MSL: Not All Tokens Are What You Need for Tuning LLM as a Recommender

Bohao Wang[*‡§]
bohao.wang@zju.edu.cn
Zhejiang University
Hangzhou, China

Feng Liu
liufeng4hit@gmail.com
OPPO Research Institute
Shenzhen, China

Jiawei Chen[†‡§¶]
sleepyhunt@zju.edu.cn
Zhejiang University
Hangzhou, China

Xingyu Lou
louxingyu@oppo.com
OPPO Research Institute
Shenzhen, China

Changwang Zhang
changwangzhang@foxmail.com
OPPO Research Institute
Shenzhen, China

Jun Wang
junwang.lu@gmail.com
OPPO Research Institute
Shenzhen, China

Yuegang Sun
bulutuo@i-i.ai
Intelligence Indeed
Hangzhou, China

Yan Feng[‡§]
fengyan@zju.edu.cn
Zhejiang University
Hangzhou, China

Chun Chen[‡§]
chenc@zju.edu.cn
Zhejiang University
Hangzhou, China

Can Wang[‡¶]
wcan@zju.edu.cn
Zhejiang University
Hangzhou, China

Abstract

Large language models (LLMs), known for their comprehension capabilities and extensive knowledge, have been increasingly applied to recommendation systems (RS). Given the fundamental gap between the mechanism of LLMs and the requirement of RS, researchers have focused on fine-tuning LLMs with recommendation-specific data to enhance their performance. Language Modeling Loss (LML), originally designed for language generation tasks, is commonly adopted. However, we identify two critical limitations of LML: 1) it exhibits significant divergence from the recommendation objective; 2) it erroneously treats all fictitious item descriptions as negative samples, introducing misleading training signals.

To address these limitations, we propose a novel **Masked Softmax Loss (MSL)** tailored for fine-tuning LLMs on recommendation. MSL improves LML by identifying and masking invalid tokens that could lead to fictitious item descriptions during loss computation. This strategy can effectively avoid the interference from erroneous negative signals and ensure well alignment with the recommendation objective supported by theoretical guarantees. During implementation, we identify a potential challenge related to gradient vanishing of MSL. To overcome this, we further introduce the temperature coefficient and propose an **Adaptive Temperature Strategy (ATS)** that adaptively adjusts the temperature without requiring extensive hyperparameter tuning. Extensive experiments conducted on four public datasets further validate the effectiveness of MSL, achieving an average improvement of 42.24% in NDCG@10. The code is available at https://github.com/WANGBohaO-jpg/MSL.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Sequential Recommendation; Large Language Model; Loss Function

ACM Reference Format:
Bohao Wang, Feng Liu, Jiawei Chen, Xingyu Lou, Changwang Zhang, Jun Wang, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. 2025. MSL: Not All Tokens Are What You Need for Tuning LLM as a Recommender. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730041

[*]This work was done during an internship at OPPO Research Institute.
[†]Corresponding author.
[‡]State Key Laboratory of Blockchain and Data Security, Zhejiang University.
[§]College of Computer Science and Technology, Zhejiang University.
[¶]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730041

1 Introduction

Large Language Models (LLMs) have showcased exceptional capabilities in content comprehension and leveraging extensive knowledge, thereby catalyzing a revolution in artificial intelligence [1].

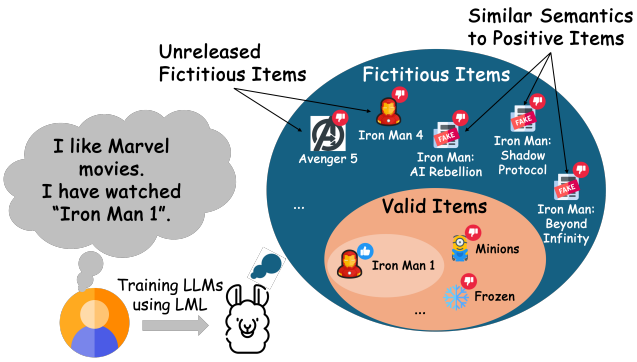

Figure 1: Language modeling loss can erroneously treat all fictional items as negative items. However, some of these items may exhibit semantic similarities to positive items (*e.g.,* "Iron Man: AI Rebellion"). Additionally, certain items might are fictional simply because they have not yet been released (*e.g.,* "Iron Man 4"). These cases represent incorrect negative signals.

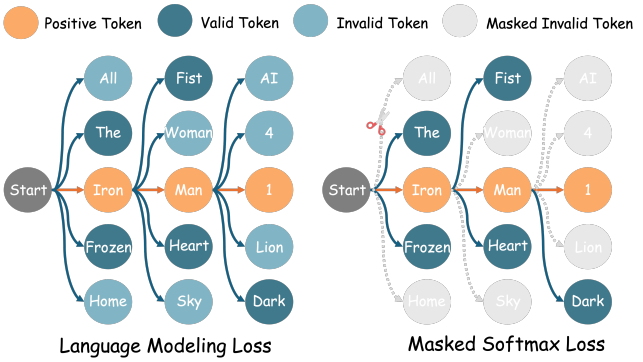

Figure 2: The schematic diagram illustrates Language Modeling Loss (LML) and Masked Softmax Loss (MSL).

Recently, LLMs have been extensively applied in the field of Recommender Systems (RS) [63]. A prominent strategy involves directly leveraging LLMs as recommenders — organizing users' historical interactions as language prompts and instructing LLMs to deduce users' preferences for predicting future interactions [16, 19, 35, 55, 59]. This paradigm has demonstrated enhanced few-shot ability [19, 55], generalization [24], explainability [16], and impressive recommendation performance [3].

To fully unlock the potential of LLMs in recommendation, supervised fine-tuning (SFT) is commonly applied for LLM-based recommenders [2, 3, 17, 23, 31, 73]. These methods typically structure users' historical interactions as prompts, paired with descriptions of positive items as target responses, and fine-tune LLMs using a Language Modeling Loss (LML) [40]. This loss, inherited from language generation tasks and expressed as a token-wise softmax loss, augments the probability (*i.e.,* logits) of tokens representing positive items while penalizing the logits of other generated content. However, we argue that this objective has significant limitations in the recommendation scenario:

- **Significant Divergence from the Recommendation Objective:** RS aims for personalized ranking performance (*e.g.,* higher NDCG), prioritizing positive items over negative ones. LML deviates significantly from this ranking objective. Through extensive theoretical and empirical analyses, we find that optimizing LML primarily focuses on generating valid item descriptions that exist in the system, while providing limited guidance to help LLMs differentiate positive items from negative ones. This deviation significantly hinders the effectiveness of LML in recommendations.
- **Improper Negative Signals:** Language modeling loss implicitly considers all other generated item descriptions as negative, including valid negative items that the user has not interacted with and fictitious items that do not exist in the RS. This treatment is flawed as it is improper to hypothesize that the user dislikes these fictitious items. In fact, some fictitious items may share similar semantic with the positive items, whose contents may be favored by users. As shown in Figure 1, a typical fan of the Marvel Universe who has watched "Iron Man 1" would likely enjoy a fictitious movie such as "Iron Man: AI Rebellion". Blindly treating all such fictitious items as negative could confuse the LLM in capturing user preferences.

To tackle these limitations, we introduce a novel loss function, termed **Masked Softmax Loss (MSL)**, specifically designed for fine-tuning LLMs as recommenders. MSL employs a masking mechanism that prevents penalization of fictitious item descriptions. As illustrated in Figure 2, this mechanism can be implemented efficiently at the token level by masking the invalid tokens in the softmax calculation that correspond fictitious items. Our theoretical analyses further demonstrate the close connection of MSL with the ranking objective, serving as a tight upper bound of the NDCG metrics.

