Lost in Sequence: Do Large Language Models Understand Sequential Recommendation?

Sein Kim*
rlatpdlsgns@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Hongseok Kang*
ghdtjr0311@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Kibum Kim
kb.kim@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Jiwan Kim
kim.jiwan@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Donghyun Kim
amandus.kim@navercorp.com
NAVER Corperation
Seongnam, Republic of Korea

Minchul Yang
minchul.yang@navercorp.com
NAVER Corperation
Seongnam, Republic of Korea

Kwangjin Oh
kj.oh@navercorp.com
NAVER Corperation
Seongnam, Republic of Korea

Julian McAuley
jmcauley@ucsd.edu
University of California San Diego
California, USA

Chanyoung Park†
cy.park@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Abstract

Large Language Models (LLMs) have recently emerged as promising tools for recommendation thanks to their advanced textual understanding ability and context-awareness. Despite the current practice of training and evaluating LLM-based recommendation (LLM4Rec) models under a sequential recommendation scenario, we found that whether these models understand the sequential information inherent in users' item interaction sequences has been largely overlooked. In this paper, we first demonstrate through a series of experiments that existing LLM4Rec models do not fully capture sequential information both during training and inference. Then, we propose a simple yet effective LLM-based sequential recommender, called LLM-SRec, a method that enhances the integration of sequential information into LLMs by distilling the user representations extracted from a pre-trained CF-SRec model into LLMs. Our extensive experiments show that LLM-SRec enhances LLMs' ability to understand users' item interaction sequences, ultimately leading to improved recommendation performance. Furthermore, unlike existing LLM4Rec models that require fine-tuning of LLMs, LLM-SRec achieves state-of-the-art performance by training only a few lightweight MLPs, highlighting its practicality in real-world applications. Our code is available at https://github.com/Sein-Kim/LLM-SRec.

CCS Concepts

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

Keywords

Recommender System, Large Language Models, Sequence modeling

*Both authors contributed equally to this research.
†Corresponding author.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737035

ACM Reference Format:
Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. 2025. Lost in Sequence: Do Large Language Models Understand Sequential Recommendation?. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737035

1 Introduction

Early efforts in LLM-based recommendation (LLM4Rec), such as TALLRec [2], highlighted a gap between the capabilities of LLMs in text generation and sequential recommendation tasks, and proposed to address the gap by fine-tuning LLMs for sequential recommendation tasks using LoRA [12]. Subsequent studies, including LLaRA [23], CoLLM [42], and A-LLMRec [16], criticized the exclusive reliance of TALLRec on textual modalities, which rather limited its recommendation performance in warm scenarios (i.e., recommendation scenarios with abundant user-item interactions) [16, 42]. These methods transform item interaction sequences into text and provide them as prompts to LLMs [23] or align LLMs with a pre-trained Collaborative filtering-based sequential recommender (CF-SRec), such as SASRec [13], to incorporate the collaborative knowledge into LLMs [16].

Despite the current practice of training and evaluating LLM4Rec models under a sequential recommendation scenario, we found that whether these models understand the sequential information inherent in users' item interaction sequences has been largely overlooked. Hence, in this paper, we begin by conducting a series of experiments that are designed to investigate the ability of existing LLM4Rec models in understanding users' item interaction sequences (Sec. 2.3). More precisely, we compare four different LLM4Rec models (i.e., TALLRec, LLaRA, CoLLM, and A-LLMRec) with a CF-SRec model (i.e., SASRec). Our experimental results reveal surprising findings as follows:

(1) **Training and Inference with Shuffled Sequences:** Randomly shuffling the order of items within a user's item interaction sequence breaks the sequential dependencies among items

[18, 37]. Hence, we hypothesize that the performance of models that understand the sequential information inherent in a user's item interaction sequence would deteriorate when the sequence is disrupted. To investigate this, we conduct experiments under two different settings. First, we compare the performance of models that have been trained on the original sequences (i.e., non-shuffled sequences) and those trained on randomly shuffled sequences when they are evaluated on the same test sequences in which sequential information is present (i.e., non-shuffled test sequences). Surprisingly, the performance of LLM4Rec models, even after being trained on shuffled sequences, is similar to the case when they are trained on the original sequences[1], while SASRec trained with shuffled sequences shows significant performance degradation when tested on the original sequences. Second, we perform inferences using shuffled sequences on the models that have been trained using the original sequences. Similar to our observations in the first experiment, we observed that LLM4Rec models exhibit minimal performance decline even when the sequences are shuffled during inference, while SASRec showed significant performance degradation. In summary, these observations indicate that LLM4Rec models do not fully capture sequential information both during training and inference.

(2) **Representation Similarity:** In LLM4Rec models as well as in SASRec, representations of users are generated based on their item interaction sequences. Hence, we hypothesize that user representations obtained from a model that successfully captures sequential information in users' interaction sequences would greatly change when the input sequences are disrupted. To investigate this, we compute the similarity between user representations obtained based on the original sequences and those obtained based on shuffled sequences during inference. Surprisingly, the similarity is much higher for LLM4Rec models compared with that for SASRec, meaning that shuffling users' item interaction sequences has minimal impact on user representations of LLM4Rec models. This indicates again that LLM4Rec models do not fully capture sequential information.

Motivated by the above findings, we propose a simple yet effective LLM-based sequential recommender, called LLM-SRec, a method that enhances the integration of sequential information into LLMs. The main idea is to distill the user representations extracted from a pre-trained CF-SRec model into LLMs, so as to endow LLMs with the sequence understanding capability of the CF-SRec model. Notably, our method achieves cost-efficient integration of sequential information without requiring fine-tuning of either the pre-trained CF-SRec models or the LLMs, effectively addressing the limitations of the existing LLM4Rec framework. Our main contributions are summarized as follows:

- We show that existing LLM4Rec models, although specifically designed for sequential recommendation, fail to effectively leverage the sequential information inherent in users' item interaction sequences.
- We propose a simple and cost-efficient method that enables LLMs to capture the sequential information inherent in users' item interaction sequences for more effective recommendations.
- Our extensive experiments show that LLM-SRec outperforms existing LLM4Rec models by effectively capturing sequential dependencies. Furthermore, the results validate the effectiveness of transferring pre-trained sequential information through distillation method, across various experimental settings.

[1]To address a potential concern that the moderate performance drop in LLM4Rec may be due to the prevalence of textual information over sequential data, we would like to emphasize that both types of information are indeed essential, as demonstrated in Sec. 4.3.3. On the other hand, disrupting the sequential information in users' item interaction sequences via shuffling still allows us to assess how effectively the models, particularly LLM4Rec, capture the sequential information, highlighting the importance of sequential information alongside textual data.

2 Do Existing LLM4Rec Models Understand Sequences?

2.1 Preliminaries

2.1.1 Definition of Sequential Recommendation in CF-SRec. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ represent the set of users, and $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$ represent the set of items. For a user $u \in \mathcal{U}$, $\mathcal{S}_u = (i_1^{(u)}, \ldots, i_t^{(u)}, \ldots, i_{n_u}^{(u)})$ denotes the item interaction sequence, where $i_t^{(u)} \in \mathcal{I}$ is the item that u interacted with at time step t, and n_u is the length of user u's item interaction sequence. Given the interaction history $\mathcal{S}_u$ of user u, the goal of sequential recommendation is to predict the next item that user u will interact with at time step $n_u + 1$ as $p(i_{n_u+1}^{(u)} \mid \mathcal{S}_u)$.

2.1.2 LLM for Sequential Recommendation. Note that existing LLM4Rec models can be largely categorized into the following two approaches: *Generative Approach (i.e., Next Item Title Generation)* [11, 16, 23] and *Retrieval Approach* (*i.e., Next Item Retrieval*) [6, 22]. In the Next Item Title Generation approach, a user's item interaction sequence and a list of candidate items are provided as input prompts to LLMs after which the LLMs *generate* one of the candidate item titles as a recommendation. Meanwhile, the Next Item Retrieval approach extracts user and candidate item representations from the LLMs and *retrieves* one of the candidate items whose similarity with the user representation is the highest. Note that although existing LLM4Rec models have typically been proposed based on only one of the two approaches, we apply both approaches to each LLM4Rec baseline to conduct more comprehensive analyses on whether existing LLM4Rec models understand the sequential information inherent in users' item interaction sequences.

