TSPORec: Token Selection via Preference Optimization for Sequential Recommendation with Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have emerged as powerful tools for improving recommendation systems. The effectiveness of LLMs arises from their ability to harness rich textual information and their capacity to model heterogeneous user preferences based on users' interaction history. However, due to the large-scale and deep architectures, LLM-based sequential recommendation approaches generally incur high inference costs, resulting in a low return on investment. To mitigate this cost, many existing approaches resort to using only the first few tokens of item descriptions, which inadvertently discards valuable information contained in the full text, thereby leading to suboptimal recommendation performance. To address this limitation, we propose a novel Token Selection approach for Preference Optimization in LLM-based sequential Recommendation, i.e., TSPORec, which accurately pinpoints informative tokens throughout the entire textual content to improve recommendation performance. Specifically, we design a three-stage pipeline to select informative tokens and introduce a novel proxy reward to facilitate the implementation. TSPORec not only enhances recommendation performance but also improves computational efficiency. Extensive experiments across two models and datasets demonstrate the superb performance (up to 31.25%) and efficiency (up to 63.4%) of our approach compared with six baseline approaches.

1 Introduction

Recommendation systems have become indispensable for mitigating information overload across a wide range of commercial platforms (Zhai et al., 2024; Zhu et al., 2025). A key factor in delivering personalized item recommendations lies in accurately modeling the heterogeneous preferences of users. Recently, Large Language Models (LLMs) have demonstrated remarkable effectiveness in capturing such preferences (Kim et al., 2024; Liao et al., 2024; Zheng et al., 2024), owing to their vast parameter scales and strong contextual understanding. In particular, Sequential Recommendation (SR), which models diverse interests of users based on their recent interaction histories, has proven highly effective in enhancing recommendation performance when integrated with LLMs (Chen et al., 2024; Liu et al., 2024a; Chen et al., 2025).

The goal of SR is to predict the next items a user is likely to prefer by modeling sequential patterns in their historical interactions. A key challenge in this field lies in simultaneously capturing item-level sequential dependencies and modeling the complex and heterogeneous preferences of users derived from these interaction sequences.

Owing to the pivotal role in advancing recommendation systems, SR has attracted sustained attention from both academia and industry (Rendle et al., 2010; Cheng et al., 2013; Chen et al., 2024; Liu et al., 2024a). Existing approaches (Kang and McAuley, 2018; Li et al., 2020; Wu et al., 2020) predominantly leveraged item IDentification (ID) features, overlooking the abundant textual content typically present in real-world settings. Furthermore, their reliance on shallow network architectures limit their capacity to model the complex and heterogeneous nature of user preferences.

Recently, LLMs have emerged as effective solutions for overcoming the limitations of previous approaches, demonstrating strong performance in SR tasks (Chen et al., 2024; Liu et al., 2024a; Chen et al., 2025). The core idea behind these approaches is to leverage the rich textual features and powerful representational capabilities of LLMs to extract robust representations of items and users. For example, HLLM (Chen et al., 2024) employs separate item and user LLMs, which are based on LLMs such as TinyLlama (Zhang et al., 2024b) or Baichuan (Baichuan, 2023), to learn comprehensive item and user representations.

However, existing LLM-based SR approaches

are highly resource-intensive. For instance, given an interaction history of length S with each item represented by n tokens, a single inference step requires processing $S \times n$ tokens, which consists of both computation-intensive and memory-intensive operations. To mitigate this cost, existing approaches (Chen et al., 2024; Liu et al., 2024a) typically exploit only the first few tokens of each item, failing to fully exploit their rich textual features. This limitation raises an important research question: *Can we identify and select the most informative tokens from textual features to maximize information utilization?* Addressing this question could benefit SR systems in two key ways. First, it enables effective usage of critical information within the text. Second, it reduces computational overhead through compressing the overall token sequence length.

In this paper, we propose a novel token selection approach for preference optimization in LLM-based Sequential Recommendation, i.e., **TSPORec**. TSPORec is composed of three stages to select the most informative tokens from textual features. First, we pre-train an SR model based on an LLM. Next, we freeze the LLM backbone and attach a policy head, which is then trained using a newly designed proxy reward function. Using the trained policy, we extract informative tokens from item texts and subsequently retrain the LLM on the refined input sequences. We conduct extensive experiments across six recommendation approaches and two publicly available benchmark datasets to demonstrate the effectiveness and generality of our approach. The key contributions of this paper are summarized as follows:

- We propose a new token selection pipeline that automatically identifies informative tokens from textual features. This pipeline shows strong generalization across LLMs and datasets.
- We design a novel proxy reward method to guide policy learning in an effective and targeted manner. This method can identify meaningful token importance, leading to an effective policy.
- Extensive experimental results demonstrate the superb performance (up to 29.29% in terms of recall and 31.25% in terms of NDCG) and efficiency (up to 63.4% inference overhead reduction) of TSPORec compared with six State-of-the-Art baseline approaches.

2 Preliminaries

Modern recommendation systems typically represent users and items as learned embedding vectors. Let $\mathbf{e}_u$ and $\mathbf{e}_i$ denote the embeddings of a user and an item, respectively. In SR, the user embedding is derived from their interaction history: $\mathbf{e}_u = g(\{\mathbf{e}_i\})$, where $g(\cdot)$ is commonly a shallow neural network or a LLM and $\mathbf{e}_i$ represents a positive item. A positive item $\mathbf{e}_i$ for User u is an item that User u has interacted with. In this work, we denote the LLM exploited to aggregate item embeddings into a user representation as the **user LLM**, and the LLM employed to generate item embeddings via token selection as the **item LLM**.

A widely adopted training objective in recommendation systems is the InfoNCE loss (van den Oord et al., 2018), as defined in Formula (1):

$$\mathcal{L}_{\text{InfoNCE}}(\mathbf{e}_i) = -\log \frac{\exp(\langle \mathbf{e}_u, \mathbf{e}_i \rangle)}{\exp(\langle \mathbf{e}_u, \mathbf{e}_i \rangle) + \sum_{j=1}^{N} \exp(\langle \mathbf{e}_u, \mathbf{e}_i^j \rangle)}, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes a similarity measure, i.e., dot product, and $\{\mathbf{e}_i^1, \ldots, \mathbf{e}_i^N\}$ are embeddings of N negative items. Negative items are typically the items that the user has not interacted with. We denote $\mathbf{E}(\mathbf{e}_i) = [\langle \mathbf{e}_u, \mathbf{e}_i \rangle, \langle \mathbf{e}_u, \mathbf{e}_i^1 \rangle, \ldots, \langle \mathbf{e}_u, \mathbf{e}_i^N \rangle]$ the vector of similarity scores between the user embedding and the positive item embedding along with N negative items. We take $label = [1, 0, \ldots, 0]$ as the corresponding one-hot label vector. The InfoNCE loss can then be equivalently expressed as a cross-entropy objective:

$$\mathcal{L}_{\text{InfoNCE}}(\mathbf{e}_i) = \mathcal{L}_{\text{CE}}\big(\text{softmax}(\mathbf{E}(\mathbf{e}_i)), label\big), \tag{2}$$

where $\mathcal{L}_{\text{CE}}$ denotes the Cross-Entropy loss.

3 Method

In this section, we first formulate the problem to address in this paper. Then, we detail our approach, i.e., TSPORec, including the new token selection pipeline and the novel proxy reward method.

