Rethinking Large Language Model Architectures for Sequential Recommendations

Hanbing Wang[1], Xiaorui Liu[3], Wenqi Fan[4], Xiangyu Zhao[5], Venkataramana Kini[2], Devendra Pratap Yadav[2], Fei Wang[2], Zhen Wen[2], Hui Liu[1]
[1]Michigan State University, [2]Amazon, [3]North Carolina State University
[4]The Hong Kong Polytechnic University, [5]City University of Hong Kong
{wangh137,liuhui7}@msu.edu, {venkini,feiww,yaddevn,zhenwen}@amazon.com
xliu96@ncsu.edu, wenqifan03@gmail.com, xy.zhao@cityu.edu.hk

Abstract

In recent times, there has been a shift towards adapting sequential recommendation to LLM paradigm to harness the capabilities of LLMs. These methods typically formulate recommendation data into natural language and train the model to forecast the subsequent item in an auto-regressive manner. Despite their notable success, the significant computational burden during inference poses a major challenge to their practical implementation. In this study, we aim to streamline current LLM-based recommendation models and introduce a straightforward yet highly effective model LITE-LLM4REC. The primary objective of LITE-LLM4REC is to ensure efficient inference for the sequential recommendation task. LITE-LLM4REC circumvents the step-by-step beam search decoding by employing a direct item projection head to produce ranking scores in one step. This design arises from our empirical finding that beam search decoding is ultimately unnecessary for sequential recommendations. Additionally, LITE-LLM4REC introduces a hierarchical LLM structure crafted to efficiently handle the extensive contextual information of items and redundant computation issue, thus diminishing computational overhead while enjoying the power of LLMs. Experiments on four publicly available datasets validate the efficacy of LITE-LLM4REC in enhancing both performance and inference efficiency (notably 46.8% performance improvement and 99.48% efficiency improvement on ML-1m) compared to existing LLM-based methods. Our implementations are available at: `https://github.com/HanbingWang2001/Lite-LLM4Rec-PyTorch`.

1 Introduction

Sequential recommendation is to predict next item a user will interact with based on his/her interaction history. Because user interests are dynamic and evolving over time, it is important to capture the sequential pattern, leading to accurate recommendations. Traditional methods model the item transition patterns based on Markov Chain (He et al., 2018; Rendle et al., 2010; He and McAuley, 2016). With the development of deep learning, a variety of deep neural networks, such as Transformer (Kang and McAuley, 2018; Sun et al., 2019), RNN (Wu et al., 2019; Hidasi and Karatzoglou, 2018) and CNN (Tang and Wang, 2018), have been proposed to advance the task, achieving remarkable performance. Furthermore, side information (e.g., attributes, titles) has been incorporated (Hidasi et al., 2016; Huang et al., 2019; Zhang et al., 2019), which helps achieve remarkable improvement, demonstrating its importance and potential.

Recently, the widespread success of large language models (LLMs) (Yang et al., 2025; Achiam et al., 2023; Touvron et al., 2023), has demonstrated their exceptional ability of contextual understanding and offers a promising direction to improve recommendation systems with heightened personalization and adaptability. Existing LLM-based recommendation algorithms (Cao et al., 2024; Zhang et al., 2023; Bao et al., 2023) mainly adapt recommendation tasks to the LLM paradigm by formulating relevant information, i.e., interaction information, meta data, or candidate items through various indexing strategies into natural language. As shown in Figure 1, such information will be wrapped in a prompt template and transformed into informative latent representations before being fed into transformer. Finally, the model will auto-regressively generate the recommendations in natural language through decoding with beam search (Yang et al., 2023).

Although existing LLM-based methods have achieved remarkable success, the exorbitant cost of inference hinders their real-world applications and poses a formidable obstacle to seamless, real-time user experiences (Rajput et al., 2023; Yue et al., 2023; Mei and Zhang, 2023). To gain deeper insights into this efficiency challenge, we perform a preliminary study in Section 2.2 and pinpoint two primary performance bottleneck:

- The Decoding Bottleneck. We observe that the step-by-step beam search decoding is the most time-consuming component. This process is employed to generate k recommendations for a user, leading to k times greater model computation complexity.
- The Input Bottleneck. We find that representing items as raw text or simple indices is highly inefficient. This approach suffers from two key problems: (i) tokenizing item identifiers unnecessarily inflates the input sequence length, and (ii) it forces redundant computations every time a frequently-occurring item appears in the data.

Crucially, our further study in Section 2.3 suggests that these computationally burdensome components—both complex decoding and verbose textual item representations—are ultimately unnecessary for achieving high performance in sequential recommendation. This finding motivates our work to design an architecture that bypasses these bottlenecks entirely.

Grounded on our preliminary studies, we aim to streamline the model architecture of existing LLM-based recommendations, introducing LITE-LLM4REC as an efficient solution for sequential recommendation. We identified that the auto-regressive beam search decoding is is a primary source of inefficiency and is unnecessary for this task. Therefore, LITE-LLM4REC pivots from a slow, token-by-token generative paradigm to a rapid, discriminative one. Instead of generating item identifiers, it circumvents the entire decoding process by projecting the final sequence representation directly into the item embedding space. This allows it to compute a full ranking over all candidate items in a single step, drastically cutting down inference latency. Additionally, treating items as long strings of text create massive computational overhead. To tackle this challenge, LITE-LLM4REC employs a hierarchical LLM structure composed of an Item LLM and a Recommendation LLM. This design is grounded on the finding that LLMs are powerful enough to interpret the semantic information encoded directly within latent representations. This architecture significantly

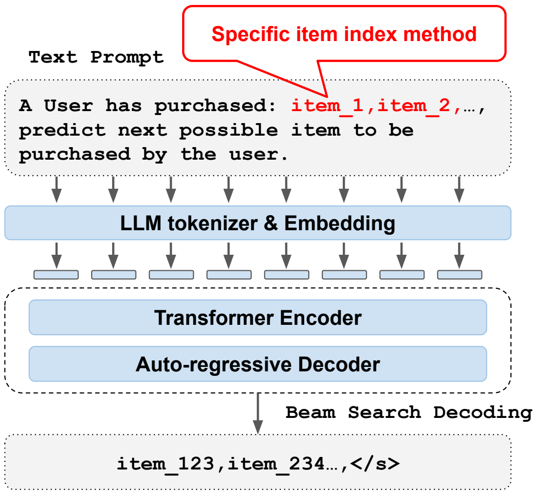
Specific item index method
Text Prompt
A User has purchased: item_1,item_2,…,
predict next possible item to be
purchased by the user.
LLM tokenizer & Embedding
Transformer Encoder
Auto-regressive Decoder
Beam Search Decoding
item_123,item_234…,</s>

Figure 1: An Illustration of LLM-based sequential recommendations.

shortens the effective input length and alleviates redundant computation while still harnessing the core strengths of large language models. Experimental results indicate that LITE-LLM4REC significantly improves not only the inference efficiency but also the overall performance. The main contributions of this paper are summarized as follows: (1) We propose LITE-LLM4REC, a streamlined architecture for sequential recommendation that challenges the standard LLM paradigm. By replacing the computationally expensive auto-regressive decoder with a direct projection head, LITE-LLM4REC simultaneously boosts inference speed and improves recommendation accuracy. (2) We introduce a novel hierarchical LLM structure designed to efficiently manage long item contexts. This architecture disentangles rich item-level feature encoding from user-sequence modeling, significantly reducing computational overhead while preserving the powerful representation capabilities of LLMs. (3) We conduct extensive experiments on several benchmark datasets, demonstrating the effectiveness and efficiency of LITE-LLM4REC.