1) Generative Approach (Next Item Title Generation). LLM4Rec models designed for Next Item Title Generation perform recommendations using instruction-based prompts as shown in Table 1. For a user u, the candidate item set of user u is represented as $\mathcal{C}_u = \left\{i_{n_u+1}^{(u)}\right\} \cup \mathcal{N}_u$, where $\mathcal{N}_u = \text{RandomSample}(\mathcal{I} \setminus (\mathcal{S}_u \cup \left\{i_{n_u+1}^{(u)}\right\}), m)$ is a negative item set for user u, and $m = |\mathcal{N}_u|$ is the number of negative items. Based on the item interaction sequence of user u, i.e., $\mathcal{S}_u$, and the candidate item set, i.e., $\mathcal{C}_u$, we write the input prompt $\mathcal{P}^u$ following the format shown in Table 1. Note that we introduce two projection layers, i.e., $f_{\mathcal{I}}$ and $f_{\mathcal{U}}$, each of which is used to project item embeddings and user representations extracted from a pre-trained CF-SRec into LLMs, respectively. Following the completed prompts shown in Table 1, LLMs are trained for the sequential recommendation task through the Next Item Title Generation approach. Note that TALLRec, LLaRA, and CoLLM use LoRA [12] to

Table 1: An example prompt for various LLM4Rec models (Next Item Title Generation approach).

	(a) **TALLRec**	(b) **LLaRA**	(c) **CoLLM/A-LLMRec**
Inputs ($\mathcal{P}^u$)	This user has made a series of purchases in the following order: (History Item List: [No.# Time: YYYY/MM/DD Title: Item Title]). Choose one "Title" to recommend for this user to buy next from the following item "Title" set: [Candidate Item Titles].	This user has made a series of purchases in the following order: (History Item List: [No.# Time: YYYY/MM/DD Title: Item Title, Item Embedding]). Choose one "Title" to recommend for this user to buy next from the following item "Title" set: [Candidate Item Titles, Item Embeddings].	This is user representation from recommendation models: [User Representation], and this user has made a series of purchases in the following order: (History Item List: [No.# Time: YYYY/MM/DD Title: Item Title, Item Embedding]). Choose one "Title" to recommend for this user to buy next from the following item "Title" set: [Candidate Item Titles, Item Embeddings].
Outputs ($\text{Text}(i^{(u)}_{n_u+1})$)	Item Title	Item Title	Item Title

Table 2: An example prompt for various LLM4Rec models (Next Item Retrieval approach).

	(a) **TALLRec**	(b) **LLaRA/LLM-SRec (Ours)**	(c) **CoLLM/A-LLMRec**
User ($\mathcal{P}^u_{\mathcal{U}}$)	This user has made a series of purchases in the following order: (History Item List: [No.# Time: YYYY/MM/DD Title: Item Title]). Based on this sequence of purchases, generate user representation token: [UserOut].	This user has made a series of purchases in the following order: (History Item List: [No.# Time: YYYY/MM/DD Title: Item Title, Item Embedding]). Based on this sequence of purchases, generate user representation token: [UserOut].	This is user representation from recommendation models: [User Representation], and this user has made a series of purchases in the following order: (History Item List: [No.# Time: YYYY/MM/DD Title: Item Title, Item Embedding]). Based on this sequence of purchases and user representation, generate user representation token: [UserOut].
Item ($\mathcal{P}^i_{\mathcal{I}}$)	The item title is as follows: "Title": Item Title, then generate item representation token: [ItemOut].	The item title and item embedding are as follows: "Title": Item Title, Item Embedding, then generate item representation token: [ItemOut]	

finetune LLMs aiming at learning the sequential recommendation task, while A-LLMRec only trains $f_{\mathcal{I}}$ and $f_{\mathcal{U}}$ without finetuning the LLMs with LoRA. Please refer to the Appendix A.1. for more details on the projection layers as well as prompt construction.

2) Retrieval Approach (Next Item Retrieval). As shown in Table 2, we use $\mathcal{P}^u_{\mathcal{U}}$ and $\mathcal{P}^i_{\mathcal{I}}$ to denote prompts for users and items, respectively. Unlike the Next Item Title Generation approach where LLMs directly generate the title of the recommended item, the Next Item Retrieval approach generates item recommendations by computing the recommendation scores between user representations and item embeddings. More precisely, it introduces learnable tokens, i.e., [UserOut] and [ItemOut], to aggregate information from user interaction sequences and items, respectively. The last hidden states associated with the [UserOut] and [ItemOut] are used as user representations and item embeddings, denoted $\mathbf{h}^u_{\mathcal{U}} \in \mathbb{R}^{l_{llm}}$ and $\mathbf{h}^i_{\mathcal{I}} \in \mathbb{R}^{l_{llm}}$, respectively, where d_{llm} denotes the token embedding dimension of LLM. Please refer to Appendix A.2. for more details on how the user representations and item embeddings are extracted as well as the prompt construction for compared models.

Then, we compute the recommendation score between user u and item i as $s(u,i) = f_{item}(\mathbf{h}^i_{\mathcal{I}}) \cdot f_{user}(\mathbf{h}^u_{\mathcal{U}})^T$, where f_{item} and f_{user} are 2-layer MLPs, i.e., $f_{item}, f_{user} : \mathbb{R}^{d_{llm}} \rightarrow \mathbb{R}^{d'}$. Finally, the Next Item Retrieval loss is defined as follows:

$$\mathcal{L}_{\text{Retrieval}} = -\mathop{\mathbb{E}}_{u \in \mathcal{U}} [\log \frac{e^{s(u, i^{(u)}_{n_u+1})}}{\sum_{k \in C_u} e^{s(u,k)}}] \tag{1}$$

All models are trained using the $\mathcal{L}_{\text{Retrieval}}$ loss. Specifically, the set of MLPs (i.e., $f_{\mathcal{I}}, f_{\mathcal{U}}, f_{item}, f_{user}$, and two token embeddings (i.e., [ItemOut], [UserOut]) are trained, while the LLM is fine-tuned using the LoRA. In contrast, A-LLMRec does not fine-tune the LLM.

Discussion regarding prompt design. It is important to highlight that the prompts in Table 1 and Table 2 are designed to ensure that LLMs interpret the user interaction history as a sequential process. Specifically, we incorporate both the interaction number and the actual timestamp of each interaction. Additionally, when shuffling the sequence, we only rearrange the item titles and embeddings while keeping the position of interaction number and timestamp unchanged. We considered that this choice is the most effective, as it allows us to maintain the integrity of the chronological order while still testing the model's ability to generalize across different item sequences.

2.2 Evaluation Protocol

In our experiments on LLMs' sequence comprehension, we employed the leave-last-out evaluation method (i.e., next item recommendation task) [13, 30, 32]. For each user, we reserved the last item in their behavior sequence as the test data, used the second-to-last item as the validation set, and utilized the remaining items for training. The candidate item set (i.e., test set) for each user in the title generation task is generated by randomly selecting 19 non-interacted items along with 1 positive item following existing studies [16, 42]. Similarly, for the next item retrieval task, following common strategies [13, 30], we randomly select 99 non-interacted items along with 1 positive item as the candidate item set (i.e., test set) for each user.

2.3 Preliminary Analysis

In this section, we conduct experiments to investigate the ability of LLM4Rec in understanding users' item interaction sequences by comparing four different LLM4Rec models (i.e., TALLRec, LLaRA, CoLLM, and A-LLMRec)[2] with a CF-SRec model (i.e., SASRec). Note that our experiments are designed based on the assumption that randomly shuffling the order of items within a user's item interaction sequence breaks the sequential dependencies among items [18, 37]. More precisely, we conduct the following two experiments: 1) Training (Sec. 2.3.1) and Inference (Sec. 2.3.2) with shuffled sequences, and 2) Representation Similarity (Sec. 2.3.3). In the following, we describe details regarding the experimental setup and experimental results.

[2]Note that TALLRec and CoLLM are designed for binary classification (YES/NO) for a target item, while LLaRA and A-LLMRec generate the title of item to be recommended (i.e., Next Item Title Generation approach). To adapt these baselines to the Next Item Retrieval approach, we modified their setup to retrieve the target item from a provided candidate item set by using the prompts in Table 2 and training with Equation 1.

Table 3: Performance (NDCG@10) of various models when trained with original sequences and shuffled sequences (Next Item Retrieval approach). Change ratio indicates the performance change of 'Shuffle' compared with 'Original'.

		Scientific	Electronics	CDs
SASRec	Original	0.2918	0.2267	0.3451
	Shuffle	0.2688	0.2104	0.3312
	Change ratio	(-7.88%)	(-7.19%)	(-4.03%)
TALLRec	Original	0.2585	0.2249	0.3100
	Shuffle	0.2579	0.2223	0.3003
	Change ratio	(-0.23%)	(-1.16%)	(-3.13%)
LLaRA	Original	0.2844	0.2048	0.2464
	Shuffle	0.2921	0.2079	0.2695
	Change ratio	(+2.71%)	(+1.51%)	(+9.38%)
CoLLM	Original	0.3111	0.2565	0.3152
	Shuffle	0.3181	0.2636	0.3143
	Change ratio	(+2.25%)	(+2.77%)	(-0.29%)
A-LLMRec	Original	0.2875	0.2791	0.3119
	Shuffle	0.2973	0.2741	0.3078
	Change ratio	(+3.41%)	(-1.79%)	(-1.31%)
LLM-SRec	Original	0.3388	0.3044	0.3809
	Shuffle	0.3224	0.2838	0.3614
	Change ratio	(-4.84%)	(-6.77%)	(-5.11%)

Table 4: Performance (HR@1) of various models when trained with original sequences and shuffled sequences (Next Item Title Generation approach).