3.1 Problem Formulation

We study the problem of token selection in LLM-based SR. Given an item represented by a token sequence $T = \{t_1, t_2, \ldots, t_n\}$, conventional LLM-based approaches typically derive an item embedding using only the first k tokens:

$$\mathbf{e}_i = \text{LLM}(t_1, t_2, \ldots, t_k),$$

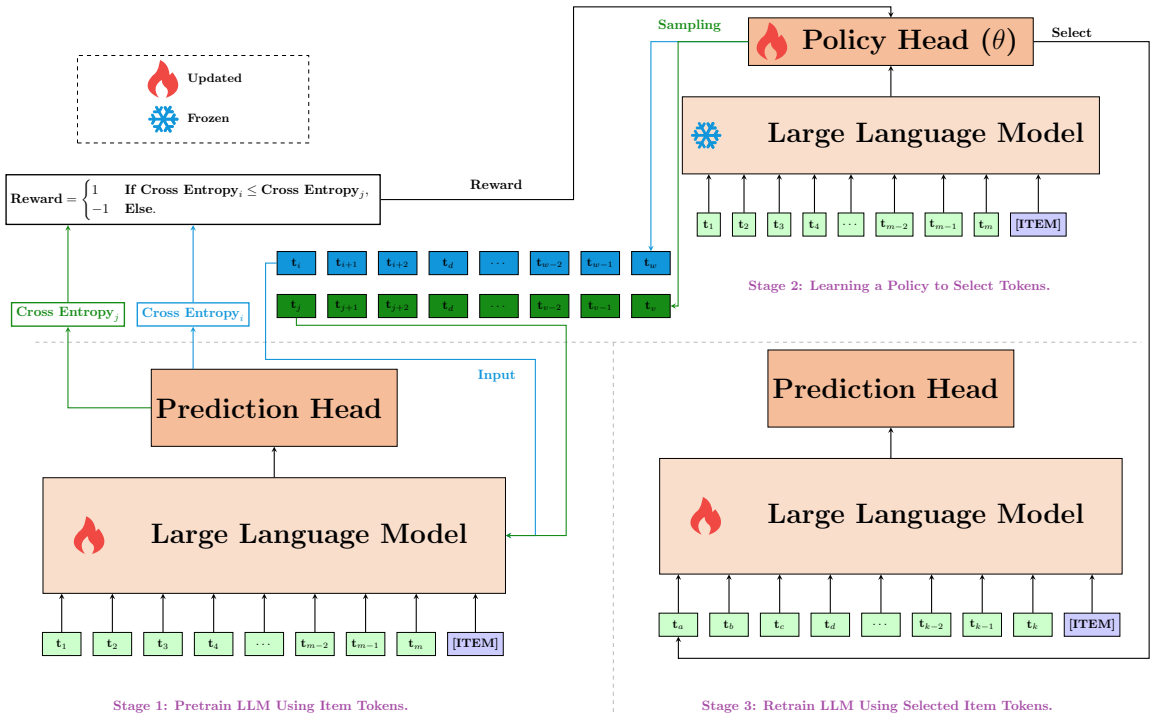


Figure 1: Overview of TSPORec's Three-Stage Training Pipeline: (1) LLM Pretraining, (2) Token Selection Policy Learning, (3) Token Selection and Retraining.

where $k \leq n$. Instead, we aim to identify an informative subset of k tokens, denoted as $T_k \subseteq T$, and feed them to the LLM to compute a refined item embedding: $\mathbf{e}'_i = \text{LLM}(T_k)$. Our goal is to enhance downstream recommendation performance by leveraging $\mathbf{e}'_i$, achieving superior results compared to those obtained using the original prefix-based embedding $\mathbf{e}_i$. The selection of T_k is guided by a learnable policy, which prioritizes tokens that contribute most to user preference modeling. The problem addressed in this paper can be formulated as maximizing the expected reward:

$$\max_{T_k} \mathcal{R}(\theta). \tag{3}$$

We defer the definition of the reward $\mathcal{R}(\theta)$ to the next section, where it is given in Formula (16).

3.2 Informative Tokens Selection

In this section, we detail our approach, i.e., TSPORec. To maximize flexibility, we perform informative token identification at the *chunk level*. A chunk is a subset of tokens consisting of several consecutive tokens in a token sequence. Specifically, we select $\lfloor k/c \rfloor$ chunks from the token sequence T, where each chunk consists of c consecutive tokens:

$$\left[(t_1^1, t_1^2, \ldots, t_1^c), \ldots, (t_{\lfloor k/c \rfloor}^1, t_{\lfloor k/c \rfloor}^2, \ldots, t_{\lfloor k/c \rfloor}^c)\right],$$

where c denotes the predefined chunk size, and $\lfloor \cdot \rfloor$ is the floor function, which returns the greatest integer less than or equal to its argument. Token-level selection is a special case of this formulation, corresponding to $c = 1$. This chunk-wise strategy allows for controllable granularity in capturing local semantic structures for downstream recommendation tasks. As shown in Figure 1, the overall pipeline of TSPORec consists of three stages: *Pretraining*, *Token Selection Policy Learning*, and *Token Selection and Retraining*.

Pretraining In this stage, we pretrain both the user LLM (denoted as g) and the item LLM (denoted as f) employing the InfoNCE loss defined in Eq. (1) to obtain a foundation model that can be leveraged for reward computation.

For item LLM pretraining, we append a special *[ITEM]* token to the end of the textual token sequence of each item. The item embedding is then extracted as the final-layer hidden state corresponding to this *[ITEM]* token. Specifically, given an input sequence $T = \{t_1, t_2, \ldots, t_k, \textit{[ITEM]}\}$, the item LLM produces hidden states:

$$[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{k+1}]. \tag{4}$$

We utilize $\mathbf{h}_{k+1}$, i.e., the representation at the *[ITEM]* position, as the item embedding.

Token Selection Policy Learning In this stage, we introduce our novel token selection policy. Given the full set of textual features for all items, exhaustively evaluating every possible token subset is computationally infeasible due to its exponential time complexity. To mitigate this burden, we propose a differentiable proxy reward that approximates the utility of a token subset, enabling efficient optimization of the selection policy.

We begin by quantifying the informativeness of each token t_i through its relevance to the item embedding. Specifically, we compute a scalar importance score, referred to as the $\mathrm{info}(\cdot)$, using a query-key attention mechanism:

$$\mathrm{info}(t_i) = \frac{\langle W_Q \mathbf{h}_i, W_K \mathbf{h}_{k+1} \rangle}{\sqrt{d}}, \tag{5}$$

where $\theta = \{W_Q, W_K\}$ denotes a set of learnable parameter matrices in $\mathbb{R}^{d \times d}$, and d is the dimensionality of the hidden embeddings. The resulting score reflects the alignment between the token representation $\mathbf{h}_i$ and the item-level representation $\mathbf{h}_{k+1}$. We then define the probability to select Token t_i as

$$p_i = \frac{\exp\left(\mathrm{info}(t_i)\right)}{\sum_{j=1}^{k} \exp\left(\mathrm{info}(t_j)\right)}. \tag{6}$$

Given a sequence of M items that a user has interacted with, denoted as $\mathcal{I}_M = \{T_1, T_2, \ldots, T_M\}$, where T_j represents the token sequence of the j-th item. We model the joint probability of the interaction sequence as

$$P(\mathcal{I}_M) = \prod_{j=1}^{M} \prod_{\ell=1}^{k} p_{j\ell}, \tag{7}$$

where $p_{j\ell}$ is the probability of the ℓ-th token in T_j.