2 Preliminaries

In this section, we empirically study the inference bottlenecks of existing LLM-based recommendations. To establish a clear framework for this analysis, we begin by formally defining the sequential recommendation problem and introducing the key notations used throughout this work. Then we quantify the inference overhead by benchmarking modern LLM-based methods against a traditional sequential model, BERT4Rec (Sun et al., 2019). Subsequently, we dissect the primary components of these LLM recommenders, analyzing their individual contributions to both computational latency and overall recommendation performance. The insights gained from this detailed analysis directly motivate the novel architectural designs proposed later in this paper.

2.1 Problem Statement and Notations

We denote $u \in \mathcal{U}$ and $i \in \mathcal{I}$ for a user and an item, where $\mathcal{U}$ and $\mathcal{I}$ indicate the user set and the item set, respectively. The interaction history of a user u can be organized as a sequence $\mathcal{I}_u = (i_1, i_2, ..., i_t)$ in a chronological order, where t is the length of $\mathcal{I}_u$ and each item i is associated with textual information $\mathcal{T}_i$ (e.g., title and genre). Given the interaction history $\mathcal{I}_u$ of user u, sequential recommendation algorithms aim to predict next item i_{t+1} the user is most likely to interact with from $\mathcal{I} \backslash \mathcal{I}_u$, which represents the item set formed by excluding the items already interacted with the user from the complete set of items $\mathcal{I}$.

2.2 Efficiency Analysis

In this subsection, we will explore the following questions: 1) In terms of inference efficiency, how do LLM-based recommendation algorithms perform compared with traditional recommendations; and 2) Why do they face such serious inference efficiency problem. To answer these questions, we first demonstrate the huge inference efficiency gap between LLM-based and traditional sequential recommendation systems and then investigate the inference time cost of each component, and the relation between input length and inference time. In this study, we focus on two typical LLM-based recommendation algorithms, i.e., P5 and POD (Geng et al., 2022; Li et al., 2023b), and one representative traditional sequential recommendation system, i.e., BERT4Rec (Sun et al., 2019).

For all the following experiments, we use the implementations released by POD (Li et al., 2023b)[1] and we keep the same experimental settings. Our analysis is based on ML-1m and Movies, two popular publicly-available datasets. Please refer to Appendix G for more results on Toys and Kindle. More details of the datasets can be found in Section 4.1.1. We calculate the inference time by measuring the total time of all the test data going through each component during inference, and then we can obtain the average inference time for a batch

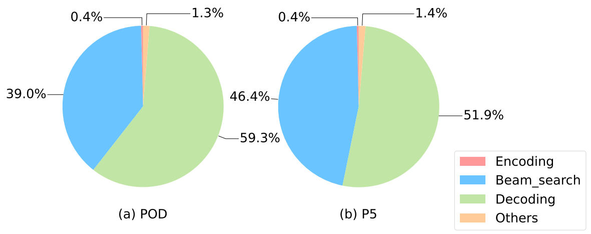

Figure 2: Inference time of different components for a batch of 32 users. 'Beam_search' refers to the beam search decoding process, 'Decoding' transfers the latent representation to token IDs, and 'Others' includes data preparation, metrics calculations etc.

of users. We denote the length of the input as the number of tokens of the tokenized context (e.g., 'item_1234' will be tokenized as 'item' '_' '12' '34' by the tokenizer of T5, and the input length will be 4).

The results are shown in Table 1. We can make the following observations: 1) Compared with BERT4Rec, the efficiency of P5 and POD has been significantly compromised, experiencing a slowdown of nearly a thousand times; and 2) POD is more efficient than P5 via reducing the input length which can lead to the decrease of the computational costs. To further explore why LLM-based methods face such formidable inference problem, we conduct time analysis on the inference time cost of each component in P5 and POD (Geng et al., 2022; Li et al., 2023b). Figure 2 demonstrates the average inference time cost of each component. From the figure, we note that the beam search decoding process is most time-consuming which takes approximately 98.2% on P5 (Geng et al., 2022) and 98.3% on POD (Li et al., 2023b) of the inference time.

Table 1: Inference time (ms) and input length

Method	Avg length	Time/Batch
BERT4Rec	21.0	2.37
P5	112.2	1,646
POD	85.0	1,468

2.3 Effectiveness Analysis

In the previous subsection, we have demonstrated that the inefficiency of LLM-based recommendations comes from the beam search decoding process and input length. In this subsection, we investigate how the beam search decoding and item indexing affect the inference efficiency and the performance. We implement two variants of P5 and

[1] https://github.com/lileipisces/POD

POD as follows: (1) w/o_d. It eliminates the decoder and uses an item projection head to perform the recommendation task (Details in Section 3.3). (2) w/o_d_TID. On the basis of eliminating the decoder, it represents items with their titles instead of random numbers.

We report the performance and inference efficiency in Table 2. We can make the following observations: (1) eliminating the beam search decoding can significantly improve the inference efficiency; (2) although representing items with their titles can improve the performance due to the incorporation of contextual information, it impairs the inference efficiency because the length of input becomes longer (increases to 297.4 on ML-1m), resulting in more computational costs.

Table 2: Performance of P5 and POD and their two variants

Datasets	ML-1m			Movies		
Methods	R@20	N@20	Time	R@20	N@20	Time
P5	0.2985	0.1442	2,280	0.1080	0.0761	1,770
w/o_d	0.3109	0.1459	**40.06**	0.1217	0.0794	**37.01**
w/o_d_TID	**0.3354**	**0.1586**	95.40	**0.1401**	**0.0905**	99.02
POD	0.2992	0.1403	2,170	0.1089	0.0761	1,400
w/o_d	0.3022	0.1416	**35.21**	0.1330	0.0866	**32.13**
w/o_d_TID	**0.3339**	**0.1567**	90.26	**0.1406**	**0.0908**	80.38

2.4 Discussion

In this subsection, we summarize key findings from the preliminary studies as follows: (1) Compared with BERT4Rec, LLM-based recommendations are much more time-consuming. (2) Reducing the input length will improve the inference efficiency. (3) Beam search decoding will have negative impacts on the efficiency of sequential recommendation. (4) Compared with random number, item title can better represent items and achieve better performance due to the incorporation of contextual information. (5) Despite the advantages of item title, they are usually very long and will increase computational costs. These findings provide the groundwork for us to simplify existing LLM-based recommendations and propose a simple but effective framework LITE-LLM4REC for sequential recommendations.

