Multi-Grained Patch Training for Efficient LLM-based Recommendation

Jiayi Liao
ljy0ustc@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, China

Ruobing Xie*
xrbsnowing@163.com
Machine Learning Platform Department, Tencent
Beijing, China

Sihang Li
sihang0520@gmail.com
University of Science and Technology of China
Hefei, China

Xiang Wang
xiangwang1223@gmail.com
University of Science and Technology of China
Hefei, China

Xingwu Sun
sunxingwu01@gmail.com
Machine Learning Platform Department, Tencent
Beijing, China

Zhanhui Kang
kegokang@tencent.com
Machine Learning Platform Department, Tencent
Shenzhen, China

Xiangnan He*
xiangnanhe@gmail.com
MoE Key Lab of BIPC, University of Science and Technology of China
Hefei, China

Abstract

Large Language Models (LLMs) have emerged as a new paradigm for recommendation by converting interacted item history into language modeling. However, constrained by the limited context length of LLMs, existing approaches have to truncate item history in the prompt, focusing only on recent interactions and sacrificing the ability to model long-term history. To enable LLMs to model long histories, we pursue a concise embedding representation for items and sessions. In the LLM embedding space, we construct an item's embedding by aggregating its textual token embeddings; similarly, we construct a session's embedding by aggregating its item embeddings. While efficient, this way poses two challenges since it ignores the temporal significance of user interactions and LLMs do not natively interpret our custom embeddings. To overcome these, we propose PatchRec, a multi-grained patch training method consisting of two stages: (1) Patch Pre-training, which familiarizes LLMs with aggregated embeddings – patches, and (2) Patch Fine-tuning, which enables LLMs to capture time-aware significance in interaction history. Extensive experiments show that PatchRec effectively models longer behavior histories with improved efficiency. This work facilitates the practical use of LLMs for modeling long behavior histories.

CCS Concepts

• **Information systems → Recommender systems**.

*Corresponding authors.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730042

Keywords

Sequential Recommendation; Large Language Model; Multi-Grained Compression

ACM Reference Format:
Jiayi Liao, Ruobing Xie, Sihang Li, Xiang Wang, Xingwu Sun, Zhanhui Kang, and Xiangnan He. 2025. Multi-Grained Patch Training for Efficient LLM-based Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3730042

1 Introduction

Large Language Models (LLMs) [5, 7, 11, 29, 37] have demonstrated remarkable success across diverse domains, sparking a growing interest in their application to recommendation. A common paradigm for using LLMs as recommender models (LLM4Rec) [3, 10, 43, 45] involves two key steps: (1) formatting a user's interaction history (*i.e.,* a sequence of previously interacted items) into an input prompt suitable for LLMs, and (2) fine-tuning the LLMs to generate the target response (*i.e.,* predicting the next item of interest). For the first step, most leading methods [3, 22, 23, 39] truncate each interaction history, retaining only the most recent K items[1], with each item represented by the textual metadata (*e.g.,* movie titles like "Gone with the Wind") [2, 3, 22], as shown in Figure 1. This truncation integrates a much shorter interaction sequence into the input prompt, primarily due to LLMs' context window size limitations and the associated computational costs. While effective for capturing short-term preferences, this approach abandons the interactions prior to the selected items, failing to capture the long-term interests encoded in extended item sequences.

To efficiently model long item sequences in LLM4Rec, we introduce a hierarchical representation approach, as depicted in Figure 4. In the embedding space, the **textual tokens** of item titles are firstly compressed into compact **item patches**. These items are then organized into sessions, with the corresponding item patches further compressed into **session patches**, capturing higher-level patterns in user behavior. This compressed representation allows for the

[1]K, with common values being 5, 10, or 20, is typically much smaller than the total length of complete interaction history.

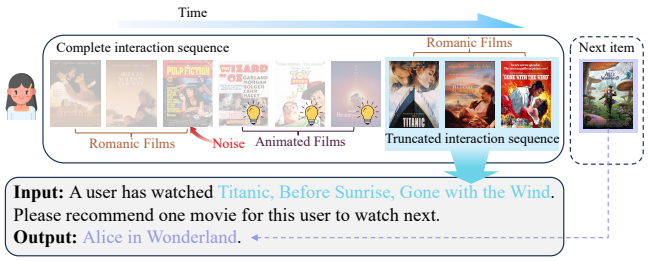


Figure 1: Most LLM4SR works consider the latest-K interactions in the complete user interaction sequence as the truncated historical item sequence fed to LLMs.

accommodation of a significantly larger number of items within a fixed context window. However, the following two challenges remain:

- Temporal significance of interactions: Such compressed representations do not explicitly regard the temporal distinctions of relevance [40], treating all items in a user's interaction sequence as equally important [2, 23]. How can we incorporate mechanisms to capture the varying importance of interactions over time, thereby aligning more closely with evolving user preferences?
- Comprehension of compression patterns: While compressed item and session patches retain the same dimensionality as original textual tokens, LLMs lack inherent understanding of them. How can we design training approaches to enable LLMs to interpret these compressed patterns effectively?

