on mitigating reward hacking, 2025. URL https://arxiv.org/abs/2501.19358.

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

Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 502–511, 2008. doi: 10.1109/ICDM.2008.16.

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

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q^*: Your language model is secretly a q-function. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kEVcNxtqXk.

Paria Rashidinejad and Yuandong Tian. Sail into the headwind: Alignment via robust rewards and dynamic labels against reward hacking. *arXiv preprint arXiv:2412.09544*, 2024.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. *CoRR*, abs/1205.2618, 2012.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pp. 565–573. ACM, 2018.

Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca D. Dragan, and Daniel S. Brown. Causal confusion and reward misidentification in preference-based reward learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=R0Xxvr_X3ZA.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. A survey on large language models for recommendation. *World Wide Web*, 27(5):60, 2024.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *ICML*. OpenReview.net, 2024.

Ziwei Xu and Mohan Kankanhalli. Strong preferences affect the robustness of preference models and value alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Upoxh7wvmJ.

Zhengyi Yang, Xiangnan He, Jizhi Zhang, Jiancan Wu, Xin Xin, Jiawei Chen, and Xiang Wang. A generic learning framework for sequential recommendation with distribution shifts. In *SIGIR*, pp. 331–340. ACM, 2023.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? ID- vs. modality-based recommender models revisited. In *SIGIR*, pp. 2639–2649. ACM, 2023.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. *CoRR*, abs/2305.07001, 2023.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM Web Conference 2024*, WWW '24, pp. 3679–3689, 2024. ISBN 9798400701719.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, 2024.

A Related Work

We provide a concise overview of LLM-based recommendation systems, highlighting challenges such as handling diverse user behavior and susceptibility to reward hacking.

A.1 LLMs for Recommendation

Large Language Models have demonstrated exceptional capabilities in natural language processing, including generative power, generalization, and complex reasoning. These advancements have spurred significant interest in leveraging LLMs for personalized recommendation tasks. Current approaches to integrating LLMs into recommendation systems can be broadly categorized into three paradigms: (1) LLMs as Recommender, which involves direct deployment of LLMs as decision-making agents (Bao et al., 2024; 2023b); (2) LLMs as Enhancer, where LLMs are used as an auxiliary enhancement, generating contextual information (*e.g.,* item descriptions or user intent explanations) to support recommendation pipelines (Liu et al., 2024; Geng et al., 2024); and (3) LLMs as Simulator, which employs LLMs to simulate user behavior for training or evaluation purposes (Zhang et al., 2024; Cai et al., 2024).

Early research primarily focused on prompt engineering to unlock LLMs' latent recommendation abilities without explicit training (Gao et al., 2023c; Hou et al., 2024). More recently, fine-tuning paradigms emerged, demonstrating that adapting LLMs to recommendation-specific datasets significantly improves task performance, especially through Supervised Fine-Tuning (SFT) frameworks (Chen et al., 2025; Liao et al., 2024b). Additionally, Direct Preference Optimization has gained traction as a method to better align LLMs with nuanced human preferences during post-training (Bai et al., 2024; Chen et al., 2024b; Liao et al., 2024a; Gao et al., 2024). However, existing studies have largely overlooked the critical issue of reward hacking in DPO-based recommendation frameworks. This behavior undermines the fairness and diversity of recommendations. To address this gap, we propose a systematic analysis of reward hacking in DPO-based recommender systems and introduce a novel mitigation strategy to align model optimization with genuine user preferences.

A.2 Reward Hacking Problem

Reward hacking is a significant issue within the reinforcement learning (RL) community, where models exploit weaknesses in reward functions to optimize rewards that do not truly align with the intended goals. Amodei et al. (Amodei et al., 2016; Skalse et al., 2022) provide an extensive analysis of reward hacking, categorizing it into various forms such as partially observed goals, complex system interactions, and abstract reward definitions. These flaws in reward specification lead to models learning unintended behaviors that undermine the overall objective, especially when reward functions are designed without careful consideration of edge cases and unintended incentives (Amodei et al., 2016; Skalse et al., 2022).