3 The Proposed Framework

Motivated by our findings, in this section, we aim to simplify the architecture to obtain a better sequential recommendation system, which is easier to train and can achieve low-latency inference and better performance. In this section, we introduce

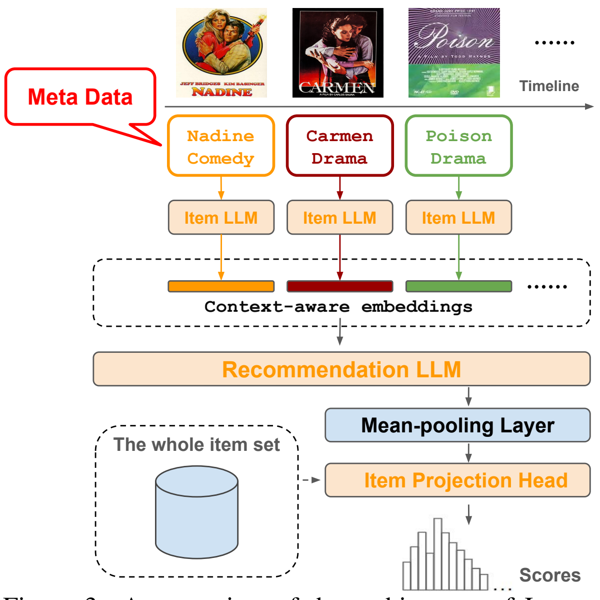

Figure 3: An overview of the architecture of LITE-LLM4REC.

the proposed framework LITE-LLM4REC. We first give an overview of LITE-LLM4REC. Then we detail its key components and finally give its training details.

3.1 An Overview

Figure 3 demonstrates the whole architecture of LITE-LLM4REC. To mitigate redundant computation and enhance inference efficiency, LITE-LLM4REC proposes a hierarchical LLM structure which contains two LLM components: Item LLM and Recommendation LLM. The Item LLM first distills verbose item contexts into compact semantic vectors. The Recommendation LLM then processes this sequence of vectors, bypassing the need to handle lengthy raw text. Finally, LITE-LLM4REC replaces the iterative beam search process with a direct item projection head, enabling the instantaneous generation of recommendation scores. The design of each component is detailed in the following subsections.

3.2 Hierarchical LLM Structure

To harness the power of LLMs, existing LLM-based recommendation methods typically formulate items into natural language using various indexing strategies. These strategies fall into two main categories. The first category encodes item relationships through indexing methods like semantic IDs (Hua et al., 2023; Li et al., 2023b; Mei and Zhang, 2023). These methods capture item relations via shared tokens but the indexing by itself doesn't contain any semantic information, which may not be able to fully explore the potential of LLMs. The second category denotes items using metadata such as titles or genres (Bao et al., 2023; Li et al., 2023a). This approach incorporates contextual information but often results in long inputs, which needs more time to process or even worse they need truncation or special architectures to process when they exceed the length limitation of LLMs.

In addition, both types of item indexing undergo tokenization before being fed into the LLM, increasing computational complexity, as each item will be represented by multiple tokens. Moreover, redundant computations occur when the same item appears multiple times in the input. For example, in the ML-1m dataset, the movie 'Star Wars: Episode I - The Phantom Menace (1999)' will be tokenized into 11 tokens by T5 (Raffel et al., 2020), and appears 539 times during inference, leading to severe redundant computations and hurting the inference efficiency.

LITE-LLM4REC aims to simplify existing item indexing while leveraging the power of LLMs. Particularly, we propose a hierarchical LLM structure comprising two distinct LLM components: Item LLM and Recommendation LLM. The Item LLM encodes extensive context information of an item into a compact, context-aware vector representation. Leveraging its capabilities, the Item LLM can effectively capture the contextual nuances and dependencies within the input sequence, facilitating the creation of a context-aware vector for each item. Then, the Recommendation LLM processes the sequence of context-aware vectors rather than the original lengthy context sequences. Thus the input length of LLM can be significantly reduced. In the following, we will give the details about acquiring context-aware vectors for items and sequence representations.

We represent the context information of item i (e.g., title, genre) after tokenization by $\mathcal{T}_i^e = (w_1, w_2, ..., w_L)$, where w denotes tokens of the context information and L is the length of the context. Then, we input $\mathcal{T}_i^e$ into the item LLM which can be denoted as:

$$Input(\mathcal{T}_i^e) = (w_1, w_2, ..., w_L). \tag{1}$$

Through the model, we can obtain the output representation for each token:

$$Output(\mathcal{T}_i^e) = (h_{w_1}, h_{w_2}, ..., h_{w_L}). \tag{2}$$

Then we can obtain the context-aware embedding for the item:

$$h_i = Mean\_pooling(h_{w_1}, h_{w_2}, ..., h_{w_L}), \tag{3}$$

where $h_{w_i} \in R^{1\times d}$ is the representation for the corresponding token w_i. Notice that deriving high quality item representation is not our focus, so we just apply a simple mean-pooling over all tokens' representation to get the final context-aware embedding for each item. More complicated representation methods will be explored as one future work.

After obtaining context-aware embeddings for items. The Recommendation LLM will take a sequence of context-aware embeddings as input instead of the lengthy natural language input. Similarly, we also apply a simple mean-pooling function to the output of the Recommendation LLM to obtain the representation of sequence:

$$h_u = Mean\_pooling(Rec\_LLM(h_1, h_2, ..., h_t)), \tag{4}$$

where h represents the context-aware embedding for an item, Rec_LLM indicates the Recommendation LLM with per-trained weights.

This architecture has several advantages: First, the Item LLM condenses an item's long-context information into a single embedding, preserving its meaning while shortening the input sequence. Second, it avoids redundant computations by directly using the precomputed item representation instead of recalculating it each time the item appears. Thus, LITE-LLM4REC enhances inference efficiency via reducing computational costs.

3.3 Item Projection Head

Our preliminary studies confirm that the beam search decoding in auto-regressive generation, inherited from natural language generation, is the the principal computational bottleneck in LLM-based recommenders. This operation is ill-suited for the sequential recommendation task, as it requires generating item identifiers (e.g., a string like 'item_1123') token-by-token. This leads to several critical inefficiencies: (1) Vocabulary Mismatch: The model's final layer computes a probability distribution over the entire LLM vocabulary, which can contain thousands of tokens. This is profoundly wasteful, as valid item tokens represent only a tiny fraction of this space. Invalid Generation: The generative process may produce syntactically plausible but non-existent item identifiers or yield repetitive outputs, necessitating complex and costly post-processing filters. Compounded Latency: Generating a slate of k recommendations requires k separate, sequential decoding operations, which multiplies the inference time and exacerbates the efficiency problem.