Despite its theoretical advantages, MSL may encounter gradient vanishing issues during practical application. This arises from the reduced number of tokens in the softmax denominator, which can lead to particularly small gradients and loss values. A simple and effective strategy to address this is the introduction of an additional hyperparameter, temperature, in the softmax function to modulate its values. While effective, this approach requires tedious and time-consuming hyperparameter tuning, which is unsatisfactory for LLM-based recommendations. To address this challenge, we propose an Adaptive Temperature Strategy (ATS). By examining the gradient of MSL and the role of temperature, we derive an adaptive configuration based on the average number of valid tokens in the dataset. This strategy effectively mitigates gradient vanishing in MSL without requiring extensive hyperparameter tuning.

Lastly, in terms of aligning LLM with the ranking objective, the most relevant work is the recently proposed S-DPO [9], which integrates Direct Preference Optimization (DPO) [41] in LLM-based recommendation. However, S-DPO exhibits several limitations: 1) Suboptimal Performance: S-DPO still relies on LML to fine-tune the model and considers the fine-tuned LLM as a reference model for optimization. Given the inherent limitations of LML, the effectiveness of S-DPO is compromised. 2) Unstable Results: S-DPO

Table 1: Prompt templates for implementing recommendation tasks (using Toy dataset as an example)

Instruction Input	
Instruction:	Given a list of toys the user has played before, please recommend a new toy that the user likes to the user.
Input:	The user has played the following toys before: "LeapFrog Discovery Ball", "Plush Elmo Knows Your Name", "Blokus Game", ...
Instruction Output	
Output:	"MindWare Q-Ba-Maze Cool Colors"

requires negative item sampling, which can lead to training instability, especially in fine-tuning tasks with limited epochs. 3) High Computational Cost: S-DPO requires more training instances and epochs, resulting in significantly longer training times (approximately 4 times) compared to MSL.

In summary, this work makes the following contributions:

- We propose a novel loss function, Masked Softmax Loss (MSL), specifically tailored for fine-tuning large language models to effectively align with recommendation objectives.
- We address the potential gradient vanishing issue of MSL by developing an adaptive temperature strategy that mitigates this issue without requiring hyperparameter tuning.
- Extensive experiments on four real-world datasets demonstrate that the proposed MSL outperforms LML by a large margin (42.24% on average in NDCG@10).

2 LLM-based Recommendation

Referring to recent work [2, 3, 31, 32, 37, 73], this work also focuses on sequential recommendation, which holds notable practical significance by considering the temporal order of user behavior. Given a sequential recommender system with a user set $\mathcal{U}$ and an item set $\mathcal{V}$, let user's historical interactions be denoted as $S = \{s_1, s_2, ...\}$, where $s_i \in \mathcal{V}$ denotes the i-th interacted item in the sequence. The objective of sequential recommendation is to infer user preferences from S and retrieve the positive item p that the user will interact with next. This task is often conceptualized as a ranking problem, aiming to position the positive item p higher in the ranking list. Consequently, ranking metrics such as NDCG are frequently adopted to evaluate recommendation performance.

Given the remarkable success of large language models (LLMs) across various domains [38, 49, 61], integrating LLMs into recommendation systems has been extensively explored [63]. A prominent strategy is to directly leverage powerful LLMs as recommenders. As shown in Table 1, this paradigm organizes users' historical interactions as language prompts x, typically consisting of the descriptions (*e.g.,* titles) of the items in S and the description of the recommendation task. This prompt is then used to instruct the LLMs to predict the item (descriptions) that the user is most likely to interact with.

Since LLMs are typically not pre-trained on recommendation data, supervised fine-tuning is necessary to align LLMs with the recommendation task. This strategy pairs the prompts x and the description of the target positive item y^p as a training instance (x, y^p), and optimizes LLMs with the following **Language Modeling Loss (LML)**:

$$\mathcal{L}_{LML}(x, y^p; \theta) = -\log P_\theta\left(y^p \mid x\right) = \sum_{t=1}^{|y^p|} -\log P_\theta\left(y_t^p \mid x, y_{<t}^p\right) = \sum_{t=1}^{|y^p|} -\log \frac{\exp(f_\theta(y_t^p|x, y_{<t}^p))}{\sum_{z\in\mathcal{Z}} \exp(f_\theta(z|x, y_{<t}^p))} \tag{1}$$

where y_t^p denotes the t-th token of the positive item description y^p, and $y_{<t}^p$ represents the token sequence preceding y_t^p. The set $\mathcal{Z}$ corresponds to the entire vocabulary of tokens in the LLM, and $f_\theta(y_t^p \mid x, y_{<t}^p)$ denotes the logit of the token y_t^p predicted by LLMs, where θ denotes the parameters of LLMs. For simplicity, we use $f_\theta(y_t^p)$ *(or* $f_\theta(z)$*)* to represent $f_\theta(y_t^p \mid x, y_{<t}^p)$ *(or* $f_\theta(z \mid x, y_{<t}^p)$*)*, and $P_\theta(y_t^p)$ to denote $P_\theta(y_t^p \mid x, y_{<t}^p)$.

The language modeling loss is directly inherited from language generation tasks, aiming to maximize the probability of the descriptions of positive items over the whole generative content space. It can be expressed in a token-wise manner with softmax loss, which augments the logits of the tokens representing positive items (numerator), while decreasing the logits of the other tokens in the vocabulary (denominator).

The notation table is presented in Table 2.

Table 2: Notations in the paper.

Notations	Descriptions
$\mathcal{U}$	user set
$\mathcal{V}$	item set
S	the user historical interaction sequence
p	the positive item of the sequence S
x	the input prompt of the sequence S
y^v	the description of the item v
y_t^v	t-th token of y^v
$\mathcal{Z}$	the vocabulary of LLM
$\mathcal{Z}_{valid}(y_{<t}^v)$	valid tokens for a given prefix $y_{<t}^v$
θ	the model parameter
f_θ	logits output by the model
$P_\theta(y_t^v)$	the probability of y_t^v over $\mathcal{Z}$
$P_\theta^{valid}(y_t^v)$	the probability of y_t^v over $\mathcal{Z}_{valid}(y_{<t}^v)$
$\mathcal{L}_{LML}$	language modeling loss
$\mathcal{L}_{MSL}$	masked softmax loss

3 Analyses on Language Modeling Loss

While language modeling loss is commonly used for fine-tuning LLMs as recommenders, we argue that it still suffers from the following limitations:

Limitation 1: Significant Divergence from the Recommendation Objective. Recommender systems aim to retrieve positive items from the valid item set in the system. In contrast, LML aims to

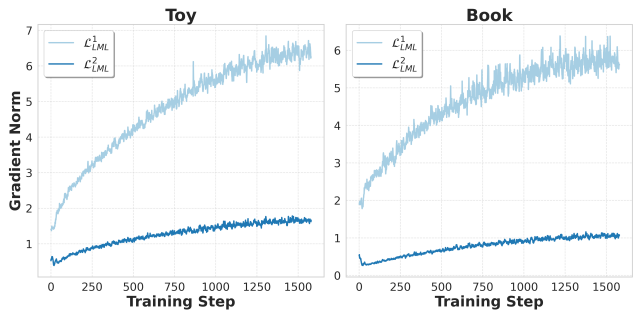

Figure 3: The gradient norms of the two components of language modeling loss during the model training.