In the context of LLMs and DPO, reward hacking can occur when the models misinterpret or exploit inconsistencies in the reward signal, often leading to misgeneralization of their performance. This can happen when the reward functions do not accurately reflect the underlying task, resulting in poor reward proxies that fail to drive the desired behavior (Casper et al., 2023; Rashidinejad & Tian, 2024). According to recent work, this misalignment can introduce causal confusion, making it difficult for models to generalize effectively across different datasets or real-world scenarios, a challenge that has been highlighted in recent studies on causal inference (Tien et al., 2023; Miao et al., 2025).In the RLHF field, existing solutions aim to mitigate reward hacking and enhance the alignment between reward signals and objectives through methods such as energy loss suppression (Miao et al., 2025), decoupled reward design (Chen et al., 2024a), and information-theoretic modeling (Miao et al., 2024).

In the domain of recommendation systems, especially in LLM-based recommender, the reward hacking problem remains under-explored. However, it manifests when models exploit certain patterns or shortcuts to maximize rewards, often ignoring the broader diversity of user preferences or failing to drive meaningful policy improvement.

B Additional Proofs

B.1 Proof of Proposition 2

Proof. Since $p : \mathbb{R} \to (0,1)$ is strictly monotone, its inverse $p^{-1} : (0,1) \to \mathbb{R}$ exists. Then

$$p_u^{ik} = p(r_u^i - r_u^k) = p\big((r_u^i - r_u^j) + (r_u^j - r_u^k)\big) = p\big(p^{-1}(p_u^{ij}) + p^{-1}(p_u^{jk})\big)\,.$$

□

B.2 Proof of Theorem 1

Proof. 1° First, we take the derivative. Since p satisfies mild regularity conditions, we denote by p' the derivative of p. From Proposition 2, we have

$$p^{ik} = p\big(p^{-1}(p^{ij}) + p^{-1}(p^{jk})\big)\,, \tag{12}$$

Taking the partial derivative of p^{ik} w.r.t. p^{ij}, by the chain rule we have

$$\frac{\partial p^{ik}}{\partial p^{ij}} = \left.\frac{\mathrm{d}p(x)}{\mathrm{d}x}\right|_{x=p^{-1}(p^{ij})+p^{-1}(p^{jk})} \times \left.\frac{\mathrm{d}p^{-1}(y)}{\mathrm{d}y}\right|_{y=p^{ij}} \tag{13}$$

$$= \left.\frac{\mathrm{d}p(x)}{\mathrm{d}x}\right|_{x=r^i-r^j+r^j-r^k} \times \left(\left.\frac{\mathrm{d}p(x')}{\mathrm{d}x'}\right|_{x'=p^{-1}(p^{ij})}\right)^{-1} \tag{14}$$

$$= \frac{p'\left(p^{-1}(p^{ik})\right)}{p'\left(p^{-1}(p^{ij})\right)}. \tag{15}$$

2° We show that $\lim_{x\to-\infty} p'(x) = \lim_{x\to+\infty} p'(x) = 0$.

We prove $\lim_{x\to+\infty} p'(x) = 0$ by contradiction; the case $x \to -\infty$ is analogous.

Suppose instead that $\lim_{x\to+\infty} p'(x) \neq 0$. By strict monotonicity, we have $p'(x) \geq 0$ for all $x \in \mathbb{R}$. And denote $p_\infty = \lim_{x\to+\infty} p(x) < 1$. Therefore, we have $p(x) < p_\infty$. Hence there exists $\varepsilon_0 > 0$ and an increasing sequence $\{x_n\}$ with $p'(x_n) \geq 2\varepsilon_0$ for all $n \in \mathbb{N}$. By continuity of p', for each x_n there exists $\delta_n > 0$ such that

$$p(x_n + \delta_n) - p(x_n) \;=\; \int_{x_n}^{x_n+\delta_n} p'(x)\,\mathrm{d}x \;\geq\; \varepsilon_0\delta_n.$$

Since $\delta_n > 0$ for all n, the right-hand side strictly exceeds $\lim_{n\to\infty} p(x_n)$. Thus we would conclude that

$$p_\infty = \lim_{x\to+\infty} p(x) > \lim_{x\to+\infty} p(x) = p_\infty,$$

which is a contradiction.