To circumvent these issues entirely, LITE-LLM4REC replaces the auto-regressive generation process with a direct item projection head. This design re-frames recommendation from a slow generation task to a rapid ranking task. The projection head takes the final user representation and directly computes a probability distribution over the entire valid item set. These probabilities are treated as ranking scores, from which the top items are selected as the final recommendations. In particular, we implement this head as a computationally lightweight one-layer Multi Layer Perceptron (MLP) (Kang et al., 2023), which can be formulated as follows:

$$logits = W_{proj} h_u, \tag{5}$$

where h_u stands for sequence representation for user u obtained by Eq. 4, W_{proj} is the projection matrix of the MLP, and $logits$ represents the output scores over the whole item set.

3.4 Model Training

For the training phase, we consider all items that a user has not interacted with as negative samples. We train the hierarchical model with a cross-entropy loss as shown below:

$$L_{CE} = -\sum_{i=1}^{N} y_i log r_i, \tag{6}$$

where N is the number of items, y_i represents the ground-truth for item i, which is 1 if item i is the ground-truth item, otherwise 0; and r_i is the predicted score of item i. We provide the training procedure in Appendix B. The time complexity of the algorithm can be found in Appendix F. We adopt two training strategies which are denoted as 'LITE-LLM4REC _sampling' and 'LITE-LLM4REC _all' in Table 3 and their details are given in 4.1.4.

4 Experiment

In this section, we conduct comprehensive experiments to verify the effectiveness and efficiency of the proposed LITE-LLM4REC. In particular, we try to answer the following questions: (1) Can the proposed LITE-LLM4REC achieve better overall performance? (Section 4.2); (2) Can the proposed LITE-LLM4REC improve inference efficiency? (Section 4.3); (3) How does LITE-LLM4REC perform on Top-N recommendation task? (Appendix I); (4) How do different components of LITE-LLM4REC affect the recommendation performance? (Section 4.6)

4.1 Experimental Settings

4.1.1 Datasets

To evaluate the effectiveness of LITE-LLM4REC, we conduct a series of experiments on four real-world benchmark datasets, including ML-1m[2] (Harper and Konstan, 2015), Amazon-Movies and TV, Amazon-Toys&Games and Amazon-Kindle Store[3] (Ni et al., 2019). We partition them into training, validation and test sets with the commonly used leave-one-out strategy. It takes the second-to-last item as the validation item, the last item as the test item and all other items as training items in each user's interaction history. Additional details of the datasets can be found in the Appendix C.

4.1.2 Evaluation protocols

We adopt two widely used metrics Recall@k and NDCG@k, where $k = 10, 20$. Recall@k represents the coverage of ground-truth items that appear in the final recommendation list. NDCG@k measures the ranking quality of the final recommendation items. For both metrics, a larger value indicates better performance. For our method and the baselines, we evaluate the performance on the whole item set, and the reported results are the average values over all users.

4.1.3 Baselines

We choose representative methods from three groups as baselines, i.e., traditional ID-based sequential models, Context-aware ID-based models, LLM-based recommendation models. We consider the following tradition ID-based sequential models: **GRU4Rec** (Hidasi et al., 2015), **SASRec** (Kang and McAuley, 2018), **BERT4Rec** (Sun et al., 2019), **NARM** (Li et al., 2017), **STAMP** (Liu et al., 2018). The context-aware ID-based sequential models include: S^3**-Rec** (Zhou et al., 2020), **FDSA** (Zhang et al., 2019). We choose the following LLM-based sequential models: **P5** (Geng et al.,

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

[3] https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews

Table 3: Overall performance comparison of on ML-1m, Movies, Toys and Kindle with conventional sequential baselines and LLM-based sequential methods. R denotes Recall, and N denotes NDCG. Improv. indicates the improvements over the best baseline models. Boldface represents the best results. Underscore indicates the second best results. A significant improvement over the best baseline is marked with * ($p < 0.05$)

Datasets	ML-1m				Movies			
Methods	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
GRU4Rec	0.1876	0.2833	0.0999	0.1241	0.1153	0.1495	0.0808	0.0894
BERT4Rec	0.1981	0.2892	0.1093	0.1323	0.0761	0.1048	0.0493	0.0565
SASRec	0.1834	0.2785	0.0801	0.1040	0.1082	0.1449	0.0714	0.0807
NARM	0.1803	0.2717	0.0917	0.1146	0.1137	0.1483	0.0795	0.0882
STAMP	0.1579	0.2325	0.0828	0.1015	0.0937	0.1203	0.0682	0.0749
S^3-Rec	0.1776	0.2733	0.0882	0.1123	0.0764	0.1074	0.0481	0.0560
FDSA	0.1962	0.2854	0.1058	0.1283	0.1151	0.1500	0.0804	0.0891
P5	0.2149	0.2985	0.1232	0.1442	0.0905	0.1080	0.0718	0.0761
POD	0.2185	0.2992	0.1201	0.1403	0.0904	0.1089	0.0715	0.0761
LightLM	0.1705	0.2531	0.0928	0.1135	0.0751	0.0958	0.0521	0.0574
LLM2Rec	0.2225	0.3165	0.1137	0.1374	0.0731	0.0878	0.0443	0.0480
Lite-LLM4Rec_sampling	0.2733*	0.3770*	0.1518*	0.1780*	0.1156	0.1447	0.0856*	0.0929*
Lite-LLM4Rec_all	**0.3209****	**0.4255****	**0.1866****	**0.2129****	**0.1253****	**0.1596****	**0.0906****	**0.0992****
Improv.	44.22%	34.43%	51.46%	47.64%	8.67%	6.4%	12.12%	10.96%
Datasets	Toys				Kindle			
Methods	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
GRU4Rec	0.0506	0.0772	0.0264	0.0331	0.1132	0.1524	0.0695	0.0794
BERT4Rec	0.0352	0.0543	0.0179	0.0227	0.0801	0.1143	0.0445	0.0531
SASRec	0.0561	0.0776	0.0312	0.0366	0.1009	0.1416	0.0586	0.0688
NARM	0.05	0.0791	0.0249	0.0323	0.1161	0.1565	0.0693	0.0794
STAMP	0.0494	0.0735	0.0275	0.0336	0.1077	0.1339	0.0747	0.0814
S^3-Rec	0.0538	0.0811	0.0276	0.0345	0.0408	0.0684	0.0198	0.0266
FDSA	0.0568	0.0860	0.0344	0.0417	0.1170	0.1640	0.0646	0.0764
P5	0.0543	0.0661	0.0356	0.0416	0.0748	0.0955	0.0484	0.0534
POD	0.0565	0.0690	0.0421	0.0452	0.0624	0.0813	0.0399	0.0444
LightLM	0.0509	0.0721	0.0302	0.0355	0.0571	0.0806	0.0323	0.0382
LLM2Rec	0.0486	0.0698	0.0244	0.0298	0.0695	0.0846	0.359	0.0397
Lite-LLM4Rec _sampling	**0.0682****	**0.0927****	**0.0426**	**0.0488****	**0.1351**	0.1673	**0.0905**	**0.0986**
Lite-LLM4Rec _all	0.0627*	0.0894	0.0382	0.0449	0.1314	**0.1686**	0.0863	0.0957
Improv.	20.07%	7.79%	1.18%	7.96%	15.47%	2.80%	30.21%	21.13%

2022), **POD** (Li et al., 2023b), **LightLM** (Mei and Zhang, 2023), **LLM2Rec** (He et al., 2025). Since **llamaRec (Yue et al., 2023)** is a two-stage framework while LITE-LLM4REC is single-stage, we do not choose it as one baseline. The details of baselines can be found in Appendix D.