Based on these definitions, we outline the procedure for training the policy parameters θ. For each token sequence T_j in the interaction history $\mathcal{I}_M$ of a user, we perform two independent sampling passes to extract distinct subsets of token chunks, yielding two derived sequences denoted T'_j and T''_j. This process results in two new token sequence sets:

$$\mathcal{I}'_M = \{T'_1, T'_2, \ldots, T'_M\} \tag{8}$$

$$\mathcal{I}''_M = \{T''_1, T''_2, \ldots, T''_M\}. \tag{9}$$

Using these sampled sequences, we construct two corresponding user embeddings via the pretrained, frozen LLM:

$$\mathbf{e}'_u = g^{\text{frozen}}\big(f^{\text{frozen}}(\mathcal{I}'_M)\big) \tag{10}$$

$$\mathbf{e}''_u = g^{\text{frozen}}\big(f^{\text{frozen}}(\mathcal{I}''_M)\big). \tag{11}$$

We construct the full item embedding based on the full token sequence as

$$\mathbf{e}_i = f^{\text{frozen}}(\mathcal{I}_M). \tag{12}$$

We compute two cross-entropy values by contrasting the user embeddings $\mathbf{e}'_u$ and $\mathbf{e}''_u$ with the full item embedding $\mathbf{e}_i$, exploiting a set of N negative item embeddings $\{\mathbf{e}_i^j\}_{j=1}^{N}$. This yields two cross-entropy values, $\mathcal{L}_{\text{ce}}^1$ and $\mathcal{L}_{\text{ce}}^2$:

$$\mathcal{L}_{\text{ce}}^1 = -\log \frac{\exp(\langle \mathbf{e}'_u, \mathbf{e}_i \rangle)}{\exp(\langle \mathbf{e}'_u, \mathbf{e}_i \rangle) + \sum_{j=1}^{N} \exp(\langle \mathbf{e}'_u, \mathbf{e}_i^j \rangle)} \tag{13}$$

$$\mathcal{L}_{\text{ce}}^2 = -\log \frac{\exp(\langle \mathbf{e}''_u, \mathbf{e}_i \rangle)}{\exp(\langle \mathbf{e}''_u, \mathbf{e}_i \rangle) + \sum_{j=1}^{N} \exp(\langle \mathbf{e}''_u, \mathbf{e}_i^j \rangle)}. \tag{14}$$

The reward signal r is then defined as:

$$r = \begin{cases} 1 & \text{if } \mathcal{L}_{\text{ce}}^1 \leq \mathcal{L}_{\text{ce}}^2, \\ -1 & \text{otherwise.} \end{cases} \tag{15}$$

Our objective is to optimize the policy parameters θ by maximizing the expected reward:

$$\mathcal{R}(\theta) = \mathbb{E}\Big[r \cdot \Big(\log P(\mathcal{I}'_M \mid \theta) - \log P(\mathcal{I}''_M \mid \theta)\Big)\Big] \tag{16}$$

Given the above description of TSPORec, we establish the following theorem:

Theorem 1. *Under the framework of TSPORec, the following properties hold:*

- *Smaller values of the cross-entropy losses $\mathcal{L}_{\text{ce}}^1$ and $\mathcal{L}_{\text{ce}}^2$ in Eq. (13) and Eq. (14) yield a tighter approximation to the ground-truth preference distribution in terms of KL divergence.*
- *Token chunks shared between the two sampled sequences $\mathcal{I}'_M$ and $\mathcal{I}''_M$ do not contribute to the gradient updates of the policy parameters θ.*
- *The objective in Eq. (16) increases the likelihood of selecting informative token chunks while suppressing less informative ones.*

Proof. We defer the detailed proof to Appendix F. □

Dataset	Method	R@5	R@10	R@50	N@5	N@10	N@50	Impr. (avg)
Amazon Books	SASRec	3.38	5.09	11.59	2.24	2.79	4.20	+0.0 %
	HSTU	2.88	4.51	11.06	1.89	2.41	3.83	-11.46%
	LLMinit	3.29	5.05	11.74	2.18	2.74	4.19	-1.14%
	HLLM(random)	3.29	5.17	12.96	2.17	2.77	4.45	+2.14 %
	HLLM(topk logits)	3.39	5.34	13.35	2.24	2.86	4.59	+5.36%
	HLLM(first-k)	4.16	6.29	14.49	2.80	3.48	5.25	+24.40%
	TSPORec (Ours)	**4.37**	**6.55**	**14.98**	**2.94**	**3.64**	**5.47**	**+29.86%**
	Best Impr.	29.29%	28.68%	29.25%	31.25%	30.47%	30.24%	N/A
Pixel	SASRec	2.41	3.77	9.63	1.59	2.03	3.29	+0.0%
	HSTU	2.14	3.41	8.85	1.40	1.81	2.97	-10.22%
	LLMinit	2.65	4.10	10.53	1.73	2.19	3.58	+8.92%
	HLLM(random)	2.82	4.39	11.07	1.84	2.34	3.78	+15.71%
	HLLM(topk logits)	2.79	4.36	11.02	1.82	2.33	3.76	+14.89%
	HLLM(first-k)	2.80	4.41	11.15	1.83	2.35	3.80	+15.88%
	TSPORec (Ours)	**2.88**	**4.51**	**11.20**	**1.88**	**2.40**	**3.85**	**+18.15%**
	Best Impr.	19.50%	19.63%	16.30%	18.24%	18.23%	17.02%	N/A

Table 1: Performance comparison of various methods on the Amazon Books and Pixel datasets, using Qwen3-Embedding-0.6B as the backbone model, with text sequences truncated to 64 tokens.

Token Selection and Retraining After optimizing the policy parameters, we proceed to select informative token chunks as follows. We define the probability of a chunk c as $P(c) = \prod_{i=1}^{|c|} p_i$, where p_i denotes the generation probability of the i-th token in the chunk, and $|c|$ is the number of tokens in c. Chunks with higher probability scores according to $P(c)$ are selected. Using this set of high-probability chunks, we construct a refined dataset and retrain the model on this updated representation to improve both its performance and efficiency.

4 Experiments

In this section, we first describe the experimental setup, followed by a series of comprehensive experiments designed to evaluate the effectiveness and the efficiency of TSPORec. Finally, we present a case study that illustrates the specific tokens selected by TSPORec, providing insights into its operational mechanism.

4.1 Experimental Setup

We conduct a comprehensive evaluation of TSPORec against four state-of-the-art baselines: (1) traditional ID-based sequential recommendation models, such as SASRec (Kang and McAuley, 2018) and HSTU (Zhai et al., 2024); (2) semantic initialization methods that leverage large language model (LLM)-derived embeddings to initialize item ID representations, exemplified by LLMinit (Harte et al., 2023); and (3) hierarchical LLM-based frameworks, specifically HLLM (Chen et al., 2024). In addition, to study the impact of token selection, we adapt these baselines with different token selection policies, resulting in six baseline variants for comparison. We evaluate our approach on two publicly available datasets: the Amazon Book Reviews dataset (McAuley et al., 2015) and the Pixel dataset (Cheng et al., 2023). For backbone LLMs, we employ Qwen3-Embedding-0.6B (Zhang et al., 2025b) and TinyLlama-1.1B (Zhang et al., 2024b). Performance is evaluated using Recall@K (R@K) and NDCG@K (N@K). Additional details on the experimental setup are given in Appendix A.

4.2 Experimental Results

Overall Performance As shown in Table 1, TSPORec significantly outperforms all baseline approaches by a substantial margin. In this setting, input sequences are truncated to 64 tokens, and the Qwen3-Embedding-0.6B model serves as the frozen LLM backbone. For TSPORec, the *chunk size* is set to 8. On the Amazon Books dataset, TSPORec improves upon SASRec by up to 29.29% in Recall@K and 31.25% in NDCG@K, yielding an average gain of 29.43% across metrics. On the Pixel dataset, it achieves improvements of up to 19.63% in Recall@K and 18.24% in NDCG@K, with an average increase of 16.79%.