		Scientific	Electronics	CDs
SASRec	Original	0.3171	0.2390	0.3662
	Shuffle	0.2821	0.2158	0.3386
	Change ratio	(-11.04%)	(-9.71%)	(-7.54%)
TALLRec	Original	0.2221	0.1787	0.2589
	Shuffle	0.2181	0.1815	0.2728
	Change ratio	(-1.81%)	(+1.57%)	(+5.37%)
LLaRA	Original	0.3022	0.2616	0.3142
	Shuffle	0.2996	0.2650	0.3530
	Change ratio	(-0.86%)	(+1.30%)	(+12.35%)
CoLLM	Original	0.3010	0.2311	0.3447
	Shuffle	0.3165	0.2323	0.3763
	Change ratio	(+5.15%)	(+0.52%)	(+9.17%)
A-LLMRec	Original	0.2804	0.2672	0.3319
	Shuffle	0.2796	0.2684	0.3528
	Change ratio	(-0.29%)	(+0.45%)	(+6.30%)

2.3.1 ***Shuffled Training.*** We hypothesize that the performance of models trained on sequences in which meaningful sequential information is removed would deteriorate when evaluated on sequences in which sequential information is present (i.e., non-shuffled test sequences). To investigate this, we compare the performance of models that have been trained on the original sequence $\mathcal{S}_u$ (i.e., non-shuffled sequence) for each user u and those trained on randomly shuffled sequences [37], when they are evaluated on the same non-shuffled test sequences. Note that users' item interaction sequences are shuffled only once before training begins, rather than at every epoch, to eliminate unintended augmentation effect [31].

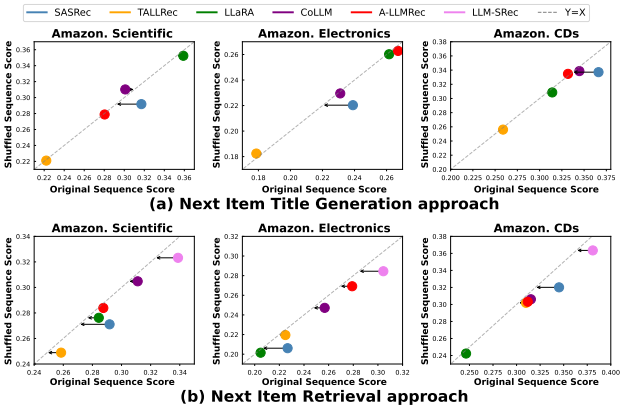

Figure 1: Performance of various models when tested with original sequences and shuffled sequences. (a) Next Item Title Generation approach (HR@1). (b) Next Item Retrieval approach (NDCG@10). "←" indicates performance drop.

Table 3 and Table 4 show the performance of various models when adapted to the Next Item Retrieval approach and the Next Item Title Generation approach, respectively. We have the following observations: 1) CF-SRec (i.e., SASRec), suffers from substantial performance degradation when the training sequences are shuffled as expected, whereas LLM4Rec models generally exhibit minimal changes in performance. This indicates that LLMs struggle to leverage sequential information, as eliminating original sequential dependencies through shuffling does not significantly impact their performance. 2) Some LLM4Rec models even show improved results despite being trained with shuffled sequences. We conjecture that in some cases random shuffling of interaction sequences during training can introduce short-term co-occurrence patterns that may coincidentally lead to improved performance. Combined with the fact that LLMs struggle to capture long-term dependencies [24], we argue that LLM4Rec models struggle to capture the sequential dependencies within the interaction sequence.

2.3.2 ***Shuffled Inference.*** We hypothesize that the performance of models that understand the sequential information inherent in a user's item interaction sequence would deteriorate when the sequence is disrupted. To investigate this, we perform inference using shuffled test sequences with the models that have been trained using the original sequences $\mathcal{S}_u$ (i.e., non-shuffled sequence). That is, unlike in Sec. 2.3.1, we shuffle the test sequences during inference rather than training sequences. It is important to note that the assumption of this experiment is that models trained with the original sequences indeed capture the sequential information.

Figure 1 (a) and (b) show the performance of various models when adapted to the Next Item Title Generation approach and the Next Item Retrieval approach, respectively. We have the following observations: 1) When the test sequences are shuffled, CF-SRec (i.e., SASRec) encounters a significant performance degradation as expected, whereas LLM4Rec remains relatively consistent (i.e., all circles except for SASRec are positioned near the $y = x$ in Figure

Table 5: Cosine similarity between user representations obtained based on original sequences and those obtained based on shuffled sequences during inference (The models are trained on the original sequences).

	Movies	Scientific	Electronics	CDs
SASRec	0.6535	0.7375	0.7083	0.7454
TALLRec	0.9731	0.9326	0.9678	0.9570
LLaRA	0.9639	0.9424	0.9800	0.9624
CoLLM	0.9067	0.9263	0.8921	0.9526
A-LLMRec	0.8872	0.8911	0.8623	0.8706
LLM-SRec	0.6128	0.7852	0.7393	0.8589

1 (a) and (b)). This implies that existing LLM4Rec models failed to capture the sequential information contained in the item interaction sequences. 2) Even though CoLLM and A-LLMRec leverage user representations derived from CF-SRec, which is solely trained based on the item interaction sequence, they fail to effectively capture the sequential information. This indicates the need for carefully distilling the user representations extracted from a pre-trained CF-SRec into LLMs.

2.3.3 ***Representation Similarity.*** In LLM4Rec models as well as in SASRec, representations of users are generated based on their item interaction sequences. Hence, we hypothesize that the user representations obtained from models that capture sequential information in a user's item interaction sequence would change when the input sequence is disrupted. To investigate this, we compute the cosine similarity between user representations obtained based on the original sequences and those obtained based on shuffled sequences during inference.

As shown in Table 5, we observe that the similarity is much higher for LLM4Rec models compared with that of CF-SRec, i.e., SASRec, indicating that LLM4Rec models are less effective than CF-SRec in capturing and reflecting changes in users' item interaction sequences. It is worth noting that among LLM4Rec models, CoLLM and A-LLMRec exhibit relatively low similarity. This is attributed to the fact that they utilize user representations extracted from a pre-trained CF-SRec in their prompts, unlike TALLRec which only uses text, or LLaRA which uses text and item embeddings. This implies that incorporating user representations enhances the ability to effectively model sequential information. However, CoLLM and A-LLMRec still exhibit higher similarity values compared to SASRec, and based on the results of previous experiments (i.e., Sec. 2.3.1 and Sec. 2.3.2, it is evident that they have yet to fully comprehend the sequential patterns.

3 METHODOLOGY: LLM-SRec

In this section, we propose LLM-SRec, a novel and simplistic LLM4Rec framework designed to enable LLMs to effectively utilize sequential information inherent in users' item interaction sequences. It is important to note that among the two prominent approaches for LLM4Rec, we employ the Next Item Retrieval approach (i.e., LLM-SRec is trained with Equation 1) due to well-known drawbacks of the Next Item Title Generation approach, i.e. restrictions on the number of candidate items [16] and the existence of position bias with candidate items [11]. In the following, we explain two additional losses aiming at: (1) distilling the sequential information extracted from a pre-trained CF-SRec to LLMs (Sec. 3.1), and (2) preventing the over-smoothing problem during distillation (Sec. 3.2). Figure 2 shows the overall framework of LLM-SRec.

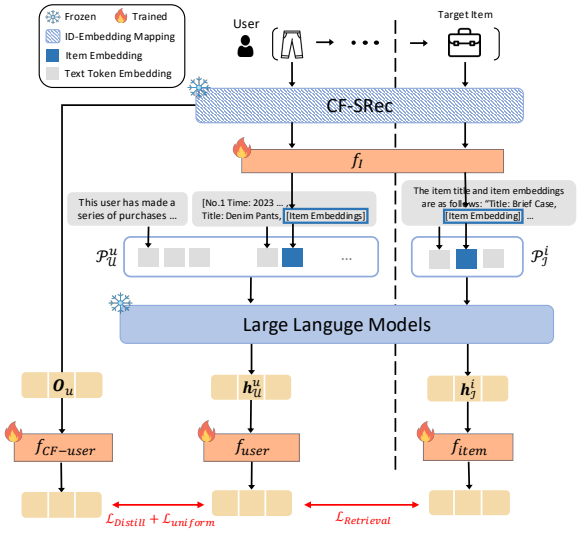

Figure 2: Overall model architecture of LLM-SRec.