4.1.4 Implementation details

We use T5-small from Huggingface[4] as our backbone in the main experiments. Following (Li et al., 2023b), we randomly sample a segment of no more than 21 items from a user's interaction history for each iteration which is denoted as 'LITE-LLM4REC _sampling' in Table 3. We also implement another training strategy where we traverse all the training data without sampling which is represented as 'LITE-LLM4REC _all'. Further implementation details can bu found in Appendix E. The encoder and decoder in this model both have 6 layers, each of which is an 8-headed attention layer. We find that after training, further fine-tuning the context-aware embeddings and the Recommendation LLM will result in better performance. We investigate the impact of different Item LLM backbones in Section 4.4, while in Section 4.5, we explore the scaling behavior of Recommendation LLM.

4.2 Effectiveness Comparison

The performance comparison is shown in Table 3. From the results, we can make the following observations: 1) LITE-LLM4REC exhibits significantly better performance than LLM-based recommendation baselines. Notably, the average Recall@10 improvements over the best results of LLM-based recommendation baselines are 44.22% on ML-1m, 38.4% on Movies, 20.7% on Toys and 80.61% on Kindle, highlighting the effectiveness of our design. The potential reasons for the performance improvement are two-fold. First, the item projection head in LITE-LLM4REC is more efficient, as it scores only valid items. This avoids the wasted computation of standard beam search, which must consider

[4]https://huggingface.co/t5-small

the entire, much larger LLM vocabulary. Second, our hierarchical structure encodes items into dense vectors, which provide a more flexible and effective input for the recommendation model than the original natural language text. 2) LITE-LLM4REC consistently outperforms the best performance over baselines. The improvement on NDCG@20 is approximately 51.4% on ML-1m, 11.3% on Movies, 7.9% on Toys and 29.05% on Kindle, which can be attributed to the power of LLMs. Notice that traditional ID-based and context-aware recommendation algorithms are still competitive.

4.3 Efficiency Comparison

In this subsection, we analyze the inference efficiency of LITE-LLM4REC. Since training and operations like obtaining context-aware embeddings can be done off-line, we just consider the time between inputting the data to the model and obtaining the final recommendations as the inference time. The time comparison for a batch of 32 users are shown in Table 4. Apparently, LITE-LLM4REC can achieve superior inference efficiency. The improvement over LightLM is approximately 99.48% on ML-1m, 99.57% on Movies, 99.57% on Toys and 99.47% on Kindle. We contribute the efficiency improvements to the following reasons. First, we remove the most time-consuming part - beam search decoding. Second, the hierarchical LLM structure we propose to process long context can mitigate the redundant computation problem and reduce the computational costs. We also report the comparison of input length in Table 5. As can be seen, the advantage of LITE-LLM4REC is evident, as the improvement of input length is 75.2%, 76.7%, 75.0% and 78.0% over POD on four datasets, respectively.

Table 4: Comparison of inference time (ms)

Datasets	ML-1m	Movies	Toys	Kindle
P5	1,646	1,596	1,620	1,549
POD	1,468	1,400	1,490	1,468
LightLM	1,196	1,209	1,190	1,131
LITE-LLM4REC	**6.13**	**5.14**	**5.03**	**5.95**
Improv.	99.48%	99.57%	99.57%	99.47%

4.4 Impact of Item LLMs

In order to explore whether the performance gains come from the alignment between the Item LLM and the Recommendation LLM (both of them are T5), we conduct experiments with other Item

Table 5: Comparison of average input length.

Datasets	ML-1m	Movies	Toys	Kindle
P5	112.2	112.1	107.1	117.1
POD	85.0	89.1	83.5	94.67
LightLM	170.8	174.3	166.6	194.3
LITE-LLM4REC	**21**	**20.7**	**20.8**	**20.8**
Improv.	75.2%	76.7%	75.0%	78.0%

LLM backbones like Bert (Devlin et al., 2018), Sbert (Reimers and Gurevych, 2019) and T5-base (Raffel et al., 2020). For Bert, we use 'bert-base-uncased' version and take the output of the pooler layer as the sequence representation. For Sbert, we use 'all-MiniLM-L6-v2' version. Since the dimension of hidden state is not matched, one-layer MLP is adopted to transform the dimension. The results are reported in Table 6. We can find that other Item LLMs can also achieve satisfying performance, which indicates that the influence of the backbones of Item LLMs is limited.

Table 6: Impact of Item LLMs

ML-1m	R@10	R@20	N@10	N@20
LITE-LLM4REC	0.2733	0.3770	0.1518	0.1780
T5-base	0.2728	0.3742	0.1512	0.1767
Bert	0.2706	0.3749	0.1529	0.1793
Sbert	0.2668	0.3707	0.1486	0.1748

4.5 Scaling Capabilities

In this section, we evaluate the scalability of LITE-LLM4REC by experimenting with larger Recommendation LLM backbones, namely T5-base[5] and T5-large[6]. The results, presented in Table 7, confirm LITE-LLM4REC's scaling capabilities, showing that performance improves as the backbone size increases.

Table 7: Impact of Item LLMs

ML-1m	R@10	R@20	N@10	N@20
T5-small	0.2733	0.3770	0.1518	0.1780
T5-base	0.2761	0.3771	0.1572	**0.1827**
T5-large	**0.2824**	**0.3773**	**0.1577**	0.1816

4.6 Ablation Study

In this section, we aim to analyse how different components influence the overall performance. We conduct experiments on ML-1m and Toys datasets

[5]https://huggingface.co/google-t5/t5-base
[6]https://huggingface.co/google-t5/t5-large

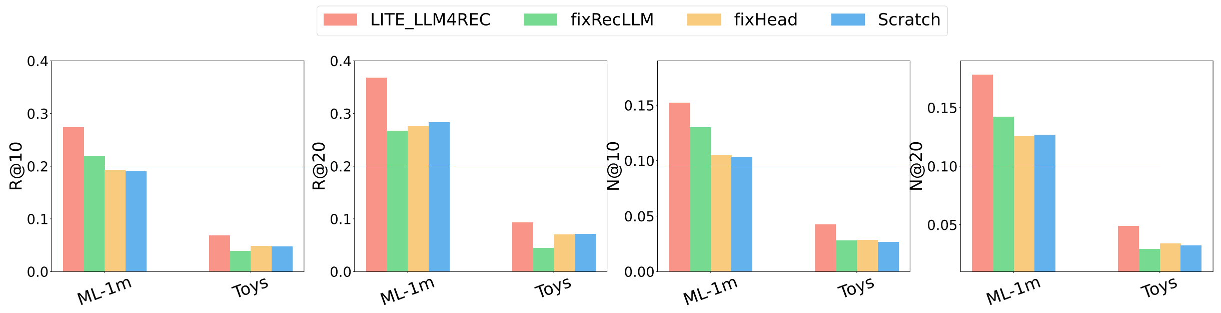

Figure 4: Ablation study results of R@10, R@20, N@10 and N@20 on ML-1m and Toys dataset.

with LITE-LLM4REC _sampling strategy to assess each component. The results on Movies and Kindle can be found in Appendix H.1. We design the following variants of our model: (1) fixT5: Recommendation LLM (the encoder of T5) is fixed. (2) fixHead: item projection head is fixed. (3) Scratch: the parameters of the Recommendation LLM is randomly initialized instead of loading pre-trained weights.