The Importance of Tokens in LLM-based Recommendation Given the significant advances achieved by LLM-based sequential recommendation methods, we investigate the impact of different types of input tokens. Specifically, we compare *random tokens*, *top-k logit tokens*, *the first k tokens*, and TSPORec. For *random tokens*, we randomly select k tokens from the item text. For *top-k logit tokens*, we select the k tokens with the high-

est logits. *The first k tokens* is the default strategy employed by many existing methods. In contrast, TSPORec selects k tokens using a learned policy.

As shown in Table 1, TSPORec achieves the most significant performance gain by leveraging the learned token selection policy, substantially outperforming existing token selection strategies. On the Amazon Books dataset, randomly selecting tokens reduces the performance improvement of HLLM from 24.40% to just 2.14%, highlighting the critical role of input token selection in LLM-based recommendation.

These results demonstrate that the choice of input tokens has a profound impact on model effectiveness—a factor largely overlooked by prior approaches. Surprisingly, the *top-k logit tokens* do not outperform the simple *first-k tokens* baseline, suggesting that more principled token selection mechanisms are needed. This limitation may stem from the unidirectional nature of standard LLM attention, which prevents hidden states from effectively capturing meaningful collaborative signals across the sequence. In contrast, as illustrated in Eq. (5), TSPORec leverages a bidirectional scoring mechanism. The importance score of each token is learned from both (i) the standard attention mechanism, which encodes semantic meaning, and (ii) the pretrained item embedding $\mathbf{h}_{k+1}$, which provides a collaborative filtering signal. This dual-source design enables TSPORec to jointly capture rich semantic information and sequential collaborative patterns, thereby facilitating the identification of the most informative tokens and ultimately improving model performance.

The Impact of Token Sequence Length in LLM-based Recommendation In this section, we further investigate how the length of input tokens impacts recommendation performance. To this end, we conduct extensive experiments with varying input token lengths, as shown in Figure 2. We denote each configuration as *Method*$-k$, where k represents the number of input tokens used. For instance, HLLM-128 indicates the use of the first 128 tokens from the input sequence, while TSPORec-64 denotes the selection of 64 tokens according to the learned token selection policy.

As illustrated in Figure 2, on the Amazon Books dataset, increasing the token length leads to performance improvements for both HLLM and TSPORec, suggesting that longer input sequences may contain additional informative tokens that benefit recommendation quality. In contrast, on the Pixel dataset, while moderate increases in token length initially enhance HLLM's performance, further lengthening the input sequence eventually degrades performance. This indicates that not all tokens contribute positively to recommendations, and including irrelevant or noisy tokens can be detrimental. This phenomenon further underscores the necessity of effective token selection to improve recommendation performance. Moreover, on both the Amazon Books and Pixel datasets, TSPORec with only 64 selected tokens achieves performance on par with or superior to the corresponding HLLM variants using 256 tokens. These results demonstrate that TSPORec not only enhances recommendation accuracy but also improves computational efficiency by enabling effective performance with significantly fewer tokens.

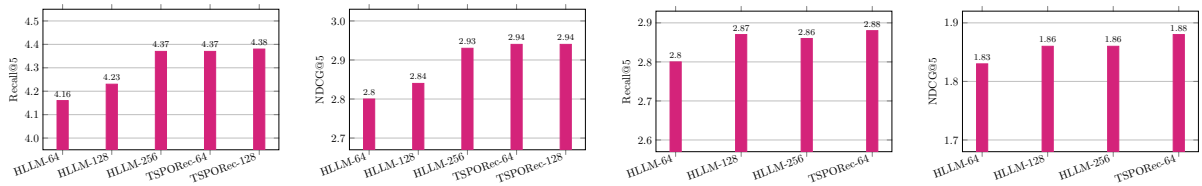
Token$_S$	I$_t$ (ms)	U$_t$ (ms)	Total (ms)
64	299	27	326
128	509	27	536
256	817	26	843

Table 2: Inference time cost during evaluation. Token$_S$ denotes the length of the input token sequence, I_t the inference time of the item LLM, and U_t the inference time of the user LLM, using Qwen3-Embedding-0.6B.

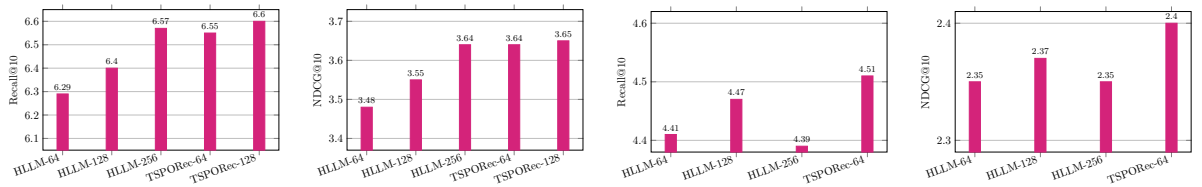
The Efficiency of TSPORec To evaluate the impact on inference efficiency of TSPORec, we analyze the computational cost across varying input sequence lengths. By selecting informative tokens, TSPORec effectively reduces the required input length without sacrificing performance, thereby lowering overall computational overhead.

We measure the inference time of LLM on an NVIDIA H100 GPU using the Amazon Books dataset, with a batch size of 32 and an item sequence length of 10. The results are summarized in Table 2. We observe that the item LLM dominates the inference time, and its computational share grows significantly as the input token sequence length increases. Since TSPORec matches the performance of HLLM using only 64 tokens with HLLM using 256 tokens, it reduces inference cost by **63.4%** in scenarios where item embeddings are precomputed offline, and by **61.3%** when both the item and user LLMs are served online.

The Generalization capability of TSPORec To evaluate the generalization ability of TSPORec, we conduct experiments using a different LLM backbone—`TinyLlama-1.1B`, as well as a different downstream SR model, LLMinit (Harte et al.,



(a) Amazon Book, Recall@5 (b) Amazon Book, NDCG@5 (c) Pixel, Recall@5 (d) Pixel, NDCG@5



(e) Amazon Book, Recall@10 (f) Amazon Book, NDCG@10 (g) Pixel, Recall@10 (h) Pixel, NDCG@10

Figure 2: Performance of HLLM and TSPORec on Amazon Books and Pixel with different token sequence lengths, using `Qwen3-Embedding-0.6B` as the backbone model.

Method	R@5	R@10	N@5	N@10	Impr.
HLLM	3.99	6.07	2.66	3.33	+0.0%
TSPORec	**4.16**	**6.30**	**2.78**	**3.46**	**+4.12%**

Table 3: Performance comparison of HLLM and TSPORec on the Amazon Books dataset, using TinyLlama-1.1B as the backbone model, with text sequences truncated to 64 tokens.

Method	R@5	R@10	Impr.
LLMinit(first-k)	3.29	5.05	+ 0.0%
LLMinit(TSPORec)	**3.42**	**5.23**	**+3.76%**

Table 4: Performance comparison of LLMinit on the Amazon Books dataset, with different tokens for LLM embedding generation.

2023). We compare TSPORec against the HLLM variant with the same LLM backbone, and the results are presented in Table 3. TSPORec consistently outperforms HLLM across all Recall and NDCG metrics, achieving an average improvement of 4.12%. Furthermore, as shown in Table 4, when TSPORec-selected tokens are used to initialize LLMinit, the model performance improves by 3.76%, demonstrating that the selected tokens carry informative signals for recommendation. These results confirm that TSPORec not only enhances downstream performance but also exhibits strong generalization across different LLM architectures and recommendation models.