3° Next, we can choose $N_1, N_2 \in \mathbb{R}$ such that for all $x \in (-\infty, N_1) \cup (N_2, \infty)$ it holds that

$$p'(x) < \varepsilon\, p'\big(p^{-1}(p^{ij})\big)\,.$$

Let $M_1 = p(N_1)$ and $M_2 = p(N_2)$. Then for all $p^{ik} \in (0, M_1) \cup (M_2, 1)$ we have

$$\left|\frac{\partial p^{ik}}{\partial p^{ij}}\right| < \varepsilon.$$

□

B.3 Proof of Proposition 3

Proof. We compute the area of the ε-insensitivity region

$$\Omega_\varepsilon(p^{ik}, p^{ij}) = \left\{ (p^{ik}, p^{ij}) \in (0,1)^2 : \left| \frac{\partial p^{ik}}{\partial p^{ij}} \right| < \varepsilon \right\}.$$

1° By Proposition 2, we know

$$\frac{\partial p^{ik}}{\partial p^{ij}} = \frac{p^{kj}(1-p^{kj})}{(p^{ij}+p^{kj}-2p^{ij}p^{kj})^2}.$$

Therefore the condition $\left|\frac{\partial p^{ik}}{\partial p^{ij}}\right| < \varepsilon$ can be expressed as

$$\frac{y(1-y)}{(x+y-2xy)^2} < \varepsilon,$$

where $x = p^{ij}$ and $y = p^{kj}$.

2° Since $x, y \in (0,1)$, we have

$$x + y - 2xy = x(1-y) + y(1-x) > 0.$$

Thus the inequality becomes

$$x + y - 2xy > \sqrt{\tfrac{y(1-y)}{\varepsilon}}.$$

Equivalently,

$$(1-2y)x > \sqrt{\tfrac{y(1-y)}{\varepsilon}} - y.$$

Hence the ε-insensitive region can be expressed as

$$\Omega_\varepsilon(p^{ik}, p^{ij}) = \left\{ (x,y) \in (0,1)^2 : \begin{array}{ll} x \geq f(y), & 0 < y < \frac{1}{2}, \\ x \leq f(y), & \frac{1}{2} < y < 1, \end{array} \right\},$$

where

$$f(y) = \frac{\sqrt{\frac{y(1-y)}{\varepsilon}} - y}{1-2y}.$$

3° The area of Ω_ε is

$$A(\Omega_\varepsilon) = \iint_{\Omega_\varepsilon} dx\, dy = \int_0^{\frac{1}{2}} \Big(\int_{f(y)}^1 dx \Big) dy + \int_{\frac{1}{2}}^1 \Big(\int_0^{f(y)} dx \Big) dy.$$

Evaluating these integrals and simplifying yields

$$A(\Omega_\varepsilon(p^{ik}, p^{ij})) = 1 - \tfrac{1}{2} \ln\Big(\tfrac{1-\varepsilon}{1+\varepsilon}\Big) - \tfrac{1}{2\sqrt{\varepsilon}} \ln\Big(\tfrac{1+\sqrt{\varepsilon}}{1-\sqrt{\varepsilon}}\Big),$$

which matches the claimed formula. □

B.4 Theorem 2

Proof. It suffices to construct such a pseudo-negative item in order to prove the theorem. The construction proceeds as follows.

From appendix B.2, we know that

$$\frac{\partial p_u^{ik}}{\partial p_u^{i0}} = \frac{p'\big(p^{-1}(p_u^{ik})\big)}{p'\big(p^{-1}(p_u^{i0})\big)},$$

Let

$$m = \min_{x \in [r_u^i - N, 0]} p'(x).$$

Since for any y_u^k we have $r_u^i < r_u^k < N$, it follows that

$$p'\big(p^{-1}(p_u^{ik})\big) \;\geq\; m.$$

Now consider any $r_u^0 \in (p')^{-1}\big((0, m/\varepsilon)\big)$, where for a non-bijective mapping p', the notation

$$(p')^{-1}(\mathcal{S}) \;=\; \{\, x \mid p'(x) \in \mathcal{S} \,\},$$

denotes the preimage of a set $\mathcal{S}$ under p'.

In this case, we obtain

$$\left| \frac{\partial p^{ik}}{\partial p^{ij}} \right| > \varepsilon,$$

which implies that p_u^{ik} does not lie within any ε-insensitive region defined with respect to p_u^{i0}. □

C Heuristic Analysis

This appendix provides qualitative insights and simplified derivations. In recommendation, preference data consists of a positive–negative pair. The positive sample y_u^i is genuine, labeled by user interaction, while the negative sample y_u^j is sampled from the unobserved item set, which may not represent true negatives and can even be a false negative. We use y_u^k to denote a negative that is unobserved during training. Our heuristic analysis suggests that optimization on such preference pairs provides limited signal for the true positive and negative items (*i.e.,* y_u^i, y_u^k). In practice, the reward of the positive item may even counter-intuitively decrease (Rafailov et al., 2024). However, since the reward of the sampled negative y_u^j decreases more substantially, the reward gap still increases. This leads to reward hacking — the apparent reward margin grows, while recommendation performance is harmed.