From the results shown in Figure 4, we can have the following observations. First, each component in our framework contributes to the overall performance since fixing any one of the components results in a performance drop. Second, LLM's knowledge helps the recommendation tasks since training a T5 from scratch instead of using the pre-trained weights leads to a drop in performance.

5 Conclusion

In this work, we propose LITE-LLM4REC, an effective and efficient LLM-based model for sequential recommendation. We found that the standard beam search decoding process is an unnecessary bottleneck, so we replace it with a direct item projection head for faster ranking. To handle long item descriptions without high computational cost, we also introduce a novel hierarchical LLM structure. These two innovations solve key inference problems in LLM-based recommenders. Experiments on four real-world datasets show that LITE-LLM4REC significantly improves both inference speed and overall performance. Future work will explore inductive learning, the impact of item indexing during training, and how the model's backbone size affects performance.

6 Limitations

This work has several limitations that should be considered when interpreting the results: (1) Lack of Inductive Learning Ability: Due to the existence of the item projection head, our model cannot adapt to new items that do not appear in the training data. As a result, the model is unable to generalize to unseen items effectively, which could be a limitation in real-world recommendation systems where new items are constantly introduced. (2) Exclusion of Larger Language Models: Our experiments did not test larger LLMs, such as those with 70B parameters, which could offer superior performance due to their increased capacity to process and generate predictions. The exclusion of these models limits the generalizability of our findings, as larger models may be able to handle more complex patterns, nuances, and domain-specific knowledge. Furthermore, there may be diminishing returns for extremely large models, and understanding the trade-offs between model size and performance is crucial. (3) High Training Costs: Compared to traditional sequential recommendation models, the proposed hierarchical LLMs face high training costs. These costs arise not only from the large number of parameters in LLMs but also from the extensive computational resources required for both training and inference.

7 Acknowledgements

Hanbing Wang and Hui Liu are supported by the National Science Foundation (NSF) under grant numbers CNS2321416, IIS2212032, IIS2212144, IIS 2504089, DUE2234015, CNS2246050, DRL2405483 and IOS2035472, the Michigan Department of Agriculture and Rural Development, US Dept of Commerce, Gates Foundation, Amazon Faculty Award, Meta, NVIDIA, Microsoft and SNAP.

References

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*.

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. *arXiv preprint arXiv:2305.00447*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

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*.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318*.

Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1096–1102.

Ruining He, Wang-Cheng Kang, Julian J McAuley, et al. 2018. Translation-based recommendation: A scalable method for modeling sequential behavior. In *IJCAI*, pages 5264–5268.

Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 191–200. IEEE.

Yingzhi He, Xiaohao Liu, An Zhang, Yunshan Ma, and Tat-Seng Chua. 2025. Llm2rec: Large language models are powerful embedding models for sequential recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 896–907.

Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 843–852.

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

Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 241–248.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 585–593.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. *arXiv preprint arXiv:2305.06569*.

Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 573–581.

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

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

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023a. Text is all you need: Learning language representations for sequential recommendation. *arXiv preprint arXiv:2305.13731*.

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

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

Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. 2023c. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. *arXiv preprint arXiv:2305.11700*.

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

Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*.

Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1053–1058. IEEE.

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1831–1839.

Qidong Liu, Xian Wu, Wanyu Wang, Yejing Wang, Yuanshao Zhu, Xiangyu Zhao, Feng Tian, and Yefeng Zheng. 2025. Llmemb: Large language model can be a good embedding generator for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12183–12191.

Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2018. A contextual attention recurrent architecture for context-aware venue recommendation. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 555–564.

Kai Mei and Yongfeng Zhang. 2023. Lightlm: A lightweight deep and narrow language model for generative recommendation. *arXiv preprint arXiv:2310.17488*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Junyan Qiu, Haitao Wang, Zhaolin Hong, Yiping Yang, Qiang Liu, and Xingxing Wang. 2023. Controlrec: Bridging the semantic gap between language model and personalized recommendation. *arXiv preprint arXiv:2311.16441*.

Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-bert: Pre-training user representations for improved recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4320–4327.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *arXiv preprint arXiv:2305.05065*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, 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*, pages 1441–1450.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agent. *arXiv preprint arXiv:2304.09542*.

Yuqi Sun, Qidong Liu, Haiping Zhu, and Feng Tian. 2025. Llmser: Enhancing sequential recommendation via llm-based data augmentation. *arXiv preprint arXiv:2503.12547*.

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*, pages 565–573.

Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2023. Large language models can accurately predict searcher preferences. *arXiv preprint arXiv:2309.10621*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 346–353.

An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2019. Cosrec: 2d convolutional neural networks for sequential recommendation. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2173–2176.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhengyi Yang, Jiancan Wu, Yanchen Luo, Jizhi Zhang, Yancheng Yuan, An Zhang, Xiang Wang, and Xiangnan He. 2023. Large language model can interpret latent space of sequential recommender. *arXiv preprint arXiv:2310.20487*.

Weihua Yuan, Hong Wang, Xiaomei Yu, Nan Liu, and Zhenghao Li. 2020. Attention-based context-aware sequential recommendation model. *Information Sciences*, 510:122–134.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id- vs. modality-based recommender models revisited. *arXiv preprint arXiv:2303.13835*.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. Llamarec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089*.

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*.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326.

Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *proceedings of the 30th acm international conference on information & knowledge management*, pages 4653–4664.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902.

A Background

Sequential Recommendation Sequential recommendations (Kang and McAuley, 2018; Sun et al., 2019) leverage user historical interactions to infer the next item the user will interact with. Since user interests are dynamic and evolving over time, it is important to capture this sequential pattern and provide appropriate recommendations. Early studies mainly depend on Markov Chain to model item transition patterns (Rendle et al., 2010; He and McAuley, 2016). Recently, deep-learning based methods have dominated the area. For example, GRU4rec (Hidasi et al., 2015) proposes to use RNN in session-based recommendation. Some approaches introduce CNN into sequential recommendation (Tang and Wang, 2018; Yan et al., 2019). SASrec and BERT4Rec (Kang and McAuley, 2018; Sun et al., 2019) employ the self-attention mechanism into sequential recommendation and achieve excellent performance. However, these methods neglect rich contextual information about items, which is important for item modeling. To tackle this problem, several algorithms are proposed (Yuan et al., 2020; Liu et al., 2016; Manotumruksa et al., 2018; Zhang et al., 2019; Zhou et al., 2020). For example, FDSA (Zhang et al., 2019) proposes a self-attention block to leverage item attribute information. S^3-rec (Zhou et al., 2020) maximizes the mutual information of context information in different forms to improve sequential recommendation. Despite the remarkable improvements made by these methods, they can still be further improved via the excellent world knowledge and context understanding ability of LLMs.