3.1 Distilling Sequential Information

User representations from CF-SRec, derived solely from users' item interaction sequences, encapsulate rich sequential information crucial for sequential recommendation tasks. Despite the efforts of CoLLM and A-LLMRec trying to understand the sequential information by incorporating the user representations directly into LLM prompts, we observe that they still fail to do so (as shown in Sec. 2). Therefore, in this paper, we propose a simple alternative approach to effectively incorporating the sequential information extracted from CF-SRec into LLMs. The main idea is to distill the sequential knowledge from pre-trained and frozen CF-SRec into LLMs. More precisely, we simply match the user representation generated by a pre-trained CF-SRec, i.e., $\mathbf{O}_u = \text{CF-SRec}(\mathcal{S}_u) \in \mathbb{R}^d$, and that generated by LLMs, i.e., $\mathbf{h}^u$, as follows:

$$\mathcal{L}_{\text{Distill}} = \mathop{\mathbb{E}}_{u \in \mathcal{U}} [\text{MSE}(f_{CF-user}(\mathbf{O}_u), f_{user}(\mathbf{h}^u_{\mathcal{U}}))] \tag{2}$$

where MSE is the mean squared error loss and both $f_{CF-user}$ and f_{user} are 2-layer MLPs , i.e., $f_{CF-user} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ and $f_{user} : \mathbb{R}^{d_{llm}} \rightarrow \mathbb{R}^{d'}$, that are trainable. This simple distillation framework enables LLMs to generate user representations that effectively capture and reflect the sequential information inherent in users' item interaction sequences. The prompt used in LLM-SRec is provided in Table 2, where user representations are derived from LLMs based on interacted item titles and collaborative information. In Appendix E.1, we empirically compare our prompt with another prompt that explicitly incorporates user representations, i.e., CoLLM and A-LLMRec, demonstrating the effectiveness of LLM-SRec despite the absence of user representation in the prompt. Additionally, Appendix E.2, we conducted experiments using a contrastive learning approach as an objective for distillation instead of the MSE loss.

3.2 Preventing Over-smoothing

Simply applying an MSE loss for distillation as in Equation 2 can lead to the over-smoothing problem, i.e., two representations are highly similar, hindering LLMs from effectively learning sequential information. In extreme cases, f_{user} and $f_{CF-user}$ could be trained to produce identical outputs [16, 34]. To mitigate this over-smoothing problem, we introduce a uniformity loss [34–36] as follows:

$$\mathcal{L}_{\text{Uniform}} = \mathop{\mathbb{E}}_{u\in\mathcal{U}}[\mathop{\mathbb{E}}_{u'\in\mathcal{U}}[e^{-2\|f_{CF-user}(\mathbf{O}_u)-f_{CF-user}(\mathbf{O}_{u'})\|_2^2}]] + \mathop{\mathbb{E}}_{u\in\mathcal{U}}[\mathop{\mathbb{E}}_{u'\in\mathcal{U}}[e^{-2\|f_{user}(\mathbf{h}^u_{\mathcal{U}})-f_{user}(\mathbf{h}^{u'}_{\mathcal{U}})\|_2^2}]] \tag{3}$$

The uniformity loss $\mathcal{L}_{\text{Uniform}}$ ensures that user representations of different users are uniformly distributed across the normalized feature space on the hypersphere, preserving both separation and informativeness.

Final objective. The final objective of LLM-SRec is computed as the sum of the Next Item Retrieval loss (Equation 1), the distillation loss (Equation 2), and the uniformity loss (Equation 3) as follows:

$$\mathcal{L} = \mathcal{L}_{\text{Retrieval}} + \mathcal{L}_{\text{Distill}} + \mathcal{L}_{\text{Uniform}} \tag{4}$$

It is important to note that LLM-SRec does not require any additional training for either the pre-trained CF-SRec or LLMs during its training process. Instead, LLM-SRec only optimizes a small set of MLP layers (i.e., $f_{\mathcal{I}}$, f_{user}, $f_{CF-user}$, and f_{item}) and two LLM tokens (i.e., [UserOut] and [ItemOut]). Therefore, LLM-SRec achieves faster training and inference time compared to existing LLM4Rec baselines, including TALLRec, LLaRA, CoLLM, and A-LLMRec, results are described in Sec. 4.3.1. Furthermore, for training efficiency, we only consider the last item in $\mathcal{S}_u$ for each user u to minimize Equation 4 [16]. That is, for each user u, we predict the last item $i^{(u)}_{n_u}$ given the sequence $(i^{(u)}_1, \ldots, i^{(u)}_t, \ldots, i^{(u)}_{n_u-1})$. In Appendix E.3, we also show the results when considering all items, i.e., autoregressive learning.

Discussion on the Efficiency of LLM-SRec. Last but not least, unlike existing LLM4Rec models such as TALLRec, LLaRA, and CoLLM, all of which require fine-tuning of LLMs, LLM-SRec eliminates the need for additional training or fine-tuning on either the pre-trained CF-SRec or the LLMs. Despite its simplicity, LLM-SRec is highly effective in equipping LLMs with the ability to comprehend sequential information, making it lightweight but effective for sequential recommendation tasks.

4 Experiments

Datasets. We conduct experiments on four Amazon datasets [10], i.e., Movies, Scientific, Electronics, and CDs. Following prior studies [13, 30], we use five-core datasets consisting of users and items with a minimum of five interactions each. The statistics for each dataset after preprocessing are summarized in Table 10 in Appendix B.

Baselines. We compare three groups of models as our baselines: models that use only interaction sequences (CF-SRec: GRU4Rec [9], BERT4Rec [30], NextItNet [40], SASRec [13]), Language Model based models (LM-based: CTRL [21], RECFORMER [20]), and Large Language Model based models (LLM4Rec: TALLRec [2], LLaRA [23], CoLLM [42], A-LLMRec [16]). For fair comparisons, we implemented all LLM4Rec baselines with Next Item Retrieval approach. Details regarding the baselines are provided in Appendix C.

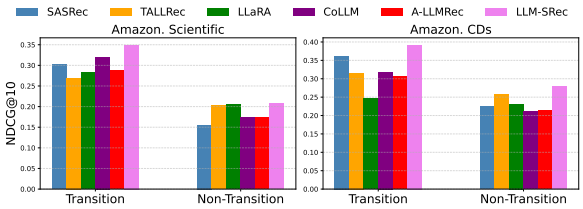

Figure 3: Performance with a varying degree of sequential information ("Transition Set" vs. "Non-Transition Set").

Evaluation Protocol. We employ the leave-last-out strategy [13] for evaluation, where the most recent item in the user interaction sequence is used as the test item, the second most recent item as the validation item, and the remaining sequence for training. To evaluate the performance of sequential recommendation, we add 99 randomly selected non-interacted items to the test set, ensuring that each user's test set consists of one positive item and 99 negative items. Evaluation is conducted using two widely adopted metrics: Normalized Discounted Cumulative Gain (NDCG@N) and Hit Ratio (HR@N), with N set to 10 and 20.

Implementation Details. For fair comparisons, we adopt pre-trained LLaMA 3.2 (3B-Instruct) as the backbone LLM for all LLM4Rec baselines (i.e., TALLRec, CoLLM, LLaRA, and A-LLMRec) including LLM-SRec. Similarly, SASRec serves as the pre-trained CF-SRec for CoLLM, LLaRA, A-LLMRec, and LLM-SRec. Please refer to the Appendix D for more details regarding the hyper-parameters and train settings.

4.1 Recommendation Performance Comparison

4.1.1 ***Overall performance.*** Table 6 presents the recommendation performance of LLM-SRec and baselines on four datasets. From the results, we have the following observations: 1) LLM-SRec consistently outperforms existing LLM4Rec models. This result highlights the importance of distilling the sequential knowledge extracted from CF-SRec into LLMs. 2) LLM-SRec outperforms CF-SRec based and LM-based models, suggesting that the reasoning ability and the pre-trained knowledge of LLMs significantly contribute to recommendation performance. 3) LLM4Rec models that utilize CF-SRec in their framework (i.e., LLaRA, CoLLM, and A-LLMRec) outperform TALLRec while being comparable to their CF-SRec backbone, i.e., SASRec. This indicates that while incorporating the CF knowledge extracted from a pre-trained CF-SRec is somewhat helpful, the lack of sequence understanding ability limits further improvements, even when using the LLMs. In summary, these findings emphasize the significance of seamless distillation of the sequential information extracted from a pre-trained CF-SRec into LLMs as in LLM-SRec.

4.1.2 ***Transition & Non-Transition Sequences.*** To examine how the degree of sequential information contained in users' item interaction sequences influences the model performance, we categorized users based on the degree of sequential transitions in their interaction history[3]. We make the following observations from

[3]We count the number of unique transitions occurring between consecutive items, i.e., $i^{(u)}_t \rightarrow i^{(u)}_{t+1}$, within the sequence of user u, i.e., $\mathcal{S}^u$. Then, we sort users according to the transition score, i.e., t-score$^u = (\sum_{t=1}^{n_u-1} Count(i^{(u)}_t \rightarrow i^{(u)}_{t+1}))/(n^u - 1)$. Users within the top-50% are assigned to the "Transition Set", while the remaining users were

Table 6: Overall model performance. The best performance is denoted in bold.