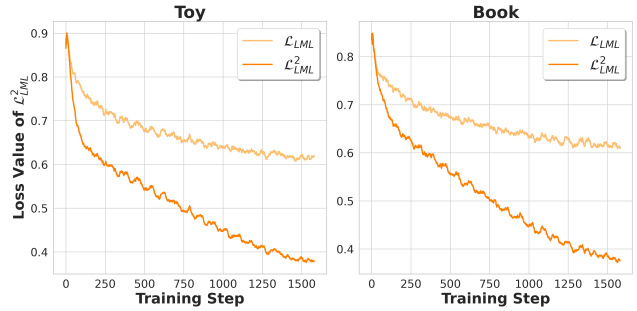

Figure 4: The variation of $\mathcal{L}^2_{LML}$ during the training when optimizing with $\mathcal{L}_{LML}$ vs. $\mathcal{L}^2_{LML}$.

retrieve positive item descriptions from the entire language space that LLMs could generate. It's important to note that the language space contains descriptions of both valid items in the system and fictitious items imagined by LLMs. This causes the objective of LML to deviate significantly from the recommendation objective.

To better understand this deviation, we can decompose LML into two components:

$$\mathcal{L}_{LML}(x, y^p; \theta) = \mathcal{L}^1_{LML}(x, y^p; \theta) + \mathcal{L}^2_{LML}(x, y^p; \theta) \tag{2}$$

$$\mathcal{L}^1_{LML}(x, y^p; \theta) = \sum_{t=1}^{|y^p|} \underbrace{-\log \frac{\sum_{z \in \mathcal{Z}_{valid}(y^p_{<t})} \exp(f_\theta(z))}{\sum_{z \in \mathcal{Z}} \exp(f_\theta(v))}}_{\text{Lifting valid items over invalid items}} \tag{3}$$

$$\mathcal{L}^2_{LML}(x, y^p; \theta) = \sum_{t=1}^{|y^p|} \underbrace{-\log \frac{\exp(f_\theta(y^p_t))}{\sum_{z \in \mathcal{Z}_{valid}(y^p_{<t})} \exp(f_\theta(z))}}_{\text{Lifting the positive items over negative items}} \tag{4}$$

where $z \in \mathcal{Z}_{valid}(y^p_{<t})$ denotes a valid token, ensuring that the combined language contents $[y^p_{<t}, z]$ can be a prefix of any valid item description. A similar definition applies to invalid tokens, $z \notin \mathcal{Z}_{valid}(y^p_{<t})$, which would make the generated contents fall outside the scope of valid items' descriptions. For simplicity, we use $\mathcal{Z}_{valid}$ to denote $\mathcal{Z}_{valid}(y^p_{<t})$ in the following text.

Table 3: The deviation of NDCG@10 under multiple random seeds.

Method	Toy	Book
S-DPO	0.0219 ± 0.0036(16.4%)	0.0124 ± 0.0019(15.3%)
MSL	0.0294 ± 0.0006(2.0%)	0.0175 ± 0.0005(2.9%)

Language modeling loss has two-fold effects: 1) $\mathcal{L}^1_{LML}$ increases the logits of valid tokens while penalizing the invalid tokens. This component would lift the probability of valid items over fictitious items, guiding the LLMs towards outputting a valid item description. 2) $\mathcal{L}^2_{LML}$ increases the logits of the positive token (*i.e.,* y^p_t) and penalizes the logits of negative tokens (*i.e.,* $\mathcal{Z}_{valid} \setminus y^p_t$). This component would lift the probability of positive items over negative items, which aligns with the recommendation objective. We will also prove the close theoretical connection of this component with the NDCG metrics in the next section (lemma 1).

The above decomposition illustrates the differences and connections between LML and the recommendation objective. While the recommendation objective serves as one component of LML, we empirically find that optimizing LML is ineffective, as the gradient is dominated by $\mathcal{L}^1_{LML}$. Figure 3 illustrates this point, showing the norm of the gradient from two components on typical datasets Toy and Book. It can be observed that $\mathcal{L}^1_{LML}$ exerts an overwhelming effect on the training, hindering the convergence of $\mathcal{L}^2_{LML}$. To further demonstrate this point, we conduct another experiment as shown in Figure 4, where we visualize the training loss of $\mathcal{L}^2_{LML}$ when we optimize $\mathcal{L}_{LML}$ or $\mathcal{L}^2_{LML}$ only for comparison. As can be seen, the training loss when we directly optimize $\mathcal{L}^2_{LML}$ decreases quickly, while the loss drop under optimizing $\mathcal{L}_{LML}$ seems hindered. These analyses demonstrate the ineffectiveness of leveraging LML in improving recommendation performance.

Limitation 2: Improper Negative Signals. From Eq.(2), we find that LML penalizes the logits of invalid tokens, implicitly considering all fictitious items that are not exist on the system as negative items. However, this treatment is flawed as it is improper to hypothesize that the user dislikes these fictitious items. In fact, some fictitious items may share semantic similarities with positive items and could potentially align with user preferences. To illustrate this issue, consider the example of a typical fan of the Marvel Universe who enjoys the movie "Iron Man 1" as shown in Figure 1. Some of the fictitious items may share semantic similarities with the positive items and may be favored by users (*e.g.,* "Iron Man: AI Rebellion"). Additionally, certain items are fictional simply because they have not yet been released (*e.g.,* "Iron Man 4"). As such, blindly treating all such fictitious items as negative could confuse the LLM, giving incorrect signals for capturing user preference.

Analyses on S-DPO. While S-DPO [9] leverages direct preference optimization to enhance LLM-based recommendation, it still suffers from the following limitations:

- **Suboptimal Performance.** S-DPO can not address the aforementioned limitations inherent in LML. S-DPO still relies LML to fine-tune LLMs, which would be utilized as a reference model for further DPO optimization. Given the inherent limitations

of LML, the effectiveness of S-DPO is compromised (*cf.* Section 5.2).

- **Unstable Performance.** S-DPO relies on sampling negative items to establish the ranking relationship between positive and negative items. However, this sampling process incurs performance instability. To evaluate this, we train the model using multiple random seeds and calculated the deviation of NDCG@10. As shown in Table 3, S-DPO exhibits significantly higher deviation compared to our proposed MSL, with performance losses reaching up to 16.4% on the Toy dataset and 15.3% on the Book dataset. This highlights the instability of S-DPO's performance.
- **High Computational Cost.** S-DPO requires further fine-tuning on the reference model, which entails additional training epochs. Furthermore, the inclusion of extra negative items substantially increases the data size. These factors contribute to its inefficiency. Empirically, S-DPO requires nearly four times the runtime of MSL and LML (*cf.* Section 5.4).

4 Methodology

In this section, we first detail the proposed Masked Softmax Loss (MSL) to address the limitations of language modeling loss (Subsection 4.1). We then highlight the potential gradient vanishing challenge in MSL and propose the Adaptive Temperature Strategy to tackle this issue (Subsection 4.2). The schematic diagrams of the MSL and LML methods are shown in Figure 2.

4.1 Masked Softmax Loss

The above analyses reveal that the limitations of LML primarily lie in the penalization of invalid tokens — it not only causes the loss to deviate from the recommendation objective but also introduces improper negative signals. To address this, a straightforward approach is to mask the invalid tokens in LML, *i.e.,* directly leverage the second component of LML to optimize LLMs. Formally, the Masked Softmax Loss is formulated as follows:

$$\mathcal{L}_{MSL}(x, y^p; \theta) = \sum_{t=1}^{|y^p|} -\log \frac{\exp(f_\theta(y_t^p))}{\sum_{z \in \mathcal{Z}_{valid}} \exp(f_\theta(z))} \tag{5}$$

This simple strategy effectively addresses the limitations by eliminating the penalization of invalid tokens. One might be concerned that this strategy could increase the risk of hallucination [21], where LLMs generate fictitious item descriptions during the inference stage. This concern can be easily mitigated by employing constrained beam search during generation [11]. Specifically, when choosing or sampling the next token in beam search, the selection can be restricted to valid tokens rather than the entire vocabulary. Such strategy ensures that the generated content corresponds to a valid item in the system, effectively mitigating the hallucination issue.