LLM-based Recommendation The prevalence of LLMs has introduced a paradigm shift into recommender systems (Fan et al., 2023; Liu et al., 2025; Sun et al., 2025). Early approaches investigate the practicality of textual representations generated by a language model for recommendations (Yuan et al., 2023; Li et al., 2023c; Qiu et al., 2021; Harte et al., 2023). By pre-training and then fine-tuning on downstream datasets, enhanced representations can be obtained (Li et al., 2023a; Ding et al., 2021;

Hou et al., 2022). The emergence of generative LLMs shifts recommendation system towards generative paradigm (Rajput et al., 2023). Early attempts explore the potential of LLMs on recommendation via prompt or in-context learning (Sun et al., 2023; Liu et al., 2023; Thomas et al., 2023). TallRec (Bao et al., 2023) trains the LLM to predict whether a user will like a new item given users' interaction history, underscoring the importance of instruction tuning. P5 (Geng et al., 2022) reformulates several recommendation tasks into a natural language generation task via personalized prompts. Refer to Figure 1 for an overall understanding of its architecture.

In order to leverage the power of LLMs, items are usually represented in natural language form via specific item indexing methods, which can be roughly divided into two categories. The first category proposes to reflect item relations via shared tokens, such as semantic IDs (Hua et al., 2023; Mei and Zhang, 2023; Deng et al., 2025). Given the continued significance of ID information, recent studies keep the IDs and represent items as 'item_1234', which will be tokenized into a token sequence 'item','_','12','34' before being input to LLMs (Li et al., 2023b; Qiu et al., 2023; Geng et al., 2022). The second category represents items via context information. Tallrec (Bao et al., 2023) uses item title to represent items. RECFORMER (Li et al., 2023a) flattens the key-value attribute pairs of the item as a sequence to accommodate more textual information, and employs a Longformer (Beltagy et al., 2020) to process the long context.

Although these methods have achieved remarkable improvements, they still face slow inference problem. Recent studies have delved into this issue. POD (Li et al., 2023b) improves P5 (Geng et al., 2022) by distilling discrete prompt into continuous prompt to reduce the input length, thus reducing inference time. llamaRec (Yue et al., 2023) proposes a two stage framework. Specifically, it retrieves candidate items via traditional ID-based methods, and designs a verbalizer approach for reranking. E4SRec (Li et al., 2023d) proposes to integrate ID embeddings extracted from a pre-trained SASRec with instruction prompt but neglects the rich context information of items. LightLM (Mei and Zhang, 2023) proposes a tailored transformer-based architecture to achieve effective recommendations. It improves the inference efficiency by reducing the number of neurons thus reducing the computation demands. However, the modification of architecture may destroy the pre-trained knowledge of LLMs, leading to sub-optimal performance. Despite the strides made in improving inference efficiency, these methods still fall short in addressing the most time-consuming component and the issue of redundant computation.

B Training Algorithms

In this section, we provide the training algorithm of LITE-LLM4REC in Algorithm 1.

Algorithm 1: Training Process of LITE-LLM4REC

```
Input: The sequential data $\mathcal{D}$, Context
       information of items $\mathcal{T}$,
       hyper-parameter settings;
Output: Model parameters $\theta$;
1  while not coverage do
2      Draw a batch of data $\mathcal{B}$ from $\mathcal{D}$;
3      for $(u, \mathcal{I}_u)$ in $\mathcal{B}$ do
4          Sample an item segment $I_u^{seg}$ from
             $\mathcal{I}_u^{|s_u|-2}$;
5          for $i$ in $I_u^{seg}$ do
6              Acquire context information $\mathcal{T}_i$;
7              Calculate context-aware
                 embeddings $h_i$ through Eq. 1 -
                 Eq. 3;
8          end
9          Acquire the user representation $h_u$
             through Eq. 4;
10     end
11     Compute logits through Eq. 5 ;
12     Compute $L_{CE}$ through Eq. 6;
13     Update the parameters $\theta$ of the model
         through $L_{CE}$ ;
14 end
```

C Datasets

Table 8 shows the statistics of these datasets. Additional details of datasets are as follows:

- The MovieLens-1m dataset is an open dataset for movie recommendations[7]. There are approximately 100k interactions. We adopt 5-core filtering strategy where we filter out users and items with less than 5 interactions.
- We consider three categories of Amazon dataset corpora: Movies and TV,

[7]https://movielens.org/

Table 8: Statistics of the datasets after pre-processing.

Dataset	Users	Items	Interact	Sparsity(%)
ML-1M	6,040	3,706	994,169	0.0446
Movies	79,276	29,946	1,775,011	0.0007
Toys	11,803	8,569	206,103	0.0020
Kindle	40,981	39,520	1,149,411	0.0007

Toys&Games (denoted as Toys for clarity) Kindle Store (denoted as Kindle for clarity). These datasets are collected from the e-commerce platform Amazon[8] with item meta data, user reviews and ratings. We adopt 10-core filtering strategy to filter the users and items with less than 10 interactions to ensure data quality.

D Baselines

In this section, we provide details concerning the baseline models. We consider the following tradition ID-based sequential models: (i) **GRU4Rec** (Hidasi et al., 2015) adapts the RNN models to the recommender setting by introducing a new ranking loss function. (ii) **SASRec** (Kang and McAuley, 2018) proposes an unidirectional attention-based sequential model which can capture long-term semantics to predict the next item. (iii) **BERT4Rec** (Sun et al., 2019) introduces a bidirectional attention-based transformer to model user behavior sequences. It introduces the Cloze objective into sequential recommendations. (iv) **NARM** (Li et al., 2017) considers both the user's sequential behavior and main purpose in the current session, and calculate recommendation scores by using a bi-linear matching approach. (v) **STAMP** (Liu et al., 2018) introduces a short--term attention and memory priority model that learns a unified embedding space for items across sessions, coupled with an innovative neural attention mechanism for next-click prediction.

The context-aware ID-based sequential models include: (i) S^3**-Rec** (Zhou et al., 2020) applies self-supervised learning to the sequential recommendation task. It proposes four self-supervised optimization objectives to maximize the mutual information of context information to learn the correlation between items. (ii) **FDSA** (Zhang et al., 2019) proposes to model feature transitions through different self-attention blocks. It integrates with item-level transitions for modeling user's sequential intents.