We model the distribution $\pi_\theta(y \mid x_u) = \mathrm{softmax}(z)$ with $\{z^i\}_{i=1}^K$ being the logits of items and K the candidate set size. Taking DPO as an example (*e.g.,* other preference optimization methods such as SimPO are similar), optimization is performed over the training dataset $\mathcal{D}_{\text{PO}}$ by maximizing the pairwise preference likelihood, $\theta^* = \arg\max_\theta \mathbb{E}_{(x_u, y_u^i, y_u^j) \sim \mathcal{D}_{\text{PO}}} \log p(y_u^i \succ y_u^j \mid x_u)$.

By computing the gradient of the DPO loss with respect to the logits of y_u^i, y_u^j, y_u^k, we obtain:

$$\begin{cases} \nabla_{z^i} \log p(y_u^i \succ y_u^j \mid x_u) & = (1 - \sigma(r_u^i - r_u^j)), \\ \nabla_{z^j} \log p(y_u^i \succ y_u^j \mid x_u) & = -1 + \sigma(r_u^i - r_u^j), \\ \nabla_{z^k} \log p(y_u^i \succ y_u^j \mid x_u) & = 0. \end{cases} \tag{16}$$

where, for convenience, we define $r_u^t = \beta \log \frac{\pi_\theta(y_u^t \mid x_u)}{\pi_{\text{ref}}(y_u^t \mid x_u)}$ for $t \in \{1, 2, \ldots, K\}$ as the implicit reward of item y_u^t given the interaction history x_u, as computed by the policy model. We omit the additive constant $\beta \log Z(x_u)$ since it cancels out in pairwise differences.

Thus, the logit of a true negative y_u^k receives zero gradient, meaning that optimization completely ignores such items during training. The model is therefore driven to update only the logits of y_u^i and y_u^j during optimization. This reasoning also applies to other preference optimization methods such as SimPO when length normalization is not used.

Furthermore, the gradient of the reward with respect to model parameters can be expanded as

$$\nabla_\theta r_u^j = \sum_{t=1}^{K} \frac{\partial r_u^j}{\partial z^t} \nabla_\theta z^t, \quad \frac{\partial r_u^j}{\partial z^t} = \begin{cases} -\pi_\theta(y_u^t \mid x_u), & \text{if } y_u^t \neq y_u^j, \\ 1 - \pi_\theta(y_u^j \mid x_u), & \text{if } y_u^t = y_u^j. \end{cases} \tag{17}$$

Note that

$$1 - \pi_\theta(y_u^j \mid x_u) = \sum_{y^l \neq y_u^j} \pi_\theta(y^l \mid x_u) > \max_{y^l \neq y_u^j} \pi_\theta(y^l \mid x_u),$$

This inequality indicates that the gradient on z^j dominates that on any other logit. As a result, optimization primarily suppresses the sampled negative y_u^j, while the positive item y_u^i is not guaranteed to receive a corresponding increase in its reward. Consequently, the apparent reward gap between y_u^i and y_u^j may enlarge, even though the absolute reward of the true positive y_u^i does not improve—or may even decrease. This mismatch is the essence of reward hacking. This effect is analogous to the well-known sampling bias in implicit-feedback recommendation, where optimization tends to overfit sampled negatives rather than reinforcing positives.

The analysis above shows only that the logit of y_u^k does not influence the loss, offering a preliminary explanation of reward hacking. The above analysis highlights why sampled negatives dominate training dynamics while true negatives remain unsupervised, offering a preliminary explanation of reward hacking.

D More analysis for methods

D.1 S-DPO for reward hacking

Let $f(\varepsilon) \in [0, 1]$ denote the fraction of real-data pairs that fall inside an ε-insensitive region $\Omega_\varepsilon(p^{ik}, p^{ij})$. For S-DPO that samples t negatives per positive, the probability that a given positive p^{ik} is simultaneously trapped in the ε-insensitive regions of all t negatives is $f(\varepsilon)^t$ (hence $f(\varepsilon)^t \leq f(\varepsilon)$). Thus, increasing the number of sampled negatives reduces the chance of reward-hacking by exponential decay in t. However, this benefit comes at a large computational cost: sampling t negatives increases the training workload roughly by a factor of about $(t + 1)/2$ compared to standard pairwise training, which is often prohibitive in practice.