Dataset	Metric	CF-SRec				LM-based		LLM4Rec				
		GRU4Rec	BERT4Rec	NextItNet	SASRec	CTRL	RECFORMER	TALLRec	LLaRA	CoLLM	A-LLMRec	LLM-SRec
Movies	NDCG@10	0.3152	0.2959	0.2538	0.3459	0.2785	0.2068	0.1668	0.1522	0.3223	0.3263	**0.3560**
	NDCG@20	0.3494	0.3303	0.2879	0.3745	0.3099	0.2337	0.2126	0.1944	0.3577	0.3629	**0.3924**
	HR@10	0.4883	0.4785	0.4221	0.5180	0.4264	0.3569	0.3234	0.2914	0.5089	0.5127	**0.5569**
	HR@20	0.6245	0.6213	0.5522	0.6310	0.5429	0.5264	0.5060	0.4599	0.6491	0.6577	**0.7010**
Scientific	NDCG@10	0.2642	0.2576	0.2263	0.2918	0.2152	0.2907	0.2585	0.2844	0.3111	0.2875	**0.3388**
	NDCG@20	0.2974	0.2913	0.2657	0.3245	0.2520	0.3113	0.3048	0.3265	0.3489	0.3246	**0.3758**
	HR@10	0.4313	0.4437	0.3908	0.4691	0.3520	0.4506	0.4574	0.4993	0.5182	0.4957	**0.5532**
	HR@20	0.5524	0.5822	0.5356	0.5987	0.4882	0.5710	0.6276	0.6658	0.6676	0.6427	**0.6992**
Electronics	NDCG@10	0.2364	0.1867	0.1712	0.2267	0.1680	0.2032	0.2249	0.2048	0.2565	0.2791	**0.3044**
	NDCG@20	0.2743	0.2172	0.2069	0.2606	0.2003	0.2441	0.2670	0.2454	0.2948	0.3173	**0.3424**
	HR@10	0.3843	0.3325	0.3017	0.3749	0.2861	0.3586	0.3802	0.3441	0.4236	0.4622	**0.4885**
	HR@20	0.5196	0.4740	0.4324	0.5096	0.4152	0.5213	0.5476	0.5032	0.5741	0.6137	**0.6385**
CDs	NDCG@10	0.2155	0.3019	0.2207	0.3451	0.2968	0.3238	0.3100	0.2464	0.3152	0.3119	**0.3809**
	NDCG@20	0.2530	0.3386	0.2562	0.3795	0.3316	0.3642	0.3493	0.2951	0.3557	0.3526	**0.4158**
	HR@10	0.3712	0.5018	0.3842	0.5278	0.4574	0.5140	0.5052	0.4665	0.5290	0.5300	**0.6085**
	HR@20	0.5092	0.6605	0.5422	0.6635	0.5957	0.6739	0.6633	0.6590	0.6895	0.6914	**0.7461**

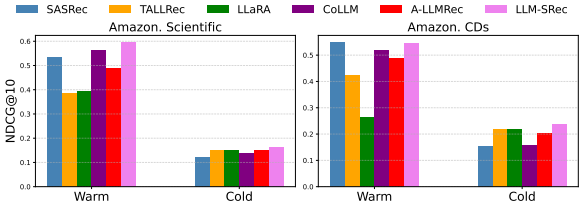

Figure 4: Performance on Warm/Cold item Scenarios.

Figure 3. 1) LLM-SRec outperforms all baselines, especially in the Transition Set, where sequential information is more abundant. This demonstrates that the distillation of sequence information enables the LLMs to comprehend and utilize sequential information inherent in users' interaction sequences. 2) The performance gap between LLM4Rec baselines and LLM-SRec is smaller in the Non-Transition Set compared with the Transition Set. This indicates that existing LLM4Rec models lack the capability to effectively capture sequential dependencies among items and further emphasizes the importance of effective sequential modeling.

Table 7: Performance on cross-domain scenarios (HR@10).

	SASRec	TALLRec	LLaRA	CoLLM	A-LLMRec	LLM-SRec
Electronics → Scientific	0.1002	0.1214	0.1225	0.1232	0.1262	**0.1310**
Electronics → CDs	0.0974	0.1132	0.1174	0.1152	0.1217	**0.1369**

4.1.3 ***Performance under Warm/Cold Scenarios.*** In this section, we conduct experiments to examine how LLM-SRec performs in both warm and cold item settings. Following the experimental setup of A-LLMRec [16], items are labeled as 'warm' if they belong to the top 35% in terms of the number of interactions with users, while those in the bottom 35% are labeled as 'cold' items. We have the following observations in Figure 4: 1) LLM-SRec consistently achieves superior performance in both warm and cold scenarios, benefiting from its ability to capture the sequential information within item interaction sequences. Additionally, the performance in the cold setting shows that LLM-SRec effectively leverages the generalizability of LLMs, utilizing pre-trained knowledge and textual understanding even though there is insufficient collaborative knowledge for cold items. 2) TALLRec, which relies solely on textual information, performs inferior in warm settings than LLM4Rec baselines (i.e., LLaRA, CoLLM, and A-LLMRec) that incorporate collaborative knowledge from CF-SRec. However, these models still underperform compared to LLM-SRec, highlighting the necessity of modeling both collaborative and sequential information for effective LLM-based recommendation. 3) As expected, SASRec particularly struggles for cold items due to its exclusive reliance on the user-item interaction data. In contrast, LLM4Rec models, especially LLM-SRec, leverage item textual information to mitigate the scarcity of interactions.

4.1.4 ***Performance under Cross-domain Scenarios.*** To further verify the generalizability of LLM-SRec, we evaluate LLM-SRec on the cross-domain scenarios, following the setting of A-LLMRec [16], where the models are evaluated on datasets that have not been used for training. Specifically, we pre-train the models on the Electronic dataset, as it contains the most users and items, and perform evaluations on the Scientific and CDs. As shown in Table 7, we observe that 1) LLM-SRec consistently outperforms all the baselines in the cross-domain scenarios. Leveraging the textual understanding of LLMs, LLM-SRec extracts textual information from unseen items which lack collaborative information. Additionally, by capturing sequential information from the source data, i.e., Electronics dataset, and aligning it with the textual understanding of LLMs, LLM-SRec generates high-quality user representations, enabling superior performance in cross-domain scenarios. 2) While LLM4Rec baselines also address the issue of unseen items through LLM's textual understanding, they fail to generate user representations that capture sequential information, resulting in lower performance than LLM-SRec. In contrast, CF-SRec struggles with unseen items due to the difficulty of generating collaborative knowledge of items, leading to inferior performance.

assigned to the "Non-Transition Set." That is, users whose item interaction sequences exhibit sequential information are assigned to the "Transition Set".

Table 8: Ablation studies on the components of LLM-SRec.

Row	Ablation	Train Set	Movies		Scientific		Electronics		CDs	
			NDCG@10	NDCG@20	NDCG@10	NDCG@20	NDCG@10	NDCG@20	NDCG@10	NDCG@20
(a)	w.o. $\mathcal{L}_{\text{Distill}}$, $\mathcal{L}_{\text{Uniform}}$	Original	0.3204	0.3569	0.3088	0.3450	0.2659	0.3066	0.2278	0.2701
		Shuffle	0.3176	0.3557	0.3013	0.3379	0.2589	0.2990	0.2224	0.2649
		Change ratio	(-0.87%)	(-0.34%)	(-2.33%)	(-2.06%)	(-2.63%)	(-2.48%)	(-2.37%)	(-1.92%)
(b)	w.o. $\mathcal{L}_{\text{Uniform}}$	Original	0.3339	0.3700	0.3283	0.3653	0.2895	0.3285	0.3622	0.4013
		Shuffle	0.3089	0.3456	0.3164	0.3536	0.2732	0.3110	0.3478	0.3885
		Change ratio	(-7.49%)	(-6.59%)	(-3.62%)	(-3.20%)	(-5.63%)	(-5.33%)	(-3.98%)	(-3.19%)
(c)	LLM-SRec	Original	**0.3560**	**0.3924**	**0.3388**	**0.3758**	**0.3044**	**0.3424**	**0.3809**	**0.4158**
		Shuffle	0.3263	0.3624	0.3224	0.3591	0.2838	0.3210	0.3614	0.3981
		Change ratio	(-8.34%)	(-7.65%)	(-4.84%)	(-4.44%)	(-6.77%)	(-6.25%)	(-5.11%)	(-4.26%)

Table 9: Train/Inference time comparison.