Overall, MSL possesses the following desirable properties:

Alignment with the Recommendation Objective. Intuitively, masking invalid tokens guide the model to focus more on differentiating positive items from negative ones. In fact, we have the following lemma establishing the theoretical connections between MSL and NDCG:

Lemma 1. *Considering a LLM-based RS that leverages the scores* $P_\theta^{valid}(y^v|x) = \prod_{t=1}^{|y^v|} P_\theta^{valid}(y_t^v|x, y_{<t}^v)$ *for ranking items, where*

$$P_\theta^{valid}(y_t^v|x, y_{<t}^v) = \frac{\exp(f_\theta(y_t^v))}{\sum_{z \in \mathcal{Z}_{valid}(y_{<t}^v)} \exp(f_\theta(z))}$$

represents the probability of the token y_t^v *within the valid token set* $\mathcal{Z}_{valid}(y_{<t}^v)$*, optimizing* $\mathcal{L}_{MSL}(x, y^p; \theta)$ *serves as a tighter upper bound of* $-logNDCG(S)$ *compared with* $\mathcal{L}_{LML}(x, y^p; \theta)$*,* i.e.,

$$-logNDCG(S) \le \mathcal{L}_{MSL}(x, y^p; \theta) \le \mathcal{L}_{LML}(x, y^p; \theta)$$

The proof is presented in the appendix. Note that the premise of ranking items based on $P_\theta^{valid}(y^v|x)$ is naturally satisfied when we mask invalid tokens during generation using constrained beam search with a large beam size. This lemma demonstrates that MSL is well-aligned with the recommendation objective and provides a tighter upper bound for optimizing NDCG compared to LML. $\mathcal{L}_{LML}^1$ in LML is redundant for NDCG optimization and may even introduce interference as previously discussed. Consequently, MSL is theoretically anticipated to achieve superior performance.

Ease of Implementation. Our MSL is simple, easily implemented, and can serve as a suitable surrogate for LML with minimal code revisions. The main implementation complexity lies in identifying valid tokens. In fact, this can be easily achieved by using a trie tree (*a.k.a.* a prefix tree) [4]. We can utilize existing packages of *marisa-trie* (with only 3 lines of codes) to construct the trie tree from all item descriptions during the pre-processing stage and calculate the masking matrix. Subsequently, we can revise the LML to MSL by simply applying the masking matrix. MSL can be seamlessly integrated into various existing LLM-based recommendation methods, including the recently proposed BIGRec [2], LLaRA [31], A-LLM [23], and consistently yield improvements (*cf.* section 5.2).

Efficiency. The primary computational challenge lies in constructing the trie tree and the masking matrix. However, this process is highly efficient, with a time and memory complexity of $O(|\mathcal{V}|\hat{l})$, where $|\mathcal{V}|$ denotes the number of items in the system, and $\hat{l}$ represents the average token length of the item description. Empirically, the Trie tree construction for all datasets is completed in under one second. Furthermore, MSL improves efficiency by excluding invalid tokens from the loss calculation (*cf.* section 5.4).

4.2 Adaptive Temperature Strategy

4.2.1 Potential Gradient Vanishing Issue. Despite the theoretical advantages of MSL, it may encounter gradient vanishing issues in practical applications, challenging its effectiveness. To illustrate this effect, the gradient of MSL over each sample can be expressed as follows:

$$\nabla_\theta \mathcal{L}_{MSL}(x, y^p; \theta) = -\sum_{t=1}^{|y^p|} w(y_t^p) g(y_t^p, \theta) \tag{6}$$

where

$$w(y_t^p) = 1 - P_\theta^{valid}(y_t^p \mid x, y_{<t}^p) \tag{7}$$

$$g(y_t^p, \theta) = \nabla_\theta f_\theta(y_t^p) - \frac{\sum_{z \in \mathcal{Z}'} \exp(f_\theta(z)) \nabla_\theta f_\theta(z)}{\sum_{z \in \mathcal{Z}'} \exp(f_\theta(z))} \tag{8}$$

$\mathcal{Z}' = \mathcal{Z}_{valid} \setminus \{y_t^p\}$ represents the set of negative tokens. For simplicity, let $P_\theta^{valid}(y_t^p)$ represent $P_\theta^{valid}(y_t^p \mid x, y_{<t}^p)$.

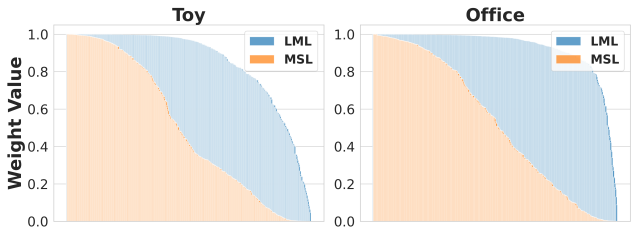

Figure 5: The weight value $w(y_t^p)$ distribution of a batch of samples for MSL and LML. The weight are sorted in descending order.

As observed, the magnitude of gradient is influenced by the weight $w(y_t^p)$. The gradient vanishing phenomenon can be attributed to a reduced number of terms in the denominator of $P_\theta^{valid}(y_t^p)$ as compared with $P_\theta(y_t^p)$, which naturally increases $P_\theta^{valid}(y_t^p)$, thereby decreasing the weight $w(y_t^p)$. This reduction can even cause the gradient to approach zero, particularly because the logits of positive tokens are often larger than those of other valid tokens[1].

Figure 5 presents the weight values of a batch of samples for both LML and MSL. When the valid token mask is applied to MSL, the weight values of all samples are substantially reduced, with some values even nearing zero. Importantly, these tokens are often crucial, as they are typically located among the first few tokens in the response and play a pivotal role in training. Empirical analysis on the Office dataset underscores this point: 61% of samples with weight values below 0.1 are concentrated within the first three tokens of the item. Similar patterns are observed across other datasets.

4.2.2 The Introduction of Temperature. To tackle this issue, we have found that the introduction of a temperature τ can effectively address this problem:

$$\mathcal{L}_{MSL}(x, y^p; \theta) = \sum_{t=1}^{|y^p|} -\tau \log \frac{\exp(f_\theta(y_t^p)/\tau)}{\sum_{z \in \mathcal{Z}_{valid}} \exp(f_\theta(z)/\tau))} \tag{9}$$

where the weight $w(y_t^p)$ can be written as:

$$w(y_t^p) = 1 - P_\theta^{valid}(y_t^p) = 1 - \frac{\exp(f_\theta(y_t^p)/\tau)}{\sum_{z \in \mathcal{Z}_{valid}} \exp(f_\theta(z)/\tau)} \tag{10}$$

The introduction of temperature can modulate the magnitude of the gradient. Considering that the logits of positive tokens are typically larger than those of other tokens, an increase in τ would relatively reduce the value of $P_\theta^{valid}(y_t^p)$, increasing $w(y_t^p)$; Figure 6 highlights the impact of incorporating temperature, which leads to a significant improvement in performance. Conversely, alternative approaches, such as adjusting the learning rate or introducing a balancing coefficient for negative tokens, fail to yield satisfactory results. The empirical evidence supporting these findings will be presented in Section 5.3.