We choose the following LLM-based sequential models: (i) **P5** (Geng et al., 2022) transforms various recommendation tasks into the conditional natural language generation task via personalized prompts and integrates them into a unified framework. (ii) **POD** (Li et al., 2023b) proposes to distill knowledge in the discrete prompt into continuous prompt vectors, which is more flexible and expressive and can reduce the inference time. (iii) **LightLM** (Mei and Zhang, 2023) proposes a tailored Transformer-based recommender, which is effective and efficient for generative recommendations. (iv) **LLM2Rec** (He et al., 2025) proposes to integrate the rich semantic understanding of LLMs with collaborative filtering awareness. Since **llamaRec (Yue et al., 2023)** is a two-stage framework while LITE-LLM4REC is single-stage, we do not choose it as one baseline.

E Implementation Details

We set the batch_size for three datasets to 256 and the learning rate to 0.0005. The embedding dimension is set to 512. The dropout rate is 0.8 and the weight_decay is 0.1 for Movies and Toys. The dropout rate is 0.7 for ML-1m. The dropout rate is 0.5 for Kindle. The warm_up rate is set to 0.1 for Movies and Toys, 0 for ML-1m and Kindle respectively. The adam_eps is set to 1e-6. All methods are implemented using Pytorch with an AdamW optimizer. For GRU4Rec, NARM, STAMP, S^3- Rec and FDSA, we implement them using the public resources released by their authors. For other models, we adopt a popular open-source recommendation library RecBole[9] (Zhao et al., 2021). We check the validation performance every epoch and adopt early-stop when the validation performance of R@10 doesn't improve for 10 consecutive times.

F Time Complexity Analysis

The section gives time complexity analysis to demonstrate the efficiency of our method which is additional to our empirical observations:

- The time complexity of self-attention based methods is $O(n^2 d + n d_{ff} d)$, where n represents the input length, d_{ff} and d represent the dimension of feed forward layers and embedding dimension respectively. In recommendation, the total inference cost scales as $\sum_u \{(\sum_{i \in N_u} n_i)^2 d + (\sum_{i \in N_u} n_i) d^2\}$, where

[8]https://www.amazon.com/

[9]https://www.recbole.io/

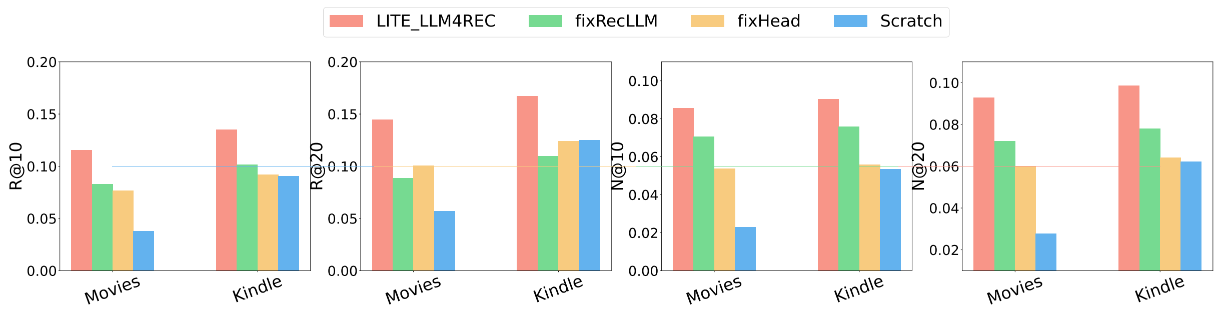

Figure 5: Ablation study results of R@10, R@20, N@10 and N@20 on Movies and Kindle dataset.

u and i represent user u and item i, and N_u and n_i represent the interaction history of user u and the number of tokens to represent item i. LITE-LLM4REC can compress the input and reduce it to $\sum_u \{|N_u|^2 d + |N_u| d^2\}$, where $|N_u|$ denotes the length of N_u and is much smaller than $\sum_{i \in N_u} n_i$.

- The model will redundantly compute the overall representation of an item every time it appears, which scales as $\sum_i (A_i n_i)$, where A_i denotes the number of appearance of item i. It will be reduced to $\sum_i (A_i)$ since n_i is reduced to 1.

- The time complexity of auto-regressive decoding is $O(n^3 d)$, where n denotes the input length and d is the embedding dimension. Since we totally remove it, it is reduced to $O(1)$.

G Additional Preliminary Results

In this section, we provide additional preliminary results which are reported in Table 9. We can reach the same conclusions as Section 2.4: (1) Reducing the input length will improve the inference efficiency. (2) Beam search decoding will have negative impacts on the efficiency of sequential recommendation. (3) Compared with random numbers, item titles can better represent items and achieve better performance due to the incorporation of contextual information. (4) Despite the advantages of item titles, they are usually very long and will increase computational costs.

H Additional Ablation Study

H.1 Additional Datasets

In this section, we give additional results of ablation experiments conducted on Movies and Kindle dataset. The results are reported in Figure 5. The observations align with Section 4.6. First, each component in our framework contributes to the overall performance since fixing any one of them results in the performance drop. Second, if we train a T5 from scratch instead of using the pretrained weights for the recommendation LLM, the performance will drop which demonstrates that the pretrained knowledge stored in LLM is of help to the recommendation task.

Table 9: Performance of P5 and POD and their two variants

Datasets	Toys			Kindle		
Methods	R@20	N@20	Time	R@20	N@20	Time
P5	0.0661	0.0416	1,620	0.0955	0.0534	1,247
w/o_d	0.0711	0.0370	**52.61**	0.1082	0.0554	**28.03**
w/o_d_TID	**0.0886**	**0.0460**	148.61	**0.1288**	**0.0691**	113.20
POD	0.0690	0.0452	1,490	0.0813	0.0444	1,209
w/o_d	0.0679	0.0345	**43.54**	0.0908	0.0497	**22.50**
w/o_d_TID	**0.0911**	**0.0468**	134.11	**0.1284**	**0.0687**	109.16

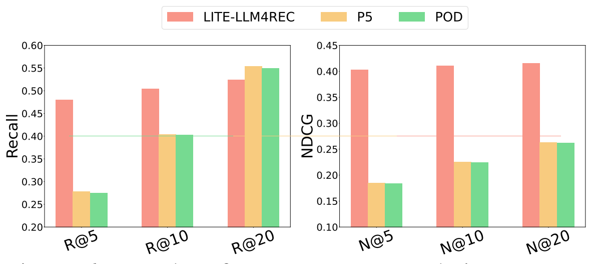

Figure 6: Results of Top-N recommendation on ML-1m.

I Top-N Recommendation

Besides sequential recommendation, we also apply LITE-LLM4REC to Top-N recommendation task. We follow the setting of POD (Li et al., 2023b). We input one ground-truth item along with 99 negative items to the LLM and fine-tune the model to predict the ground-truth item. Finally, We test the model over the 99 negative examples. The results are reported in Figure 6.

From the figure, we can find that LITE-LLM4REC can also perform well on this task. Our method can achieve much better performance on

NDCG than P5 (Geng et al., 2022) and POD (Li et al., 2023b). This indicates that our method can greatly improve the ranking quality especially when candidates are given. For Recall, our method can perform better when k is small. This is because the leave-one-out strategy we adopt where we only have one ground-truth item and our method can already rank the ground truth item in high position.