D.2 Additional Analysis under the Bradley-Terry Model

In this appendix, we present the detailed derivation of how pseudo-negatives reshape the optimization landscape under the Bradley-Terry (BT) model. We formally characterize the flat regions of p_u^{ik} with respect to p_u^{i0}, and show that misordered pairs $(y_u^k \succ y_u^i)$ never fall into the ε-insensitive region for any $\varepsilon < 1$. This guarantees that pseudo-negatives provide effective gradient propagation to unsampled negatives.

Specifically, since $r_u^i, r_u^j, r_u^k < r_u^0$, both p_u^{i0} and p_u^{k0} lie below 0.5, which is because

$$p_u^{i0} = p_{\mathrm{BT}}(r_u^i - r_u^0) < p_{\mathrm{BT}}(0) = 1/2,$$

similar for p_u^{k0}. This observation restricts the relevant analysis to the $[0, 0.5]$ region of the preference space. For a misordered pair $(y_u^k \succ y_u^i)$, we have $r_u^k > r_u^i$ and thus $p_u^{k0} > p_u^{i0}$, which places the pair above the diagonal in Figure 5. We can then show that p_u^{ik} never falls into the ε-insensitive region of p_u^{i0} for any $\varepsilon < 1$, guaranteeing non-vanishing optimization signals. In contrast, correctly ranked pairs $(y_u^i \succ y_u^k)$ require no correction and remain preserved.

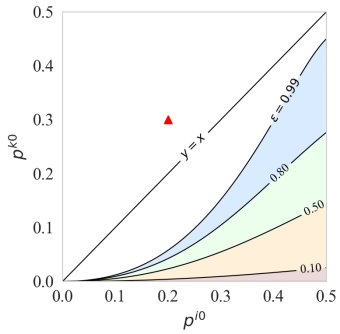


Figure 5: Insensitive region of p_u^{ik} with respect to p_u^{i0} under the BT model. Misordered pairs $(y_u^k \succ y_u^i)$ always lie outside the insensitive region, ensuring effective gradient updates.

In summary, the BT model analysis confirms that pseudo-negatives act as universal anchors: they prevent unsampled misordered pairs from being trapped in ε-insensitive regions and thus fundamentally mitigate reward hacking.

Table 2: Statistics of Datasets.

Dataset	LastFM	Goodreads	Steam
# Sequence	1220	6,031	11,938
# Item	4606	4,550	3,581
# Interaction	73,510	220,100	274,726

D.3 Pseudo code for SIRIUS

Algorithm 1 Training with Pseudo-Negative (SIRIUS)

Input: Recommendation data $\mathcal{D} = \{(u, \mathcal{H}_u, y_u^i, \mathcal{C}_u)\}$, model parameters θ
Output: trained recommendation policy $\pi_\theta(y \mid x_u)$
repeat
  Sample $(u, \mathcal{H}_u, y_u^i, \mathcal{C}_u) \sim \mathcal{D}$ ▷ user u, history $\mathcal{H}_u$, positive item y_u^i, candidate set $\mathcal{C}_u$
  Construct recommendation prompt x_u for user u, including task description and historical interactions $\mathcal{H}_u$
  Sample a negative item $y_u^j \sim \mathcal{C}_u \setminus \{y_u^i\}$ ▷ candidate negative from the same slate
  Compute an upper bound on the implicit reward: $N_u \geq \max_{y \in \mathcal{C}_u} r(x_u, y)$
  Choose a pseudo-negative reward $r_u^0 \in [N_u, \infty)$
  Introduce a virtual pseudo-negative item y_u^0 with reward $r(x_u, y_u^0) = r_u^0$
  Compute $p(y_u^i \succ y_u^n \mid x_u)$ for all $y_u^n \in \mathcal{N}_u$ ▷ where $\mathcal{N}_u = \{y_u^j, y_u^0\}$
  $\mathcal{L}_u = -\sum_{y_u^n \in \mathcal{N}_u} \log p(y_u^i \succ y_u^n \mid x_u)$
  Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_u$
until convergence

E Experiments Details

E.1 Datasets

Our experimental evaluation is conducted on three real-world datasets from different domains:

- LastFM (Cantador et al., 2011): A widely used dataset in the music recommendation domain, containing user interactions with various tracks, sourced from a well-known online music service.
- Goodreads[2]: Sourced from a popular online book community, this dataset includes user ratings and reviews of books.
- Steam (Kang & McAuley, 2018b): This dataset comprises user ratings and reviews for video games, collected from a leading digital distribution platform.