[1]This assumption is reasonable, as the optimization process tends to increase the logits of positive tokens while decreasing those of negative tokens.

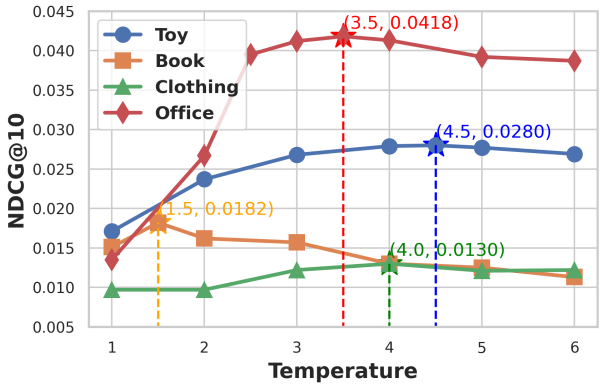

Figure 6: The performance of MSL across different temperature τ.

4.2.3 Adaptive Temperature Strategy. Despite its effectiveness, the introduction of τ incurs another hyperparameter tuning challenge. Given that the average number of valid tokens varies across different datasets, the optimal value of τ naturally evolves. For example, the optimal τ on dataset Book is 1.5, while it is 4.5 on Toy as shown in Figure 6. Transferring the optimal τ from one dataset to another without adjustment can lead to significant performance drops. This necessitates extensive hyperparameter tuning of τ, which can be particularly time-consuming, especially for heavy LLM-based recommenders.

To address this, inspired by recent studies on temperature [7], we develop an Adaptive Temperature Strategy (ATS) for MSL. This strategy dynamically and adaptively adjusts τ to ensure that $P_\theta^{valid}(y_t^p)$ remains close to a target value η, preventing it from becoming excessively large and incurring gradient vanishing. Specifically, we have the following lemma:

Lemma 2. *For each training token instance (x, y_t^p), assuming the logits of the valid tokens $f_\theta(z), z \in \mathcal{Z}_{valid}$ follow a Gaussian distribution $\mathcal{N}(\mu_t, \sigma_t^2)$. Then τ_t for the equation $P_\theta^{valid}(y_t^p) = \eta$ can be approximated as:*

$$\tau_t \approx \frac{(f_\theta(y_t^p) - \mu_t) - \sqrt{(f_\theta(y_t^p) - \mu_t)^2 - 2\sigma_t^2 \log(|\mathcal{Z}_{valid}|\eta)}}{2\log(|\mathcal{Z}_{valid}|\eta)} \tag{11}$$

The proof is presented in the appendix. The proof references the work [7] but adapts the process to token-wise LLM-based recommendation scenarios and different distribution conditions. The assumption of a Gaussian distribution nearly holds, as discussed in the appendix.

Eq.(11) gives the token-wise optimal configuration of τ_t. To make the training more stable and reduce the extra effort of calculating the token-wise τ_t, we prefer to set a global uniform τ across various training instances:

$$\tau \approx \frac{(f_\theta(y_t^p) - \mu) - \sqrt{(f_\theta(y_t^p) - \mu)^2 - 2\sigma^2 \log m\eta}}{2\log m\eta} \tag{12}$$

where μ, σ^2 denote the mean and variance of $f_\theta(z)$ for all training instances, and m denotes the average number of valid tokens. This Eq.(12) can adaptively adjust the value of τ according to the current

Table 4: Statistics of the datasets. AVT represents the average number of valid tokens per token instance.

Dataset	#Users	#Items	#Interactions	#Density	#AVT
Toy	19124	11758	165247	0.0735%	54.38
Book	16559	6344	151928	0.1446%	70.90
Clothing	39230	22948	277534	0.0308%	53.74
Office	4895	2414	53149	0.4498%	8.98

model state and the average number of valid tokens in the datasets, serving as an efficient alternative to brute-force hyperparameter search.

5 Experiments

We aim to answer the following research questions:

- **RQ1:** How does MSL perform compare to existing state-of-the-art recommendation methods?
- **RQ2:** How do different components of MSL affect?
- **RQ3:** How does MSL perform compared with state-of-the-art in terms of both accuracy and efficiency?

5.1 Experimental Settings

5.1.1 Datasets. Four conventional real-world datasets: *Amazon Toys and Games*, *Amazon Books*, *Amazon Clothing, Shoes and Jewelry* and *Amazon Office Products* [2] are utilized in our experiments, which are commonly used for the studies of LLM-based recommendation [2, 5, 10, 25, 28]. To ensure a fair comparison, we adopt the same data preprocessing used in recent studies [2, 10]. Specifically, we firstly apply the 5-core setting to the original dataset, then for user interaction sequences longer than 11 interactions, a sliding window of length 11 is applied to segment the sequences. The resulting sequences are then sorted in ascending order by timestamp and split into training, validation, and testing sets with an 8:1:1 ratio. We randomly retain 100,000 items for *Amazon Books* before 5-core processing due to its large size. The processed dataset statistics are presented in Table 4.

5.1.2 Baselines. The methods compared fall into several categories:

- **Traditional recommenders (SASRec [22], BERT4Rec [46], DROS [69])** SASRec utilizes a self-attention-based model to capture user interests. BERTRec adopts the BERT to bidirectional model user preferences. DROS incorporates DRO to improve the model's resilience to distributional shifts.
- **LLM-enhanced recommenders (DLLM2Rec [10], LLM-CF [47])** DLLM2Rec introduces a distillation module designed to bridge the performance gap between LLMs and traditional RS. LLM-CF enhances traditional RS by integrating reasoning-driven collaborative filtering features derived from LLMs using CoT techniques. We use SASRec as the backbone for LLM-enhanced recommenders.
- **LLM-based recommenders (BIGRec [2], LLaRA [31], A-LLM [23])** BIGRec develops instruction-tuning templates to fine-tuning LLMs on RS datasets. LLaRA enhances collaborative signals by incorporating embeddings produced by traditional models into prompts. A-LLM further alignes these embeddings with corresponding textual information.
- **Improved Loss Function for LLM-based Recommenders (S-DPO [9])** S-DPO leverages the DPO to guide LLMs using the ranking information of positive and negative samples.

5.1.3 Implementation Details. LLaMA3 8B model [12] is utilized as the backbone of all the LLM-based recommenders. As for training LLM-based recommenders, we train the models for 10 epochs and report the results of the epoch with the highest NDCG@5. For inference, we evaluate two mainstream methods as baselines: grounding [2] and constrained beam search [74], and we report the better-performing results. The ranking results obtained from constrained beam search are used to construct the recommendation list, with the number of beams fixed at 10. For MSL, we only modify the loss function of the original backbone while following its original hyperparameter settings. The parameter η is set to 0.25. To ensure fair comparisons, we leverage the source code provided in the original papers and tune the hyperparameters of all baseline methods following the guidelines specified in their respective works. Two widely-used metrics *NDCG@K* and *Hit Ratio@K* are employed for evaluating the recommendation accuracy (K = 5, 10).

5.2 Performance Comparison (RQ1)

Table 5 provides a comparative analysis of the performance of the proposed MSL method against baseline approaches.

MSL demonstrates a significant enhancement in the performance of various LLM-based recommenders. MSL consistently outperforms all baseline across all datasets. This remarkable improvement can be attributed to the design of MSL as a specialized loss function tailored for LLM-based recommenders.

The performance improvements of LLM-enhanced recommenders remain relatively limited. LLM-CF exhibits negative gains on three out of the four datasets. This underperformance is primarily due to the inherent gap between LLMs and traditional models, which hinders the effective transfer of knowledge. DLLM2Rec, which directly generates ranking results using LLMs and incorporates a specially designed distillation mechanism, partially addresses this gap. However, its performance remains constrained by the limitations of LLMs' recommendation capabilities as teacher models.