	Scientific		Electronics	
	Train (min/epoch)	Inference (min)	Train (min/epoch)	Inference (min)
TALLRec	194.43	37.04	236.73	29.04
LLaRA	202.20	38.79	241.17	30.62
CoLLM	214.12	39.86	251.51	32.58
A-LLMRec	190.94	35.01	235.02	28.14
LLM-SRec	**185.91**	**34.17**	**218.21**	**27.57**

4.2 Ablation Studies

In this section, we evaluate the contribution of each component LLM-SRec. To analyze not only the contribution of each component in terms of the final performance but also its impact on the sequence understanding ability, we conduct experiments under the setting described in Sec. 2.3.1. In other words, we compare the performance of training with the original sequences versus training with shuffled sequences. Table 8 presents the following observations: 1) When both $\mathcal{L}_{\text{Distill}}$ and $\mathcal{L}_{\text{Uniform}}$ are present (i.e., vanilla LLM-SRec in row (c)), the model consistently achieves the highest performance on the original sequence due to the benefits of sequence understanding ability. Furthermore, compared with its variants (i.e., row (c) vs (a,b)), the performance of vanilla LLM-SRec drops the most rapidly when shuffling is applied, ensuring that vanilla LLM-SRec indeed comprehends and effectively utilizes sequential information. 2) In the absence of both $\mathcal{L}_{\text{Distill}}$ and $\mathcal{L}_{\text{Uniform}}$ (i.e., row (a)), where the sequential knowledge extracted from CF-SRec is not distilled to the LLMs, the model exhibits the lowest performance on the original sequence. Additionally, even when random shuffling is applied, the performance drop is minor. This result suggests that the model lacks sequence understanding capability without $\mathcal{L}_{\text{Distill}}$ and $\mathcal{L}_{\text{Uniform}}$. 3) When only $\mathcal{L}_{\text{Uniform}}$ is removed, the model suffers from the over-smoothing problem as discussed in Sec. 3.2, leading to lower performance compared to LLM-SRec. However, since $\mathcal{L}_{\text{Distill}}$ is still present, we observe a significant performance drop when applying random shuffling. This once more indicates that the model is endowed with the sequence understanding ability with $\mathcal{L}_{\text{Distill}}$.

Remark. Recall that simply injecting item embeddings or user representations from CF-SRec into LLMs, as done in existing LLM4Rec models, is insufficient for effective sequence understanding as shown in Sec. 2.3. In contrast, our ablation studies validate the effectiveness of our simple approach of incorporating the sequential knowledge into LLMs.

4.3 Model analysis

4.3.1 ***Train/Inference Efficiency.*** Note that LLM-SRec is efficient as it does not require fine-tuning either CF-SRec or the LLM itself. To quantitatively analyze the model efficiency, we compare the training and inference time of LLM-SRec with LLM4Rec baselines (i.e., TALLRec, LLaRA, CoLLM, and A-LLMRec). Specifically, we measure the training time per epoch and the total inference time on the Scientific and Electronics datasets. As shown in Table 9, LLM-SRec achieves significantly faster training and inference times than all baselines. This is mainly because baselines using Parameter-Efficient Fine-Tuning methods such as LoRA require fine-tuning the LLMs, increasing both training and inference time. Furthermore, compared to A-LLMRec, which does not fine-tune the LLM, LLM-SRec remains more efficient. This is mainly because A-LLMRec involves two-stage learning, leading to increase training time. Additionally, since A-LLMRec incorporates user representations into its prompts, it requires longer prompts during inference, resulting in higher computational overhead and slower inference speed compared to LLM-SRec. These results highlight the efficiency of LLM-SRec, supporting the model as a more computationally feasible solution for real-world applications for recommendation tasks while maintaining superior recommendation performance.

4.3.2 ***Size of LLMs.*** Note that all baseline LLM4Rec models including LLM-SRec use LLaMA 3.2 (3B-Instruct) as the backbone LLMs. In this section, to investigate the impact of LLM size on recommendation performance, we replace the backbone with larger LLMs, i.e., LLaMA 3.1 (8B) [1]. We have the following observations in Figure 5: 1) Replacing the backbone LLMs to larger LLMs greatly enhances the overall performance of LLM4Rec models. This aligns well with the scaling law of LLMs observed in other domains [14]. Moreover, the superior performance of LLM-SRec is still valid when the backbone is replaced to larger LLMs. 2) Surprisingly, LLM-SRec with the smaller backbone LLMs outperforms the baseline LLM4Rec models with the larger backbone LLMs. This indicates that distilling sequential information is more crucial than merely increasing the LLM size to enhance the overall performance of sequential recommendation, which again highlights the importance of equipping LLMs with sequential knowledge to improve the sequential recommendation capabilities of LLMs.

4.3.3 ***Case Study.*** In this section, we conduct a case study on the Electronics dataset to qualitatively validate the benefit of sequential knowledge and LLMs' textual understanding. Figure 6 shows

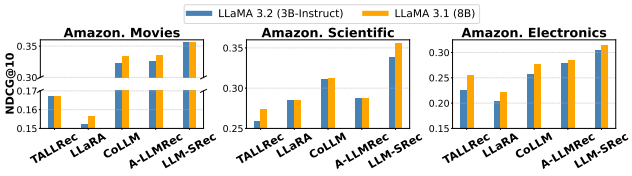

Figure 5: Performance of different sizes of backbone LLMs.

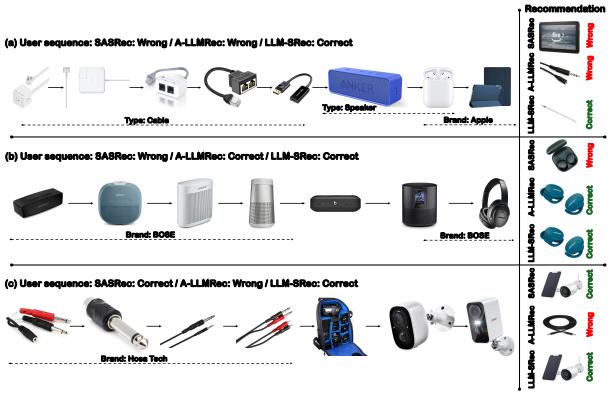

Figure 6: Case study on Electronics dataset.

three cases highlighting the effect of sequential knowledge and textual knowledge on recommendation. The cases are categorized as follows: (a) only LLM-SRec provides correct recommendations, (b) LLM4Rec models (i.e., A-LLMRec and LLM-SRec) provide correct recommendations while SASRec fails, and (c) LLM-SRec and SASRec provide correct recommendations while an LLM4Rec (i.e., A-LLMRec) baseline fails. We have the following observations: 1) In case (a), the user's preference shifts from cable-related items to products from "Apple" brand. LLM-SRec correctly recommends "Apple Pencil," while SASRec captures the changing preference but fails to recognize the textual information "Apple," leading to a wrong recommendation of an "Amazon Fire 7 Tablet." On the other hand, A-LLMRec, which struggles to capture sequential information, recommends "Audio Cable" based on textual knowledge of "Cable" and "Speaker." This emphasizes the importance of leveraging both sequential and textual information. 2) In case (b), the user focused on "BOSE" brand products. Both LLM-SRec and A-LLMRec, leveraging textual knowledge, successfully recommended the "BOSE" earbuds, while SASRec, lacking textual information, recommended the "SAMSUNG Galaxy Buds." This highlights the importance of textual knowledge in generating accurate recommendations. 3) In case (c), the user's preference shifts from "Hosa Tech" cables to security camera. LLM-SRec and SASRec capture this preference shift and provide relevant recommendations, while A-LLMRec recommends another "Hosa Tech" cable ignoring the sequential patterns. Those cases demonstrate that both textual and sequential information are crucial for accurate recommendations, showcasing the superiority of LLM-SRec in integrating both for improved performance.

5 Related Work

Sequential Recommender Systems. Recommendation systems primarily focus on capturing collaborative filtering (CF) to identify similar items/users. Matrix Factorization-based approaches, achieved notable success by constructing CF knowledge in a latent space [3, 8, 15, 26]. However, in conjunction with CF knowledge, understanding dynamic evolution in temporal user preferences has become a powerful tool, leading to the development of collaborative filtering-based sequential recommenders (CF-SRec) [4, 9, 13, 17, 27, 30, 39]. Initial approaches combined Matrix Factorization with Markov Chains to model temporal dynamics [28]. Subsequently, neural network-based methods advanced sequential recommender systems, with GRU4Rec [9] leveraging recurrent architectures, while methods such as Caser [33] and NextItNet [40] adopted Convolutional Neural Networks [19]. More recently, models such as SASRec [40] and BERT4Rec [30], based on attention mechanisms, have demonstrated superior performance by focusing on the more relevant interaction sequences. These advancements underscore the importance of effectively modeling user behavior dynamics for improved recommendation accuracy.

LLM-based Recommender Systems. LLMs have recently gained attention in recommendation systems [5, 7, 38, 41], leveraging their reasoning ability and textual understanding for novel approaches such as zero-shot recommendation [11] and conversational recommendation [29]. However, TALLRec [2] highlights the gap between LLMs' language modeling tasks and recommendation tasks, proposing a fine-tuning approach through Parameter-Efficient Fine-Tuning (PEFT) to adapt LLMs to recommendation tasks. More recently, LLaRA [23], CoLLM [42], and A-LLMRec [16] have been proposed. LLaRA and CoLLM combine CF-SRec item embeddings with text embeddings from item titles, enabling LLMs to utilize CF knowledge. A-LLMRec further incorporates item descriptions into a latent space, enabling the model to demonstrate robust performance in various scenarios. Despite these advancements, prior methods fail to capture dynamic user preferences inherent in user interaction sequences as shown in Sec. 2.3.