4.3 Case Study

In this section, we conduct a case study to examine the underlying mechanisms of token selection. An illustrative example is shown in Figure 3, with additional cases provided in Appendix B for further analysis. We compare the tokens selected by the first-k strategy and TSPORec on the Pixel dataset. As depicted in Figure 3, notable differences exist between the two strategies, highlighted in green (tokens selected by TSPORec but not by first-k) and blue (tokens selected by first-k but not by TSPORec).

We observe that TSPORec removes the text segment *"released in the summer of In accordance"* and instead selects new content such as *"cucumber, endless mode, Penny"*. The newly selected tokens are predominantly content words—nouns, adjectives, or named entities—that convey specific and semantically rich information about the item. In contrast, the removed tokens consist largely of function words (e.g., prepositions, pronouns, and articles) that carry limited discriminative power. By prioritizing content words, TSPORec enriches item representations with more informative features, thereby enhancing downstream recommendation performance.

Figure 4 further illustrates that TSPORec tends to filter out frequent ("hot") tokens. This behavior is meaningful: hot tokens are shared across many items and thus exhibit low semantic specificity, making it difficult for the user LLM to distinguish between items and learn effective representations.

5 Related Work

LLM-based SR LLMs possess extensive world knowledge and strong reasoning capabilities, offering two principal advantages for sequential recommendation: rich semantic understanding and powerful modeling capacity derived from their high-dimensional parameter space. These characteristics enable LLMs to effectively capture complex user behavior patterns and thereby enhance recommendation performance (Sun et al., 2024; Zhang et al.,

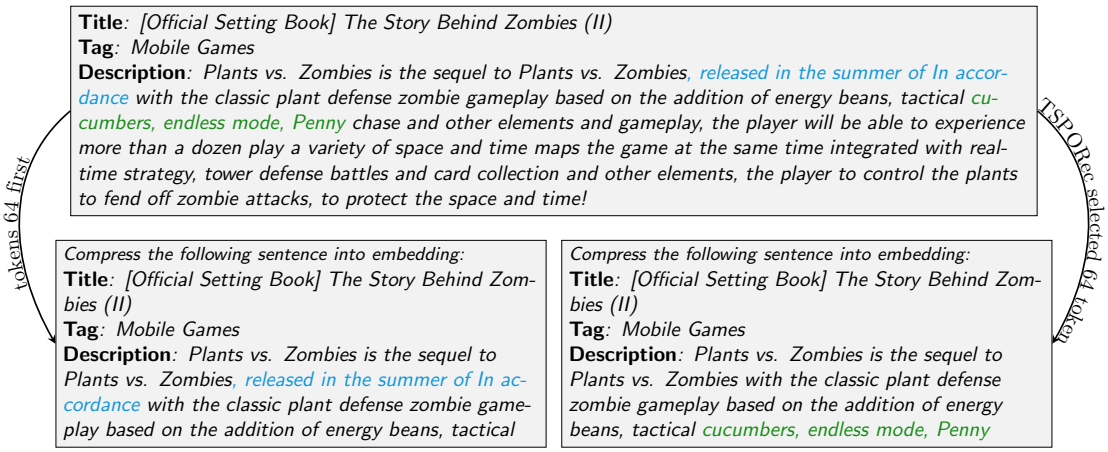


Figure 3: Case study on token selection for item "i9019" in the Pixel dataset, with 64 tokens selected by each method.

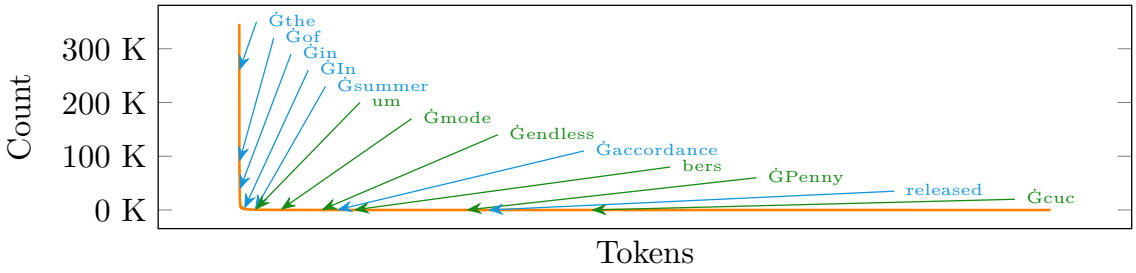


Figure 4: Comparison of token distributions selected by first-k and TSPORec ($\dot{G}$ is a leading space token).

2024a, 2025a; Qiao et al., 2024; Liu et al., 2024b; Xu et al., 2024). In the context of SR, LLM-based approaches can be broadly categorized into two groups.

The first category leverages LLMs as semantic initializers to enrich ID-based item embeddings (Harte et al., 2023; Liu et al., 2024a; Hu et al., 2025). A key challenge in these methods arises from the dimensional mismatch between the high-dimensional LLM embeddings and the typically lower-dimensional ID embeddings used in recommendation systems. To address this, dimensionality reduction techniques are commonly employed—such as Principal Component Analysis in LLMEmb (Liu et al., 2024a) and Singular Value Decomposition in AlphaFuse (Hu et al., 2025).

The second category adopts LLMs in an end-to-end manner, where the model directly utilizes the LLM and tokenized item sequences to predict the next item in the interaction sequence of users. HLLM (Chen et al., 2024) demonstrates significant performance improvements over traditional methods. This end-to-end integration enables modeling of both semantic content and collaborative signals from user interactions, thereby advancing the recommendation performance.

However, due to computational constraints, only the first portion of the textual description of each item is used. This truncation limits the model's ability to capture the full richness of the semantic information present in longer descriptions, potentially leading to suboptimal performance. In contrast, our method selects the most informative tokens from item descriptions.

6 Conclusion

In this paper, we propose a novel Token Selection approach tailored for LLM-driven Sequential Recommendation systems, i.e., TSPORec. We develop a three-stage workflow. First, we pre-train the foundational LLM-based sequential recommendation model to establish a robust baseline. Second, we train a dedicated policy to identify informative tokens; additionally, a proxy reward function is designed to facilitate chunk-oriented token selection, addressing the challenges of granularity in token identification. Finally, leveraging the identified tokens, we retrain the pre-trained baseline model to optimize its recommendation performance. Comprehensive experimental results demonstrate the effectiveness (up to 31.25%) and efficiency (up to 63.4%) of the proposed method.

Limitations

The proposed TSPORec employs a three-stage pipeline to select the most informative tokens from the item's textual features. While TSPORec significantly improves both recommendation performance and inference efficiency compared to standard training methods, it incurs additional training time due to Stages 2 and 3. However, this overhead is practically acceptable, as models are typically trained once and deployed over an extended period—rendering inference efficiency a more critical concern than the one-time training cost.

References

Baichuan. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv: 2309.10305*.

Junyi Chen, Lu Chi, Bingyue Peng, and Zehuan Yuan. 2024. Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling. *arXiv preprint arXiv:2409.12740*.

Junyi Chen, Lu Chi, Siliang Xu, Shiwei Ran, Bingyue Peng, and Zehuan Yuan. 2025. Hllm-creator: Hierarchical llm-based personalized creative generation. *arXiv preprint arXiv:2508.18118*.

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, pages 2605–2611.

Yu Cheng, Yunzhu Pan, Jiaqi Zhang, Yongxin Ni, Aixin Sun, and Fajie Yuan. 2023. An image dataset for benchmarking recommender systems with raw pixels. *arXiv preprint arXiv:2309.06789*.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008.

Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, D. Jannach, and Marios Fragkoulis. 2023. Leveraging large language models for sequential recommendation. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic preference optimization without reference model. In *EMNLP*.

Guoqing Hu, An Zhang, Shuo Liu, Zhibo Cai, Xun Yang, and Xiang Wang. 2025. Alphafuse: Learn id embeddings for sequential recommendation in null space of language embeddings. *arXiv preprint arXiv:2504.19218*.

Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*, pages 197–206.

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. KDD '24, page 1395–1406, New York, NY, USA. Association for Computing Machinery.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *WSDM*, pages 322–330.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. SIGIR '24, page 1785–1795, New York, NY, USA. Association for Computing Machinery.

Qidong Liu, Xian Wu, Wanyu Wang, Yejing Wang, Yuanshao Zhu, Xiangyu Zhao, Feng Tian, and Yefeng Zheng. 2024a. Large language model empowered embedding generator for sequential recommendation. *arXiv preprint arXiv:2409.19925*.

Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024b. Llm-esr: Large language models enhancement for long-tailed sequential recommendation. *Advances in Neural Information Processing Systems*, 37:26701–26727.

Zhiwei Liu, Yongjun Chen, Jia Li, Philip S. Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479*.

Junru Lu, Jiazheng Li, Siyu An, Meng Zhao, Yulan He, Di Yin, and Xing Sun. 2024. Eliminating biased length reliance of direct preference optimization via down-sampled kl divergence. *arXiv preprint arXiv:2406.10957*.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 43–52.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. In *NeurIPS*.

Shutong Qiao, Chen Gao, Junhao Wen, Wei Zhou, Qun Luo, Peixuan Chen, and Yong Li. 2024. Llm4sbr: A lightweight and effective framework for integrating large language models in session-based recommendation. *arXiv preprint arXiv:2402.13840*.

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

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820.

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*, page 1441–1450.

Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. Large language models enhanced collaborative filtering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2178–2188.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 17–22.

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*, page 565–573.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. Sse-pt: Sequential recommendation via personalized transformer. In *RecSys*, pages 328–337.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. 2021. Contrastive learning for sequential recommendation. *arXiv preprint arXiv:2010.14395*.

Wujiang Xu, Qitian Wu, Zujie Liang, Jiaojiao Han, Xuying Ning, Yunxiao Shi, Wenfang Lin, and Yongfeng Zhang. 2024. Slmrec: Distilling large language models into small for sequential recommendation. *arXiv preprint arXiv:2405.17890*.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. 2023. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. *arXiv preprint arXiv:2310.20453*.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. 2024. Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*.

Chao Zhang, Shiwei Wu, Haoxin Zhang, Tong Xu, Yan Gao, Yao Hu, and Enhong Chen. 2024a. Notellm: A retrievable large language model for note recommendation. In *Companion Proceedings of the ACM Web Conference 2024*, pages 170–179.

Chao Zhang, Haoxin Zhang, Shiwei Wu, Di Wu, Tong Xu, Xiangyu Zhao, Yan Gao, Yao Hu, and Enhong Chen. 2025a. Notellm-2: Multimodal large representation models for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2815–2826.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024b. Tinyllama: An open-source small language model. *arXiv preprint arXiv: 2401.02385*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Chenzhuo Zhao, Ziqian Liu, Xingda Wang, Junting Lu, and Chaoyi Ruan. 2025. Pmpo: Probabilistic metric prompt optimization for small and large language models. *arXiv preprint arXiv:2505.16307*.

Zhi Zheng, {Wen Shuo} Chao, Zhaopeng Qiu, Hengshu Zhu, and Hui Xiong. 2024. Harnessing large language models for text-rich sequential recommendation. In *WWW 2024 - Proceedings of the ACM Web*

Conference, WWW 2024 - Proceedings of the ACM Web Conference, pages 3207–3216. Association for Computing Machinery, Inc.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*.

Jie Zhu, Zhifang Fan, Xiaoxie Zhu, Yuchen Jiang, Hangyu Wang, Xintian Han, Haoran Ding, Xinmin Wang, Wenlin Zhao, Zhen Gong, Huizhi Yang, Zheng Chai, Zhe Chen, Yuchao Zheng, Qiwei Chen, Feng Zhang, Xun Zhou, Peng Xu, Xiao Yang, and 2 others. 2025. Rankmixer: Scaling up ranking models in industrial recommenders. CIKM '25, New York, NY, USA. Association for Computing Machinery.

A More Experimental Setup

Dataset	#Users	#Items	#Interactions
Amazon Books	694,898	686,624	10,053,086
Pixel	200,000	96,282	3,965,656

Table 5: Statistics of the Pixel and Amazon Books Datasets.

In this section, we detail the experimental setup used to evaluate our approach.

We conduct experiments on two publicly available datasets: the Amazon Book Reviews dataset (McAuley et al., 2015) and the Pixel dataset (Cheng et al., 2023). We present the statistics of the datasets used in Table 5. The Amazon Books dataset contains 694,898 users and 686,624 items, while the Pixel dataset contains 200,000 users and 96,282 items. Both datasets exhibit high sparsity, with user-item interaction densities of 2.1×10^{-5} and 2.1×10^{-4}, respectively. Items with missing attributes are filtered out. We adopt a leave-one-out evaluation protocol: the most recent interaction is used for testing, the second-to-last for validation, and all earlier interactions for training. Performance is evaluated using Recall@K (R@K) and NDCG@K (N@K).

Following the setup of HLLM (Chen et al., 2024), we set the learning rate to 1×10^{-4} for all baseline models. HLLM is trained for 5 epochs, while other models are trained for up to 200 epochs with early stopping to prevent overfitting. We fix the number of negative samples at 128 and set the maximum sequence length to 10. In the policy learning phase, the model is trained for 5 epochs, with a learning rate 1×10^{-3}.

To investigate the impact of textual input length, we truncate item descriptions to 64, 128, and 256 tokens, respectively. For the Amazon Books dataset, we use the *title* and *description* fields. For the Pixel dataset, we utilize the *title*, *tag*, and *description* text features. Since starting tokens occupy a distinct feature space (Han et al., 2024), TSPORec retains the first *prefix size* to 16 tokens from all sampled sequences.

We train all models on a GPU cluster equipped with 8 H100 and 8 A800 GPUs. Each experiment is conducted three times, and we report the average performance to ensure statistical reliability.

B Case Study

We conduct an additional case study to further examine the token selection behavior of the competing methods. The results for item *"i113948"* are illustrated in Figure 5. Consistent with earlier observations, significant differences are evident between the two strategies: tokens selected exclusively by TSPORec are highlighted in green, while those selected only by the first-k strategy are marked in blue, allowing for a fine-grained comparison of their selection patterns.

In this instance, TSPORec removes tag-related segments such as *"tag: Cover Song"* and *"Fill in the lyrics cover original song: model original song"*, and instead selects the segment *"Ronghao Lyrics: Lok Yang, Cold Ashes Singing: Cold test service to play Penglai play"*. This selection is semantically meaningful: for a music item, metadata about the lyricist and composer provides more concrete, fine-grained, and discriminative signals than generic categorical tags. By prioritizing such specific metadata, TSPORec enriches the item representation, thereby improving downstream recommendation performance.

As shown in Figure 6, consistent with the previous case study, TSPORec tends to filter out high-frequency tokens. This behavior encourages the item LLM to learn more discriminative embeddings, as frequent tokens often carry less item-specific semantic information. Please note that, whereas Figure 4 shows the token distribution across the full set of text features, Figure 6 depicts the distribution of token counts in the selection results. The results reveal that—under both counting metrics—the proposed method consistently filters out frequent ("hot") tokens.