S-DPO demonstrates limited and inconsistent performance improvements in LLM-based recommendation systems. Its dependence on the reference model, coupled with instability introduced by sampling, significantly constrains its effectiveness. As a result, S-DPO often fails to deliver consistent improvements, and in some cases, even demonstrates negative performance gains. For example, NDCG@10 of A-LLM+S-DPO decreases from 0.0132 to 0.0093 on the Book dataset.

5.3 Ablation Study (RQ2)

Table 6 presents the results of ablation study. Specifically, we investigate the effects of MSL, temperature τ and the ATS module, as well as alternative approaches to mitigating the vanishing gradient problem.

[2] https://jmcauley.ucsd.edu/data/amazon/index_2014.html

Table 5: The performance comparison on four real-world datasets. The best result is bolded. Improvement denotes the improvement of MSL over the best results obtained using LML and S-DPO. "N" represents NDCG, and "H" represents Hit Ratio.

Method	Toy				Book				Clothing				Office			
	N@5	N@10	H@5	H@10	N@5	N@10	H@5	H@10	N@5	N@10	H@5	H@10	N@5	N@10	H@5	H@10
SASRec	0.0101	0.0126	0.0190	0.0265	0.0097	0.0133	0.0176	0.0285	0.0046	0.0056	0.0086	0.0116	0.0132	0.0183	0.0260	0.0421
BERT4Rec	0.0157	0.0191	0.0229	0.0336	0.0118	0.0171	0.0187	0.0351	0.0071	0.0093	0.0110	0.0180	0.0225	0.0307	0.0358	0.0618
SASRec+DROS	0.0129	0.0160	0.0217	0.0311	0.0110	0.0156	0.0196	0.0340	0.0050	0.0067	0.0088	0.0142	0.0130	0.0226	0.0260	0.0561
LLM-CF	0.0103	0.0132	0.0186	0.0275	0.0106	0.0142	0.0178	0.0292	0.0041	0.0052	0.0074	0.0116	0.0144	0.0192	0.0239	0.0395
DLLM4Rec	0.0104	0.0134	0.0190	0.0284	0.0099	0.0136	0.0178	0.0303	0.0042	0.0061	0.0082	0.0138	0.0137	0.0198	0.0239	0.0504
BIGRec+LML	0.0138	0.0182	0.0213	0.0353	0.0109	0.0137	0.0169	0.0258	0.0047	0.0073	0.0092	0.0174	0.0113	0.0220	0.0203	0.0530
BIGRec+SDPO	0.0174	0.0219	0.0271	0.0413	0.0118	0.0145	0.0189	0.0276	0.0062	0.0089	0.0114	0.0198	0.0129	0.0256	0.0239	0.0629
BIGRec+MSL	0.0245	0.0288	0.0357	0.0488	0.0125	0.0172	0.0214	0.0356	0.0091	0.0120	0.0146	0.0236	**0.0402**	**0.0438**	**0.0556**	0.0665
Improvement	41.06%	31.35%	31.54%	18.18%	5.70%	18.55%	13.25%	28.93%	46.05%	34.52%	28.07%	19.19%	213.17%	70.80%	132.61%	5.79%
LLaRA+LML	0.0145	0.0193	0.0225	0.0375	0.0102	0.0132	0.0173	0.0265	0.0050	0.0083	0.0088	0.0190	0.0093	0.0204	0.0177	0.0514
LLaRA+SDPO	0.0164	0.0211	0.0250	0.0394	0.0094	0.0114	0.0128	0.0187	0.0077	0.0099	0.0138	0.0206	0.0196	0.0242	0.0379	0.0524
LLaRA+MSL	0.0233	0.0283	0.0336	0.0488	**0.0135**	0.0175	**0.0244**	0.0365	**0.0108**	**0.0140**	**0.0176**	**0.0274**	0.0374	0.0417	0.0535	**0.0670**
Improvement	41.98%	34.05%	34.17%	23.81%	31.96%	32.71%	40.79%	37.93%	40.12%	41.61%	27.54%	33.01%	90.87%	72.08%	41.10%	27.82%
A-LLM+LML	0.0151	0.0197	0.0236	0.0378	0.0107	0.0132	0.0173	0.0251	0.0055	0.0082	0.0100	0.0182	0.0110	0.0219	0.0192	0.0519
A-LLM+SDPO	0.0155	0.0201	0.0248	0.0388	0.0065	0.0093	0.0094	0.0180	0.0072	0.0104	0.0116	0.0216	0.0189	0.0240	0.0306	0.0468
A-LLM+MSL	**0.0248**	**0.0296**	**0.0365**	**0.0513**	0.0130	**0.0184**	0.0228	**0.0395**	0.0097	0.0127	0.0158	0.0252	0.0353	0.0388	0.0488	0.0597
Improvement	59.81%	47.73%	47.06%	32.26%	21.01%	39.44%	31.58%	57.27%	34.65%	22.54%	36.21%	16.67%	86.84%	61.46%	59.32%	15.00%

Adjusting the temperature τ significantly enhances MSL performance. MSL without temperature underperforms compared to LML on certain datasets (e.g., Toy and Office) due to the vanishing gradient issue. Employing an hyperparameter search strategy for the τ effectively addresses this issue, resulting in substantial performance improvements. This highlights the critical role of τ in unlocking MSL's potential. Notably, a similar hyperparameter search for LML revealed only marginal performance gains, suggesting that the observed improvements are attributable to MSL itself rather than the tuning of τ.

The ATS module rivals or surpasses brute-force hyperparameter search in effectiveness. This demonstrates the efficacy of ATS as a dynamic optimization strategy. ATS offers better flexibility than fixed temperature and eliminates the need for exhaustive manual tuning.

Alternative strategies for mitigating vanishing gradients show limited effectiveness. We also investigate two alternative strategies to address the vanishing gradient problem in MSL: adjusting the learning rate (i.e., MSL + tuning lr) and introducing a balancing coefficient α for negative tokens (i.e., MSL + α, where the negative token component in the denominator of $\mathcal{L}_{MSL}$ is multiplied by α). α is set as $|\mathcal{Z}|/|\mathcal{Z}_{\text{valid}}|$. While these adjustments yielded minor improvements, the results remained significantly inferior to those achieved through temperature adjustment.

Table 6: Ablation study. Results are reported in NDCG@10. "+tuning τ" indicates performing a hyperparameter search for the temperature τ. "+tuning lr" indicates performing a hyperparameter search for the learning rate. "+ α" denotes introducing a coefficient for negative tokens. MSL (w/o τ) represents MSL without the temperature. MSL (w/ ATS) represents MSL with the temperature adjusted using ATS.

Method	Toy	Book	Clothing	Office
LML	0.0182	0.0137	0.0073	0.0220
LML + tuning τ	0.0182	0.0146	0.0074	0.0248
MSL (w/o τ)	0.0171	0.0151	0.0097	0.0135
MSL + tuning lr	0.0184	0.0151	0.0101	0.0177
MSL + α	0.0202	0.0157	0.0078	0.0221
MSL + tuning τ	0.0280	0.0182	0.0130	0.0418
MSL (w/ ATS)	0.0288	0.0172	0.0120	0.0438

5.4 Efficiency Comparison (RQ3)

In this section, we analyze and compare the efficiency and performance of various methods. As illustrated in Figure 7, MSL achieves optimal recommendation performance while simultaneously demonstrating superior computational efficiency. Specifically, MSL improves efficiency by 315% and 324% on the Toy and Book datasets, respectively, compared to S-DPO. In comparison to LML, MSL achieves efficiency gains of 4.4% and 6.7% on the same datasets.