6 Conclusion

In this paper, we address a fundamental limitation of LLM4Rec models, i.e., their inability to capture sequential patterns, and empirically demonstrate this shortcoming through extensive experiments. To address the limitation, we propose a simple yet effective distillation framework, named LLM-SRec, which effectively transfers sequential knowledge extracted from CF-SRec into LLMs. By doing so, LLM-SRec enables LLMs to effectively capture sequential dependencies, leading to superior recommendation performance compared to existing CF-SRec, LM-based recommender systems, and LLM4Rec. Furthermore, LLM-SRec achieves high efficiency, as it does not require fine-tuning either CF-SRec or the LLM, demonstrating the effectiveness of our simple yet efficient architecture.

Acknowledgments

This work was supported by NAVER Corporation, the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2024-00335098), and the National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT (RS-2022-NR068758).

References

[1] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

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

[3] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. 2015. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 43–50.

[4] Seungyoon Choi, Sein Kim, Hongseok Kang, Wonjoong Kim, and Chanyoung Park. 2025. Dynamic Time-aware Continual User Representation Learning. *arXiv preprint arXiv:2504.16501* (2025).

[5] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1126–1132.

[6] 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*. 299–315.

[7] 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*. 1096–1102.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

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

[10] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952* (2024).

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

[12] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

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

[14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[15] Jiwan Kim, Hongseok Kang, Sein Kim, Kibum Kim, and Chanyoung Park. 2025. Disentangling and Generating Modalities for Recommendation in Missing Modality Scenarios. *arXiv preprint arXiv:2504.16352* (2025).

[16] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 1395–1406. doi:10.1145/3637528.3671931

[17] Sein Kim, Namkyeong Lee, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2023. Task Relation-aware Continual User Representation Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1107–1119.

[18] Anton Klenitskiy, Anna Volodkevich, Anton Pembek, and Alexey Vasilev. 2024. Does It Look Sequential? An Analysis of Datasets for Evaluation of Sequential Recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems* (Bari, Italy) *(RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 1067–1072. doi:10.1145/3640457.3688195

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[20] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1258–1267.

[21] Xiangyang Li, Bo Chen, Lu Hou, and Ruiming Tang. 2023. Ctrl: Connect tabular and language model for ctr prediction. *CoRR* (2023).

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

[23] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. LLaRA: Large Language-Recommendation Assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 1785–1795. doi:10.1145/3626772.3657690

[24] Hao Liu and Pieter Abbeel. 2024. Blockwise parallel transformers for large context models. *Advances in Neural Information Processing Systems* 36 (2024).

[25] Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* 364 (2019).

[26] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[27] Yunhak Oh, Sukwon Yun, Dongmin Hyun, Sein Kim, and Chanyoung Park. 2023. Muse: music recommender system with shuffle play recommendation enhancement. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 1928–1938.

[28] 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*. 811–820.

[29] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*. 890–896.

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

[31] Kyosuke Takahagi and Hiroyuki Shinnou. 2023. Data Augmentation by Shuffling Phrases in Recognizing Textual Entailment. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*. Association for Computational Linguistics, Hong Kong, China, 194–200. https://aclanthology.org/2023.paclic-1.19/

[32] 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* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 565–573. https://doi.org/10.1145/3159652.3159656

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

[34] Chen Wang, Liangwei Yang, Zhiwei Liu, Xiaolong Liu, Mingdai Yang, Yueqing Liang, and Philip S. Yu. 2024. Collaborative Alignment for Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (Boise, ID, USA) *(CIKM '24)*. Association for Computing Machinery, New York, NY, USA, 2315–2325. doi:10.1145/3627673.3679535

[35] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards Representation Alignment and Uniformity in Collaborative Filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1816–1825. doi:10.1145/3534678.3539253

[36] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*. PMLR, 9929–9939.

[37] Daniel Woolridge, Sean Wilner, and Madeleine Glick. 2021. Sequence or Pseudo-Sequence? An Analysis of Sequential Recommendation Datasets.. In *Perspectives@ RecSys*.

[38] Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. 2024. Coral: Collaborative retrieval-augmented large language models improve long-tail recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3391–3401.

[39] 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*, Vol. 33. 346–353.

[40] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 582–590.

[41] 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* (2023).

[42] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2023. Collm: Integrating collaborative embeddings into large language models for recommendation. *arXiv preprint arXiv:2310.19488* (2023).

A Details of LLM4Rec Prompt Construction

This section provides additional details on how to construct prompts for LLM4Rec models discussed in Sec. 2.1.

A.1 Next Item Title Generation

Based on the user interaction sequence $\mathcal{S}_u$ and candidate set C_u of user u, the textual data for the interacted items and candidate items are defined as $\mathcal{T}_{\mathcal{S}_u} = \{\text{Text}(i) \mid i \in \mathcal{S}_u\}$ and $\mathcal{T}_{C_u} = \{\text{Text}(i) \mid i \in C_u\}$, respectively, where $\text{Text}(i)$ represents textual information (e.g., title or description) of item i.

For models such as LLaRA [23], CoLLM [42], and A-LLMRec [16], which incorporate item embeddings and user representations from a pre-trained CF-SRec, we use $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}| \times d}$ to denote the item embedding matrix of the pre-trained CF-SRec, where d is the hidden dimension of the embeddings. We also define $f_{\mathcal{I}}$ and $f_{\mathcal{U}}$ as the item and user projection layers used in LLaRA, CoLLM, and A-LLMRec (includes Stage-1 item encoder of A-LLMRec), respectively. The embeddings of items in the item interaction sequence $\mathcal{S}_u$ are defined as $\mathbf{E}_{\mathcal{S}_u} = \{f_{\mathcal{I}}(\mathbf{E}_i) \mid i \in \mathcal{S}_u\}$, while the embeddings for the candidate items C_u are represented as $\mathbf{E}_{C_u} = \{f_{\mathcal{I}}(\mathbf{E}_i) \mid i \in C_u\}$, where $f_{\mathcal{I}}(\mathbf{E_i}) \in \mathbb{R}^{d_{llm}}$ and d_{llm} denotes the token embedding dimension of LLM. Furthermore, the user representation is defined as $\mathbf{Z}_u = f_{\mathcal{U}}(\text{CF-SRec}(\mathcal{S}_u)) \in \mathbb{R}^{d_{llm}}$, where $\text{CF-SRec}(\mathcal{S}_u)$ represents the user u's representation obtained from the item interaction sequence $\mathcal{S}_u$ using a pre-trained CF-SRec. Then, using the prompts shown in Table 1, LLMs are trained for the sequential recommendation task through the Next Item Title Generation approach as follows:

$$p(\text{Text}(i^{(u)}_{n_u+1}) \mid \mathcal{P}^u, \mathcal{D}^u) \tag{5}$$

where $\mathcal{P}^u$ is the input prompt for user u, and D^u represents the set of interacted and candidate item titles and their corresponding embeddings used in Table 1 for user u as follows:

$$\mathcal{D}^u = \begin{cases} \mathcal{T}_{\mathcal{S}_u}, \mathcal{T}_{C_u} & \text{TALLRec} \\ \mathcal{T}_{\mathcal{S}_u}, \mathcal{T}_{C_u}, \mathbf{E}_{\mathcal{S}_u}, \mathbf{E}_{C_u} & \text{LLaRA} \\ \mathcal{T}_{\mathcal{S}_u}, \mathcal{T}_{C_u}, \mathbf{E}_{\mathcal{S}_u}, \mathbf{E}_{C_u}, \mathbf{Z}_u & \text{CoLLM/A-LLMRec} \end{cases} \tag{6}$$