C More Experimental Results

The Impact of Training and Selection Chunk Sizes Since the training and selection chunk sizes may differ, we conduct an ablation study to investigate the impact of training and selection chunk sizes on recommendation performance. we denote each configuration as TSPORec-Tk_1Sk_2, where the model is trained with chunk size k_1 and performs token selection during inference with chunk size k_2. We evaluate TSPORec on the Amazon Books dataset using an input sequence length of 128 tokens. The results, shown in Figure 7, demonstrate that: (1) across all configurations, the proposed TSPORec consistently outperforms the HLLM

Figure 5: Case study on token selection for item "i113948" in the Pixel dataset, with 64 tokens selected by each method.

baseline by up to 3.54%, indicating robustness to varying parameter settings.

(2) On average, TSPORec-T16S1, TSPORec-T16S8, TSPORec-T16S16, and TSPORec-T16S32 outperform HLLM by 3.54%, 2.15%, 2.54%, and 0.53%, respectively. A general downward trend in performance is observed as the selection chunk size increases, indicating that smaller chunks yield more effective token selection. We further validate this trend using top-k logit-based selection, with results presented in Table 7. The consistent performance degradation with larger chunk sizes across both methods suggests a shared preference for finer-grained selection units. This observation further implies that the limited effectiveness of top-k logit selection may stem from its unidirectional nature, rather than from disruptions to semantic sentence structure.

Training Time Since TSPORec employs a three-stage training pipeline, we report the training time for each stage in Table 6. The model is trained on a GPU cluster equipped with 8 NVIDIA H100 GPUs. Due to the additional stages, TSPORec requires approximately 21.5 hours more than a single-stage baseline. This overhead is acceptable in practice, as the full training procedure is performed only once and the resulting model is deployed for an extended period.

D More Related Work

D.1 Sequential Recommendation (SR)

Sequential recommendation (SR) captures the diverse preferences and evolving interests of users by modeling their recent interaction sequences (Hidasi et al., 2015; Tan et al., 2016; Zhou et al., 2019; Kang and McAuley, 2018; Li et al., 2020; Wu et al., 2020; Zhai et al., 2024; Harte et al., 2023; Chen et al., 2024).

Traditional SR Traditional sequential recommendation methods primarily focus on enhancing neural architectures to more accurately model users' diverse interests. Representative examples include Caser (Tang and Wang, 2018), SASRec (Kang and McAuley, 2018), BERT4Rec (Sun et al., 2019), CLS2Rec (Xie et al., 2021), and CoSeRec (Liu et al., 2021). While effective, these approaches often struggle to scale to large parameter counts. To address this limitation, generative recommendation methods such as DreamRec (Yang et al., 2023) and HSTU (Zhai et al., 2024) have been proposed, which are inherently scalable to larger models. However, these generative approaches typically fail to leverage the rich semantic knowledge pre-encoded in pre-trained LLMs, thereby limiting their potential to further improve recommendation performance. In contrast, our proposed TSPORec effectively harnesses the full semantic knowledge embedded in pre-trained LLMs, enabling more accurate sequential recommendations.

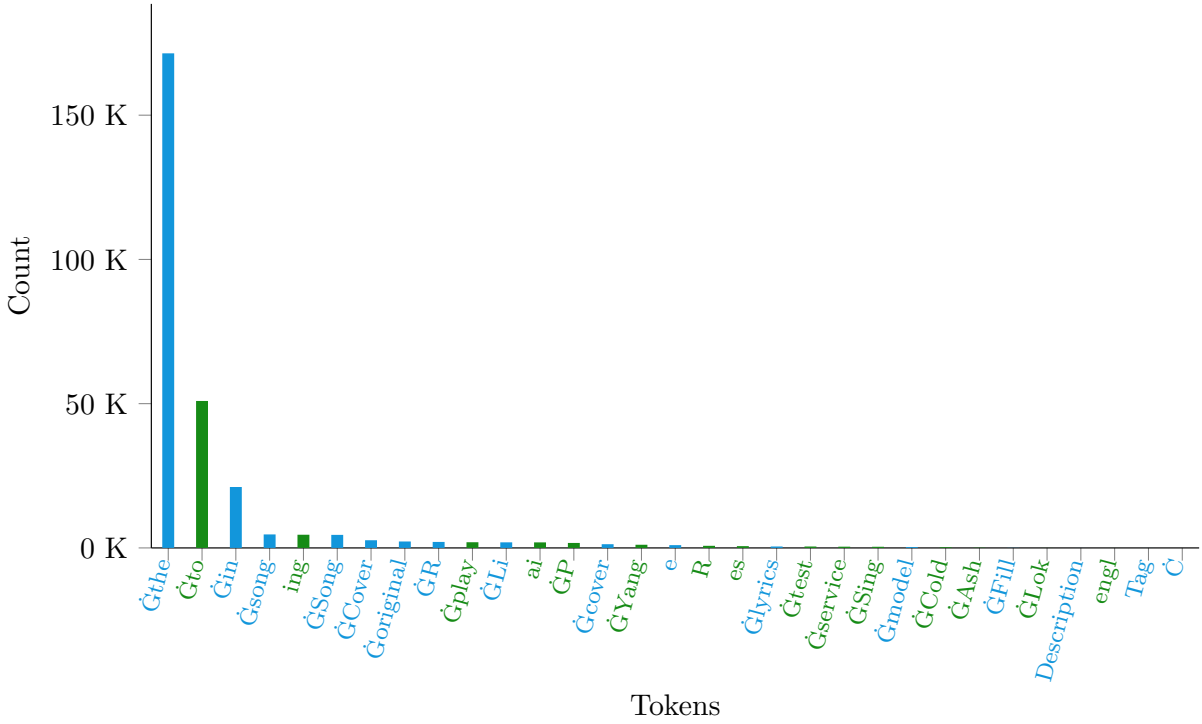
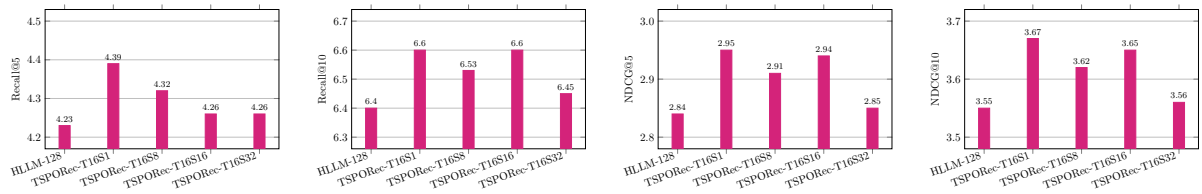


Figure 6: Comparison of token distributions selected by first-k and TSPORec.



(a) Recall@5 (b) Recall@10 (c) NDCG@5 (d) NDCG@10

Figure 7: Performance of HLLM and TSPORec on Amazon Books with different training and selection chunk sizes, using `Qwen3-Embedding-0.6B` as the backbone model.

D.2 Prompt Engineering and Preference Optimization

Prompt Engineering A substantial body of research shows that fine-grained control over individual tokens can unlock more intelligent, efficient, and trustworthy LLM applications, even under constrained token budgets (Lester et al., 2021; Li and Liang, 2021; Zhao et al., 2025; Deng et al., 2022). *Prompt Tuning* (Lester et al., 2021) and *Prefix-Tuning* (Li and Liang, 2021) operate in the model's continuous embedding space, using gradient-based optimization to learn effective prompt representations. RL-Prompt (Deng et al., 2022) reformulates discrete prompt optimization as a reinforcement learning problem, where token sequences are selected to maximize task-specific rewards. The method achieves strong performance across a range of NLP tasks and intriguingly reveals that optimal prompts often deviate from standard grammatical conventions to enhance effectiveness. In contrast to these methods—which optimize prompts to steer model outputs—our work focuses on identifying informative tokens directly from the input.