To ensure data quality and consistency, we apply specific filtering criteria for each dataset, following (Liao et al., 2024b). For LastFM, we maintain the titles as textual descriptions for each dataset. For Goodreads, only interactions with ratings of 5 or above are retained, and users with fewer than 20 interactions are excluded to ensure sufficient engagement. In the case of the Steam dataset, we randomly sample 30% of the available games and their corresponding interaction sequences to control the dataset scale.

Each dataset is subsequently partitioned into training, validation, and test sets in an 8:1:1 ratio, based on the chronological order of interaction timestamps, thereby mitigating the risk of data leakage. A sliding window approach with a window size of 11 is employed to extract sequential interaction data, where the final item in each sequence is designated as the target for the subsequent prediction task. The statistics of the datasets are summarized in Table 2.

[2] https://www.goodreads.com/

E.2 BASELINES

We compare SIRIUS against both traditional recommendation models and recent LLM-based methods.

For traditional recommendation models, we consider three widely adopted baselines:

- GRU4Rec (Hidasi et al., 2016), which utilizes recurrent neural networks (RNNs) for sequential recommendation;
- Caser (Tang & Wang, 2018), a convolutional model that captures sequential patterns;
- SASRec (Kang & McAuley, 2018a), an attention-based model designed to learn sequential dependencies.

For LLM-based recommendation approaches, we select five representative models from different paradigms:

- LLaMA2 (Touvron et al., 2023), which directly utilizes the vanilla LLaMA2-7B model to generate recommendations through zero-shot prompting without fine-tuning.
- ChatRec (Gao et al., 2023b), which generates recommendation lists by prompting an LLM without fine-tuning. [3]
- MoRec (Yuan et al., 2023), which enhances traditional recommendation systems by initializing item embeddings with textual representations encoded by an LLM. [4]
- TallRec (Bao et al., 2023b), which transforms user interaction sequences into textual prompts and fine-tunes large language models on domain-specific corpora to enhance recommendation quality.
- LLaRA (Liao et al., 2024b), which integrates traditional models into an LLM using a projector and curriculum-tunes the LLM with prompts containing hybrid representations.

For DPO-based recommendation approaches, we consider three representative models from distinct methodological frameworks:

- DPO (Rafailov et al., 2023), which formulates a closed-form solution for the reward model in RLHF and optimizes the preference model in an offline manner.
- SimPO (Meng et al., 2024), which eliminates the need for an explicit reference model by leveraging the average log probability of a sequence as the reward signal and enforces a target reward margin between chosen and rejected responses.
- S-DPO (Chen et al., 2024b), which extends DPO to the recommendation domain by randomly sampling multiple negative items as rejected responses, thereby improving recommendation accuracy.

E.3 IMPLEMENTATION DETAILS

E.3.1 IMPLEMENTATION

Following (Liao et al., 2024b; Yang et al., 2023), we implement conventional recommender baselines using the Adam optimizer with a learning rate of 1e-3. The embedding dimension is set to 64, and we explore a range of L2 regularization coefficients, selecting from [1e-3, 1e-4, 1e-5, 1e-6, 1e-7] through a grid search. For the LLM-based methods, we adapt their framework to output rankings of candidate items, training the models for a single epoch in each tuning stage, utilizing a warm-up strategy for the learning rate.

For DPO-based methods, including our approach, we conduct experiments using 4 A100 NVIDIA RTX GPUs. Our model leverages the widely adopted Llama-2-7B and Llama-3-8B-Instruct as the

[3] For ChatRec, we adopt GPT-4 as the LLM backbone.

[4] For MoRec, we follow the official implementation and employ SASRec as the recommender backbone and BERT as the text encoder.

backbone, fine-tuned with 32×8 LoRA (Hu et al., 2021) across both the SFT and preference alignment stages. The SFT phase is trained for up to five epochs, followed by preference alignment for 5 epochs on the all three datasets, respectively.

In constructing prompts, items are represented by their titles as textual features, and recommendations are grounded using the output probability distribution. For hyperparameter tuning, we search β within $\{1, 1.5, 2.0, 2.5, 3.0\}$. In the experiments with SimPO and SimPO w/o LN, we search γ within $\{0.4, 0.6, 0.8, 1.0\}$. The batch sizes are configured as 128 for both LastFM and Goodreads, and 256 for Steam.