A.2 Next Item Retrieval

Based on the prompt $\mathcal{P}^u_{\mathcal{U}}$ and $\mathcal{P}^i_{\mathcal{I}}$ in Table 2, we extract representation of user $u \in \mathcal{U}$, denoted $\mathbf{h}^u_{\mathcal{U}}$ and the embedding of item $i \in C_u$, denoted $\mathbf{h}^i_{\mathcal{I}}$ as follows:

$$\mathbf{h}^u_{\mathcal{U}} = \text{LLM}(\mathcal{P}^u_{\mathcal{U}}, \mathcal{D}'^u), \quad \mathbf{h}^i_{\mathcal{I}} = \text{LLM}(\mathcal{P}^i_{\mathcal{I}}, \mathcal{D}'^i) \tag{7}$$

where $\mathcal{P}^u_{\mathcal{U}}$ denotes the input prompt for user u to extract representation of user u, $\mathcal{P}^i_{\mathcal{I}}$ denotes the input prompt for item i to extract embedding of item i, D'^u denotes the set of interacted item titles and their corresponding embeddings for user u, while D'^i denotes the item title and its embedding for candidate item i, as presented in Table 2, as follows:

$$\mathcal{D}'^u = \begin{cases} \mathcal{T}_{\mathcal{S}_u} & \text{TALLRec} \\ \mathcal{T}_{\mathcal{S}_u}, \mathbf{E}_{\mathcal{S}_u} & \text{LLaRA} \\ \mathcal{T}_{\mathcal{S}_u}, \mathbf{E}_{\mathcal{S}_u}, \mathbf{Z}_u & \text{CoLLM/A-LLMRec/LLM-SRec} \end{cases}$$

$$\mathcal{D}'^i = \begin{cases} \text{Text}(i) & \text{TALLRec} \\ \text{Text}(i), f_{\mathcal{I}}(\mathbf{E}_i) & \text{LLaRA/CoLLM/A-LLMRec/LLM-SRec} \end{cases} \tag{8}$$

Then, using the user representation $\mathbf{h}^u_{\mathcal{U}}$ and item embedding $\mathbf{h}^i_{\mathcal{I}}$, LLMs are trained for the sequential recommendation task through the Next Item Retrieval approach as follows:

$$p(i^{(u)}_{n_u+1} \mid \mathcal{S}_u) \propto s(u, i^{(u)}_{n_u+1}) = f_{item}(\mathbf{h}^{i^{(u)}_{n_u+1}}_{\mathcal{I}}) \cdot f_{user}(\mathbf{h}^u_{\mathcal{U}})^T \tag{9}$$

where f_{item} and f_{user} denote projection layers defined in Equation 2.

B Datasets

Table 10 shows the statistics of the dataset after preprocessing.

Table 10: Statistics of datasets after preprocessing.

Dataset	Movies	Scientific	Electronics	CDs
# Users	11,947	23,627	27,601	18,481
# Items	17,490	25,764	31,533	30,951
# Interactions	144,071	266,164	292,308	284,695

C Baselines

(1) Collaborative filtering based (CF-SRec)
- **GRU4Rec** [9] employs a recurrent neural network (RNN) to capture user behavior sequences for session-based recommendation.
- **BERT4Rec** [30] utilizes bidirectional self-attention mechanisms and a masked item prediction objective to model complex user preferences from interaction sequences.
- **NextItNet** [40] applies temporal convolutional layers to capture both short-term and long-term user preferences.
- **SASRec** [13] is a self-attention based recommender system designed to capture long-term user preference.

(2) Language model based (LM-based)
- **CTRL** [21] initializes the item embeddings of the backbone recommendation models with textual semantic embeddings using the RoBERTa [25] encoding models. And fine-tunes the backbone models for the recommendation task.
- **RECFORMER** [20] leverages a Transformer-based framework for sequential recommendation, representing items as sentences by flattening the item title and attributes.

(3) Large Language Model based (LLM4Rec)
- **TALLRec** [2] fine-tunes LLMs for the recommendation task by formulating the recommendation task as a target item title generation task.
- **LLaRA** [23] uses CF-SRec to incorporate behavioral patterns into LLM. To align the behavioral representations from the CF-SRec this model employs a hybrid prompting which is a concatenated form of textual embedding and item representations.

Table 11: Performance comparison of prompts with/without explicit user representations.

Dataset	Metric	With User Representations	LLM-SRec
Movies	NDCG@10	**0.3625**	0.3560
	HR@10	**0.5626**	0.5569
Scientific	NDCG@10	0.3342	**0.3388**
	HR@10	0.5516	**0.5532**
Electronics	NDCG@10	0.2924	**0.3044**
	HR@10	0.4725	**0.4885**

- **CoLLM** [16] integrates the collaborative information as a distinct modality into LLMs by extracting and injecting item embeddings from CF-SRec.
- **A-LLMRec** [16] enables LLMs to leverage the CF knowledge from CF-SRec and item semantic information through a two-stage learning framework.

D Implementation Details

In our experiments, we adopt SASRec as a CF-SRec backbone for CoLLM, LLaRA, A-LLMRec, and LLM-SRec, with its item embedding dimension fixed to 64 and batch size set to 128. For LLM4Rec baselines, including Stage-2 of A-LLMRec, the batch size is 20 for the Movies, Scientific, and CDs datasets, while 16 is used for the Electronics dataset. For Stage-1 of A-LLMRec, the batch size is set to 64. When using Intel Gaudi v2, we set the batch size to 4 due to 8-bit quantization constraints. All LLM4Rec models are trained for a maximum of 10 epochs, with validation scores evaluated at every 10% of the training progress within each epoch, where early stop with patience of 10 is applied to prevent over-fitting. All models are optimized using Adam with a learning rate 0.0001, and the dimension size of the projected embedding d' is 128. Experiments are conducted using a single NVIDIA GeForce A6000 (48GB) GPU and a single Gaudi v2 (100GB).

E Additional Experiments

E.1 Effectiveness of Input Prompts

Note that rather than explicitly having the user representations in the input prompt, we rely on the [UserOut] token to extract the user representations as shown in (Table 2 (b)). In Table 11, to validate whether it is sufficient, we compare the performance of LLM-SRec with and without the explicit user representations. The results show a comparable performance between the two prompts. This suggests that through Equation 2, the sequential information contained in the user representation is effectively transferred to the LLMs, enabling them to understand sequential dependencies using only the user interaction sequence without explicitly incorporating the user representation in the prompt. Furthermore, omitting the user representation and its associated text from the prompt reduces input prompt length, improving training/inference efficiency, which implies the practicality of LLM-SRec's prompt.

E.2 Distillation with Contrastive Learning

Recall that we distill sequential information from CF-SRec to LLMs using MSE loss in Equation 2. To further investigate the impact of the distillation loss function, we adapt a naive contrastive learning method for sequential information distillation, i.e., Equation 10.

Table 12: Distillation with contrastive learning (NDCG@10).

Distillation Loss	Inference	Movies	Scientific	Electronics
Contrastive	Original	0.3410	0.2767	0.2553
	Shuffle	0.3151	0.2638	0.2398
	Change ratio	(-7.60%)	(-4.66%)	(-6.07%)
LLM-SRec (MSE)	Original	**0.3560**	**0.3388**	**0.3044**
	Shuffle	0.3272	0.3232	0.2845
	Change ratio	(-8.10%)	(-4.60%)	(-6.53%)

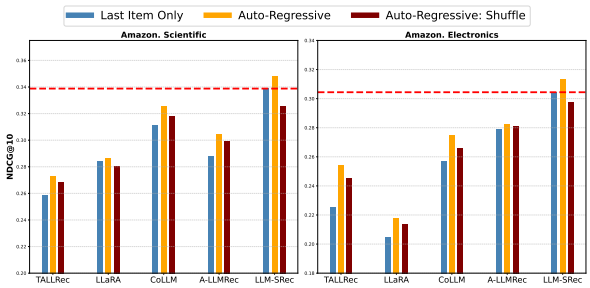

Figure 7: Performance with auto-regressive training strategy.

$$\mathcal{L}_{\text{Distill-Contrastive}} = -\mathop{\mathbb{E}}_{u\in\mathcal{U}} \log \frac{e^{s(f_{user}(\mathbf{h}^u_{\mathcal{U}}), f_{CF-user}(\mathbf{O}_u))}}{\sum_{k\in\mathcal{U}} e^{s(f_{user}(\mathbf{h}^u_{\mathcal{U}}), f_{CF-user}(\mathbf{O}_k))}} \quad (10)$$

Table 12 shows the performance of different distillation loss functions, and we have the following observations: 1) MSE loss (Equation 2) consistently outperforms contrastive loss (Equation 10) across all datasets, indicating that effective sequential information transfer to LLMs requires more than just aligning overall trends. Instead, explicitly matching fine-grained details in representations plays a crucial role in preserving sequential dependencies. 2) Performance degradation occurs when inference is performed on shuffled sequences regardless of the chosen loss function, indicating that both losses successfully captures the sequential information.

E.3 Auto-regressive Training

Recall that, for training efficiency, we only consider the last item in the user sequences as the target item to train LLM-SRec. On the other hand, we can consider all items in the user sequence as the target item to train the models in an auto-regressive manner. As shown in Figure 7, when all the models are trained in an auto-regressive manner, their performance improves, demonstrating the benefits of leveraging more historical interactions. One notable result is that our LLM-SRec without the auto-regressive training outperforms other models with the auto-regressive strategy. This is a notable result as the number of samples used for training is much less without auto-regressive training. This result underscores the efficacy of our framework in capturing sequential patterns. Furthermore, in the shuffled setting, baselines exhibit a relatively small change ratio compared to LLM-SRec, indicating that they still fall short of understanding sequence although the baselines learn the fine-grained item sequences through the auto-regressive manner.