The enhanced efficiency of MSL stems from its ability to significantly reduce the number of invalid tokens. MSL restricts the scope to a small subset of valid tokens (the average number of valid tokens for each dataset is shown in Table 4) compared to the 128,000 tokens in LLaMA3 vocabulary. This targeted approach minimizes computational overhead. The only additional overhead introduced by MSL arises from the construction of the Trie tree during the data preprocessing stage. As discussed in Section 4.1, this process is highly efficient. Table 7 shows that the consuming time for all datasets is consistently within one second. In contrast,

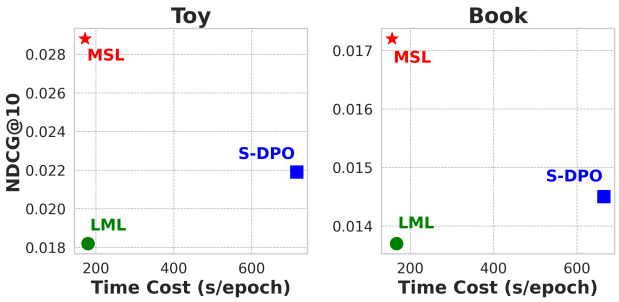

Figure 7: Performance comparisons in terms of both recommendation accuracy and efficiency.

Table 7: Time of constructing trie tree.

Dataset	Toy	Book	Clothing	Office
Time	0.38s	0.17s	0.81s	0.23s

S-DPO introduces additional computational overhead by requiring the sampling of n negative samples for each positive sample, leading to approximately four times runtime compared to MSL.

6 Related Work

6.1 Sequential Recommendation

Sequential recommendation aims to predict the next item of interest for a user based on their historical interactions. It has increasingly adopted various deep learning models over recent years. For example, GRU4Rec [18] leverages recurrent neural networks (RNNs), while Caser [48] utilizes convolutional neural networks (CNNs). More recently, models such as SASRec [22] and BERT4Rec [46] are built upon the self-attention mechanism [8, 50], which automatically assigns weights to each interaction to capture their relative importance. Additionally, DROS [69] incorporates distributionally robust optimization (DRO) [42, 62] to improve the model's robustness to out-of-distribution scenarios, which are common in RS [6, 14, 15, 33, 51, 72]. The readers may refer to the survey [13, 56] for more details.

6.2 LLMs for Recommendation

Large language models (LLMs), with their powerful comprehension capabilities and extensive knowledge [1, 12, 52], have been widely applied in RS. Two primary paradigms for using LLMs in RS as following.

LLM-based Recommenders. This paradigm attempts directly leveraging pre-trained LLMs as the backbone for recommendations using their zero-shot capabilities [16, 19, 35, 55, 59]. However, these methods often perform poorly due to the significant discrepancy between the recommendation tasks and the training objectives of LLMs [68]. To address this, subsequent research has reformulated recommendation data into prompt formats and fine-tuned LLMs to improve their performance, achieving better results [2, 3, 17, 20, 26, 27, 29, 30, 34, 45, 53, 54, 57, 65, 70, 71, 74].

However, these studies all rely on the LML for fine-tuning, without addressing the inherent misalignment between LML and recommendation tasks. To address this issue, S-DPO [9] builds upon DPO [41] by constructing positive and negative samples, explicitly incorporating ranking information into the model training process. However, it suffers from performance instability, limited effectiveness, and low efficiency. Our proposed MSL effectively addresses these issues and significantly improves the performance of LLM-based RS.

LLM-enhanced Recommenders. This paradigm primarily utilizes LLMs in auxiliary roles, such as encoders to embed the semantic information of users and items [25, 43, 44, 54, 60, 64, 66], as an additional knowledge base [39, 67], as a reasoning tool to generate chain-of-thought (CoT) data [47, 58], or serves as teachers using distillation [10, 36]. The main challenge of this paradigm lies in the significant gap between LLMs and traditional recommendation models, which hinders the effective transfer of knowledge.

7 Conclusion

In this paper, we introduce a novel loss function, MSL, specifically tailored for LLM-based RS. MSL excludes invalid tokens from participating in the loss calculation, achieving better alignment with the recommendation objectives and avoid the interference from erroneous negative signal. Despite its advantages, it can lead to gradient vanishing issues during training. To mitigate this, we introduce the temperature coefficient and propose an Adaptive Temperature Strategy, which adaptively adjusts the temperature without requiring extensive hyperparameter tuning. We validate the effectiveness of MSL through theoretical analysis and empirical experiments. Our findings demonstrate that MSL significantly improves the performance of state-of-the-art LLM-based recommendation models.

This study highlights the importance of optimizing LLMs for recommendation tasks by refining their loss functions. Future research could explore the design of specialized LLM architectures to further enhance their suitability for recommendation systems.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62372399, 62476244), OPPO Research Fund, and the advanced computing resources provided by the Supercomputing Center of Hangzhou City University.

Appendix

Due to page limitations, the appendix is provided in the arXiv version of this paper, available at: https://arxiv.org/pdf/2504.04178.

References

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

[2] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnaan He, and Qi Tian. 2023. A bi-step grounding paradigm for large language models in recommendation systems. *arXiv preprint arXiv:2308.08434* (2023).

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

[4] Ferenc Bodon and Lajos Rónyai. 2003. Trie: an alternative data structure for data mining algorithms. *Mathematical and Computer Modelling* 38, 7-9 (2003), 739–751.

[5] Yuwei Cao, Nikhil Mehta, Xinyang Yi, Raghunandan Keshavan, Lukasz Heldt, Lichan Hong, Ed H Chi, and Maheswaran Sathiamoorthy. 2024. Aligning Large Language Models with Recommendation Knowledge. *arXiv preprint arXiv:2404.00245* (2024).

[6] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 21–30.

[7] Jiawei Chen, Junkang Wu, Jiancan Wu, Xuezhi Cao, Sheng Zhou, and Xiangnan He. 2023. Adap-τ: Adaptively modulating embedding magnitude for recommendation. In *Proceedings of the ACM Web Conference 2023*. 1085–1096.

[8] Sirui Chen, Jiawei Chen, Sheng Zhou, Bohao Wang, Shen Han, Chanfei Su, Yuqing Yuan, and Can Wang. 2024. SIGformer: Sign-aware Graph Transformer for Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1274–1284.

[9] Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. 2024. On Softmax Direct Preference Optimization for Recommendation. *arXiv preprint arXiv:2406.09215* (2024).

[10] Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. 2024. Distillation Matters: Empowering Sequential Recommenders to Match the Performance of Large Language Models. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 507–517.

[11] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904* (2020).

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[13] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.

[14] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*. 238–248.

[15] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting filter bubbles by counterfactual interactive recommender system. *ACM Transactions on Information Systems* 42, 1 (2023), 1–27.

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

[17] Zhaolin Gao, Joyce Zhou, Yijia Dai, and Thorsten Joachims. 2024. End-to-end Training for Recommendation with Language-based User Profiles. *arXiv preprint arXiv:2410.18870* (2024).

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[19] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*. Springer, 364–381.

[20] Zhiyu Hu, Yang Zhang, Minghao Xiao, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024. Exact and Efficient Unlearning for Large Language Model-based Recommendation. *arXiv preprint arXiv:2404.10327* (2024).

[21] Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating LLM hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 1827–1843.

[22] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[23] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1395–1406.