Preference Optimization Preference optimization aims to align large language models (LLMs) with human preferences and values. Direct Preference Optimization (DPO) (Rafailov et al., 2023) is a representative method that increases the likelihood of outputs preferred by humans while decreasing the likelihood of disfavored ones. DPO has inspired numerous follow-up studies (Meng et al., 2024; Gheshlaghi Azar et al., 2024; Lu et al., 2024; Ethayarajh et al., 2024; Hong et al., 2024). Unlike these approaches, our method does not modify the output probability distribution of LLMs. Instead, we adjust token-level probabilities in the

Stage 1: *Pretraining*	Stage 2: *Token Selection Policy Learning*	Stage 3: *Token Selection and Retraining*
13h 49m 1s	8h 56m 24s	12h 40m 52s

Table 6: Training time of TSPORec for each stage.

Method	R@5	R@10	Impr. (avg)
HLLM-64-S1	**3.39**	**5.34**	+0.0%
HLLM-64-S8	3.16	5.03	-6.29%

Table 7: Performance comparison of HLLM with top-k token selection under different selection chunk sizes on the Amazon Books dataset, using Qwen3-Embedding-0.6B as the backbone and truncating text sequences to 64 tokens.

input space to identify which input segments, when emphasized, can enhance downstream recommendation performance.

E Algorithm Framework

We present the pseudocode of TSPORec in Algorithm 1. TSPORec consists of three main stages. In Stage 1, an LLM-based recommendation model (e.g., HLLM) is pretrained on user-item interaction data using textual features. In Stage 2, the LLM backbone is frozen, and a policy head is introduced; TSPORec then learns a token selection policy using the proposed proxy reward. In the final stage, the learned policy is applied to select the most informative tokens from the dataset, and the base model is retrained on the refined item representations.

Algorithm 1: The TSPORec pipeline

Input : Training Instances: $\mathcal{D}_0$, Number of Epochs: U, Initial Model: π_0.
Output : Final Model: π_U, Updated Training Instances: $\mathcal{D}_U$

// Pretrain LLM-based recommendation model π_1
$\pi_1 \leftarrow \pi_0$
for $i \leftarrow 1$ **to** U **do**
  $\mathbf{x} \sim \mathcal{D}_0$ // Sample training data from $\mathcal{D}_0$
  $\pi_1 \leftarrow \text{UPDATE}(\pi_1, \mathbf{x})$
end
// Freeze the LLM backbone and initialize the policy head.
$\pi_\theta \leftarrow \text{FREEZE-AND-INIT-POLICY}(\pi_1, \theta)$
for $i \leftarrow 1$ **to** U **do**
  $\mathbf{x} \sim \mathcal{D}_0$ // Sample training data from $\mathcal{D}_0$
  $\mathbf{s}_0 \sim \pi_\theta(\mathbf{x})$ // Sampling Item Tokens.
  $\mathbf{s}_1 \sim \pi_\theta(\mathbf{x})$
  $c_0 \leftarrow \text{CROSS-ENTROPY}(\mathbf{s}_0, \pi_1)$
  $c_1 \leftarrow \text{CROSS-ENTROPY}(\mathbf{s}_1, \pi_1)$
  $r = \text{COMPUTE-REWARD}(c_0, c_1)$
  $\pi_\theta \leftarrow \text{UPDATE}(\pi_\theta, r)$
end
$\mathcal{D}_U \leftarrow \text{SELECT}(\pi_\theta, \mathcal{D}_0)$
$\pi_U \leftarrow \pi_0$
for $i \leftarrow 1$ **to** U **do**
  $\mathbf{x} \sim \mathcal{D}_U$ // Sample training data from $\mathcal{D}_U$
  $\pi_U \leftarrow \text{UPDATE}(\pi_U, \mathbf{x})$
end
return $\pi_U, \mathcal{D}_U$

F Proof

Theorem 1. *Under the framework of TSPORec, the following properties hold:*

- *Smaller values of the cross-entropy losses* ce_1 *and* ce_2 *in Eq. (13) and Eq. (14) yield a tighter approximation to the ground-truth preference distribution in terms of KL divergence.*
- *Token chunks shared between the two sampled sequences* $\mathcal{I}'_M$ *and* $\mathcal{I}''_M$ *do not contribute to the gradient updates of the policy parameters* θ*.*
- *The objective in Eq. (16) increases the likelihood of selecting informative token chunks while suppressing less informative ones.*

Proof. Let $\mathcal{L} = -\mathcal{R}(\theta)$.

(1) We denote the preference distribution associated with $\mathbf{e}'_u$ as p', that associated with $\mathbf{e}''_u$ as p'', and the ground-truth preference distribution as q. By definition, the Kullback–Leibler (KL) divergences are given by

$$\mathrm{KL}(q \,\|\, p') = \int q(x) \log \frac{q(x)}{p'(x)} \,\mathrm{d}x \tag{17}$$

$$= \mathrm{ce}_1 - H(q), \tag{18}$$

and

$$\mathrm{KL}(q \,\|\, p'') = \int q(x) \log \frac{q(x)}{p''(x)} \,\mathrm{d}x \tag{19}$$

$$= \mathrm{ce}_2 - H(q), \tag{20}$$

where $H(q) = -\int q(x) \log q(x) \,\mathrm{d}x$ denotes the entropy of q.

Since $H(q)$ is constant with respect to the model parameters, the comparison between ce_1 and ce_2 is equivalent to the comparison between $\mathrm{KL}(q \,\|\, p')$ and $\mathrm{KL}(q \,\|\, p'')$. In other words, a smaller cross-entropy value corresponds to a tighter approximation of the ground-truth preference distribution in terms of KL divergence.

(2) Assume there exists a common chunk C shared between $P_1 = P(\mathcal{I}'_M \mid \theta)$ and $P_2 = P(\mathcal{I}''_M \mid \theta)$. The gradient of the loss with respect to C is given by:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial C} &= -r \frac{\partial(\log P_1 - \log P_2)}{\partial C} \\ &= -r \frac{\partial(\log P_1(C) - \log P_2(C))}{\partial C} + 0 \\ &= 0 \end{aligned} \tag{21}$$

Hence, the gradient $\frac{\partial \mathcal{L}}{\partial C}$ vanishes.

(3) If $\mathrm{ce}_1 < \mathrm{ce}_2$, indicating that the sequence associated with $P_1 = P(\mathcal{I}'_M \mid \theta)$ yields lower cross-entropy than $P_2 = P(\mathcal{I}''_M \mid \theta)$ and is thus more informative, we set the reward $r = 1$. In this case, the optimization objective is:

$$\mathcal{L} = \log P_2 - \log P_1 \tag{22}$$

Minimizing Eq. (22) increases the likelihood of P_1 while decreasing that of P_2.

Conversely, if $\mathrm{ce}_1 > \mathrm{ce}_2$, implying that P_2 corresponds to the more informative sequence, we set $r = -1$. The resulting loss becomes:

$$\mathcal{L} = \log P_1 - \log P_2 \tag{23}$$

Minimizing Eq. (23) promotes P_2 and suppresses P_1.

In both scenarios, the training objective effectively increases the probability of the more informative sequence, thereby guiding the policy network to select token chunks that preserve semantically salient information. □