In addition, we observe that although SimPO generally outperforms the DPO paradigm in recommendation quality, its training is unstable, with large variance across runs. We believe this instability arises because SimPO removes the reference model π_{ref} from the implicit reward function, which means it no longer incorporates the KL penalty in the original formulation (Eq. 2). As a result, the quality of training data has a much stronger impact on performance. Specifically, when data quality is poor (*e.g.,* when sampled negatives are not true negatives), SimPO lacks the weighting mechanism of DPO that downweighs such noisy samples. To address this, during training of SimPO w/o LN and SimPO w/o LN + SIRIUS we apply a reward-margin based filtering strategy. We discard samples where the reward margin between the negative sample y_u^j and the positive sample y_u^i exceeds γ, i.e., $r_u^j - r_u^i > \gamma$. This filtering improves training stability. However, we emphasize that this approach only stabilizes optimization and does not address the broader issue of reward hacking.

E.3.2 Evaluation

Due to the limited context window and slower inference speed of LLM-based approaches, our method is best suited for the fine-tuning stage in recommendation systems, where the objective is to rank a small set of candidate items based on user preferences. Following the setup in (Liao et al., 2024b), we conduct our primary experiments using 20 randomly selected non-interacted items as candidates. We adopt the widely used Hit Ratio (HR@1) metric to evaluate recommendation performance.

E.4 Additional Experiments and Analyses

To further validate the robustness, generality, and scalability of our method, we provide additional experiments along four dimensions: (i) expanded HR@K and NDCG@K metrics; (ii) upgrading the backbone from LLaMA-2 to LLaMA-3; (iii) enlarging the candidate set size from 20 to 100; and (iv) an ablation study on the pseudo-negative reward design for SimPO w/o LN + SIRIUS. These results reinforce the main findings presented in the paper and offer additional insights into the behavior of SIRIUS under different configurations.

E.4.1 Expanded HR@K and NDCG@K Results

We report the complete HR@K and NDCG@K metrics across all datasets and across different evaluation cutoffs for experiment setting same as Table 1. These full results, as shown in Table 3, corroborate the observations made in the main paper.

E.4.2 Backbone Upgrade: LLaMA-2 to LLaMA-3

To examine whether the effect of SIRIUS transfers to stronger backbones, we upgrade the model from **LLaMA-2-7B** to **LLaMA-3-8B-Instruct** for *SimPO w/o LN* and *SimPO w/o LN + SIRIUS* settings on the LastFM dataset. As shown in Table 4, the results show that the improvements brought by SIRIUS persist under LLaMA-3: optimization remains stable, reward drift is further reduced, and both HR@K and NDCG@K metrics continue to improve. This confirms that SIRIUS retains its stabilizing effect even with a more capable backbone.

E.4.3 Increasing the Candidate Set Size from 20 to 100

We further test robustness under harder ranking conditions by enlarging the candidate set from 20 to 100 on LastFM. A larger candidate pool greatly increases the number of unsampled negatives, amplifying reward miscalibration and making over-optimization more likely. Consistent with this

Table 3: Performance comparison across different datasets and baselines.

Classification	Method		LastFM		Goodreads		Steam	
			HitRatio	NDCG	HitRatio	NDCG	HitRatio	NDCG
Traditional	GRU	@1	0.2666	0.2666	0.3867	0.3867	0.4120	0.4120
		@5	0.3770	0.2712	0.5049	0.3817	0.5221	0.4210
		@10	0.5656	0.2895	0.6857	0.4124	0.7016	0.4342
	Caser	@1	0.2410	0.2410	0.4174	0.4174	0.4288	0.4288
		@5	0.3525	0.2671	0.5307	0.4384	0.5363	0.4526
		@10	0.5000	0.2902	0.6672	0.4574	0.6869	0.4772
	SASRec	@1	0.2492	0.2492	0.3581	0.3581	0.4037	0.4037
		@5	0.3443	0.2705	0.4448	0.3717	0.4896	0.4341
		@10	0.5082	0.3216	0.6254	0.4383	0.6687	0.4745
Preference Method	DPO	@1	0.6393	0.6393	0.6462	0.6462	0.7333	0.7333
		@5	0.8906	0.7764	0.8937	0.7763	0.8523	0.7883
		@10	0.9503	0.7959	0.9568	0.7969	0.9063	0.8154
	SimPO w/o LN	@1	0.6633	0.6633	0.7010	0.7010	0.8002	0.8002
		@5	0.8926	0.7850	0.9037	0.7598	0.9460	0.8979
		@10	0.9555	0.8056	0.9684	0.7843	**0.9772**	0.9079
	S-DPO-3	@1	0.6493	0.6493	0.6744	0.6744	0.7712	0.7712
		@5	0.8982	0.7831	0.9203	0.8057	0.8714	0.8164
		@10	0.9531	0.8010	0.9734	0.8232	0.9272	0.8371
+SIRIUS	DPO	@1	0.6593	0.6593	0.6744	0.6744	0.7757	0.7757
		@5	0.8906	0.7770	0.9070	0.8062	0.8848	0.8088
		@10	0.9519	0.7969	0.9734	0.8281	0.9322	0.8501
	SimPO w/o LN	@1	**0.7315**	**0.7315**	**0.7857**	**0.7857**	**0.8550**	**0.8550**
		@5	**0.9154**	**0.8299**	**0.9468**	**0.8793**	**0.9469**	**0.9073**
		@10	**0.9639**	**0.8456**	**0.9801**	**0.8904**	**0.9772**	**0.9172**