[24] Thomas Elmar Kolb. 2024. Enhancing Cross-Domain Recommender Systems with LLMs: Evaluating Bias and Beyond-Accuracy Measures. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 1388–1394.

[25] Dong-Ho Lee, Adam Kraft, Long Jin, Nikhil Mehta, Taibai Xu, Lichan Hong, Ed H Chi, and Xinyang Yi. 2024. STAR: A Simple Training-free Approach for Recommendations using Large Language Models. *arXiv preprint arXiv:2410.16458* (2024).

[26] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1348–1357.

[27] Weitao Li, Junkai Li, Weizhi Ma, and Yang Liu. 2024. Citation-Enhanced Generation for LLM-based Chatbot. *arXiv preprint arXiv:2402.16063* (2024).

[28] Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443* (2023).

[29] Xinyi Li, Yongfeng Zhang, and Edward C Malthouse. 2023. Exploring fine-tuning chatgpt for news recommendation. *arXiv preprint arXiv:2311.05850* (2023).

[30] Zelong Li, Jianchao Ji, Yingqiang Ge, Wenyue Hua, and Yongfeng Zhang. 2024. PAP-REC: Personalized Automatic Prompt for Recommendation Language Model. *arXiv preprint arXiv:2402.00284* (2024).

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

[32] Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3497–3508.

[33] Siyi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. 2024. How do recommendation models amplify popularity bias? An analysis from the spectral perspective. *arXiv preprint arXiv:2404.12008* (2024).

[34] Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Bridging items and language: A transition paradigm for large language model-based recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1816–1826.

[35] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 452–461.

[36] Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024. LLM-ESR: Large Language Models Enhancement for Long-tailed Sequential Recommendation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[37] Hyunsoo Na, Minseok Gang, Youngrok Ko, Jinseok Seol, and Sang-goo Lee. 2024. Enhancing Large Language Model Based Sequential Recommender Systems with Pseudo Labels Reconstruction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 7213–7222.

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

[39] Jiarui Qin, Weiwen Liu, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. D2K: Turning Historical Data into Retrievable Knowledge for Recommender Systems. *arXiv preprint arXiv:2401.11478* (2024).

[40] Alec Radford. 2018. Improving language understanding by generative pre-training. (2018).

[41] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2024).

[42] Hamed Rahimian and Sanjay Mehrotra. 2019. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659* (2019).

[43] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3464–3475.

[44] Yankun Ren, Zhongde Chen, Xinxing Yang, Longfei Li, Cong Jiang, Lei Cheng, Bo Zhang, Linjian Mo, and Jun Zhou. 2024. Enhancing Sequential Recommenders with Augmented Knowledge from Aligned Large Language Models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 345–354.

[45] Wentao Shi, Xiangnan He, Yang Zhang, Chongming Gao, Xinyue Li, Jizhi Zhang, Qifan Wang, and Fuli Feng. 2024. Enhancing Long-Term Recommendation with Bi-level Learnable Large Language Model Planning. *arXiv preprint arXiv:2403.00843* (2024).

[46] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[47] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. Large Language Models Enhanced Collaborative Filtering. *arXiv preprint arXiv:2403.17688* (2024).

[48] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[49] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language

models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 491–500.
[50] Ashish Vaswani. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[51] Bohao Wang, Jiawei Chen, Changdong Li, Sheng Zhou, Qihao Shi, Yang Gao, Yan Feng, Chun Chen, and Can Wang. 2024. Distributionally Robust Graph-based Recommendation System. In *Proceedings of the ACM on Web Conference 2024*. 3777–3788.
[52] Bohao Wang, Feng Liu, Changwang Zhang, Jiawei Chen, Yudi Wu, Sheng Zhou, Xingyu Lou, Jun Wang, Yan Feng, Chun Chen, et al. 2024. Llm4dsr: Leveraing large language model for denoising sequential recommendation. *arXiv preprint arXiv:2408.08208* (2024).
[53] Hangyu Wang, Jianghao Lin, Bo Chen, Yang Yang, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards Efficient and Effective Unlearning of Large Language Models for Recommendation. *arXiv preprint arXiv:2403.03536* (2024).
[54] Hangyu Wang, Jianghao Lin, Xiangyang Li, Bo Chen, Chenxu Zhu, Ruiming Tang, Weinan Zhang, and Yong Yu. 2023. ALT: Towards Fine-grained Alignment between Language and CTR Models for Click-Through Rate Prediction. *arXiv preprint arXiv:2310.19453* (2023).
[55] Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153* (2023).
[56] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830* (2019).
[57] Yidan Wang, Zhaochun Ren, Weiwei Sun, Jiyuan Yang, Zhixiang Liang, Xin Chen, Ruobing Xie, Su Yan, Xu Zhang, Pengjie Ren, et al. 2024. Enhanced generative recommendation via content and collaboration integration. *arXiv preprint arXiv:2403.18480* (2024).
[58] Yuling Wang, Changxin Tian, Binbin Hu, Yanhua Yu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Liang Pang, and Xiao Wang. 2024. Can Small Language Models be Good Reasoners for Sequential Recommendation?. In *Proceedings of the ACM on Web Conference 2024*. 3876–3887.
[59] Zhefan Wang, Weizhi Ma, and Min Zhang. 2024. To Recommend or Not: Recommendability Identification in Conversations with Pre-trained Language Models. *arXiv preprint arXiv:2403.18628* (2024).
[60] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 806–815.
[61] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671* (2023).
[62] Junkang Wu, Jiawei Chen, Jiancan Wu, Wentao Shi, Xiang Wang, and Xiangnan He. 2023. Understanding contrastive learning via distributionally robust optimization. *Advances in Neural Information Processing Systems* 36 (2023), 23297–23320.
[63] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web* 27, 5 (2024), 60.
[64] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, et al. 2023. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933* (2023).
[65] Wujiang Xu, Qitian Wu, Zujie Liang, Jiaojiao Han, Xuying Ning, Yunxiao Shi, Wenfang Lin, and Yongfeng Zhang. 2024. Slmrec: empowering small language models for sequential recommendation. *arXiv preprint arXiv:2405.17890* (2024).
[66] Wentao Xu, Qianqian Xie, Shuo Yang, Jiangxia Cao, and Shuchao Pang. 2024. Enhancing Content-based Recommendation via Large Language Model. *arXiv preprint arXiv:2404.00236* (2024).
[67] Shenghao Yang, Weizhi Ma, Peijie Sun, Min Zhang, Qingyao Ai, Yiqun Liu, and Mingchen Cai. 2024. Common Sense Enhanced Knowledge-based Recommendation with Large Language Model. *arXiv preprint arXiv:2403.18325* (2024).
[68] Weiqin Yang, Jiawei Chen, Xin Xin, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. 2024. PSL: Rethinking and Improving Softmax Loss from Pairwise Perspective for Recommendation. *arXiv preprint arXiv:2411.00163* (2024).
[69] Zhengyi Yang, Xiangnan He, Jizhi Zhang, Jiancan Wu, Xin Xin, Jiawei Chen, and Xiang Wang. 2023. A generic learning framework for sequential recommendation with distribution shifts. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 331–340.
[70] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001* (2023).
[71] Wenlin Zhang, Xiangyang Li, Yuhao Wang, Kuicai Dong, Yichao Wang, Xinyi Dai, Xiangyu Zhao, Huifeng Guo, Ruiming Tang, et al. 2024. Tired of Plugins? Large Language Models Can Be End-To-End Recommenders. *arXiv preprint arXiv:2404.00702* (2024).
[72] Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. 2022. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2022), 9920–9931.
[73] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1435–1448.
[74] Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*. 3162–3172.