Table 4: Results on LastFM using the LLaMA3-8B-Instruct backbone.

Classification	Method	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
LLaMA3-8B-Instruct	SimPO w/o LN	0.713	0.879	0.741	0.962	0.769
	SimPO w/o LN + **SIRIUS**	**0.764**	**0.883**	**0.756**	0.952	**0.778**

Table 5: Results on LastFM with a candidate set size of 100.

Classification	Method	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
Traditional	GRU	0.0246	0.0902	0.0555	0.1475	0.0740
	Caser	0.0410	0.0902	0.0637	0.1393	0.0800
	SASRec	0.0204	0.1066	0.0566	0.1639	0.0742
LLM-based	TALLRec	0.0970	0.3355	0.2167	0.5507	0.2858
Preference Method	SimPO w/o LN	0.1800	0.4293	0.3085	0.6212	0.3700
	SimPO w/o LN + **SIRIUS**	**0.2665**	**0.5323**	**0.4038**	**0.7030**	**0.4586**

prediction, *SimPO w/o LN + SIRIUS* exhibits substantially larger gains over *SimPO w/o LN* in this setting—around a 50% relative improvement, compared with only about 10% under 20 candidates. Full results are shown in Table 5, and these results confirm that the virtual-anchor mechanism becomes even more beneficial as the evaluation scenario becomes more adversarial.

E.4.4 Ablation on the Pseudo-Negative Reward

We study the sensitivity of *SimPO w/o LN + SIRIUS* to the pseudo-negative anchor r_u^0 by testing $r_u^0 \in \{0, 1, 10\}$. The results in Table 6 show that the anchor magnitude has only minor influence on overall performance: all three settings yield nearly identical learning curves on LastFM and Goodreads, with small differences attributable to optimization dynamics. Larger anchors occasionally speed up early-stage convergence, reflecting stronger separation between positives and unsampled negatives. On Steam, however, overly large anchors can destabilize training when pushed beyond convergence, leading to collapse. Overall, the virtual-anchor mechanism is robust to reasonable choices of r_u^0.

Table 6: Ablation on the pseudo-negative reward r_u^0 under LLaMA2-7B SimPO w/o LN + SIRIUS.

Dataset	r_u^0	epoch 0	epoch 1	epoch 2	epoch 3	epoch 4	epoch 5
LastFM	0	0.6239	0.6589	0.6966	0.7242	0.7302	0.7315
	1	0.6239	0.6581	0.6950	0.7174	0.7311	0.7263
	10	0.6239	0.6637	0.6938	0.7226	0.7323	0.7255
Goodreads	0	0.6102	0.6827	0.7259	0.7541	0.7740	0.7857
	1	0.6102	0.6910	0.7342	0.7674	0.7791	0.7890
	10	0.6102	0.6844	0.7508	0.7542	0.7741	0.7807
Steam	0	0.6878	0.7715	0.8111	0.8465	0.8550	0.8465
	1	0.6878	0.7864	0.8196	0.8272	0.8575	0.8541
	10	0.6878	0.7993	0.8465	0.8651	0.0658	0.0000

F USE OF LARGE LANGUAGE MODELS

In the preparation of this paper, we made limited use of large language models (LLMs), such as ChatGPT, exclusively for writing assistance. Specifically, LLMs were employed to:

- Polishing grammar, clarity, and style;
- Suggesting alternative phrasings to improve readability.

No part of the research design, experiments, data analysis, or results interpretation was generated by LLMs. All technical content, ideas, and contributions originated solely from the authors. The use of LLMs did not influence the research methodology, results, or conclusions of this work.