To tackle these challenges, we propose **PatchRec**, a multi-grained patch training framework that unifies: (1) session patches for high-level behavioral patterns, (2) item patches for fine-grained preference modeling, and (3) raw textual tokens for semantic grounding. This framework enables compact yet expressive representation of user interaction history, while maintaining dynamic adaptability to varying sequence lengths. As illustrated in Figure 5, PatchRec operates in the following two stages:

- **Patch Pre-training**. During each training step, the model simultaneously learns both the original uncompressed and compressed versions of the same data sample, thereby establishing a correspondence between the compressed item patches and their original textual tokens.
- **Patch Fine-tuning**. This stage models the temporal variation in item importance using distinct compression granularities. Specifically, items from earlier interactions are compressed to a higher degree, while more recently interacted items are represented by a lower degree of compression, allowing the model to effectively capture longer user histories with varying levels of importance.

Extensive experiments are conducted on datasets and yield the following key findings: (1) For the same truncated interaction sequence, PatchRec uses only 7.34% LLM input tokens compared to TALLRec, while simultaneously improving HR@20 by up to 32% on the Goodreads dataset. (2) For the same computational cost, PatchRec models 3.44 times more user behaviors than TALLRec, resulting in a substantial performance improvement of up to 13% on the MovieLens-1M dataset.

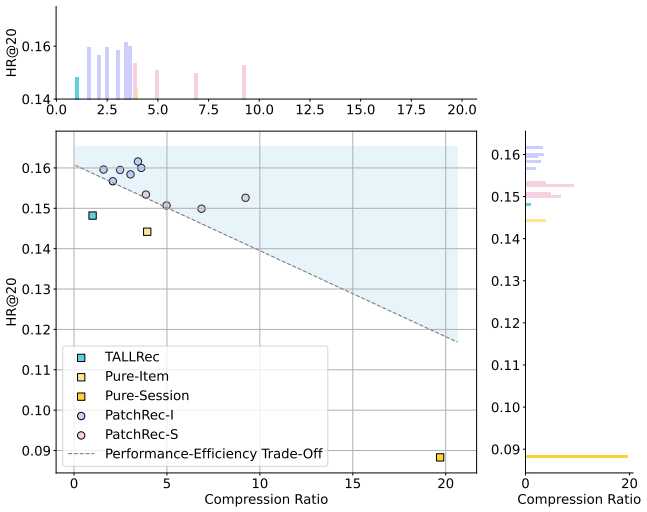


Figure 2: Performance-efficiency trade-off for TALLRec, Pure-Item/Session, and PatchRec-I/S on MovieLens-1M. PatchRec variants demonstrate improvements in both metrics compared to the baseline. The shaded region highlights the area above the performance-efficiency trade-off curve, representing desirable configurations.

Our contributions can be summarized as follows:

(1) We propose a multi-grained (*i.e.,* both item- and session-level) patch training framework — PatchRec, modeling users' long-term historical behaviors in LLM4Rec.
(2) Our proposed pre-training and fine-tuning framework enables LLMs to internalize compressed representations as well as time-aware preferences for sequential modeling.
(3) Experimental results demonstrate that our approach not only saves computational resources but also enhances recommendation performance.

2 Preliminary

Given an input prompt describing a user's chronological interaction history with items, an LLM-based recommender predicts the next item that best matches this user's preference. Building upon a foundational work in LLM-based recommendation, TALLRec [3], we formalize this baseline as supervised finetuning an LLM parameterized by Θ to maximize the autoregressive likelihood:

$$P(Y|X;\Theta) = \prod_{t=1}^{T} P(\mathbf{y}_t|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}; \Theta), \quad (1)$$

where $X = x_{1:N}$ is the LLM input sequence of N tokens representing the user interaction history, and $Y = y_{1:T}$ is the output sequence denoting the next item. The input sequence X encodes a truncated interaction sequence of K items using a structured prompt template. Each item is represented through its title, which is subsequently tokenized into several tokens $\mathbf{x}_i \in \mathbb{R}^d$, where d is the hidden size of the LLM. To adapt TALLRec from click-through rate prediction (answering YES/NO given a target item) to next-item recommendation, we let the LLM generate the next item's title Y token-by-token through autoregressive decoding, employing constrained beam

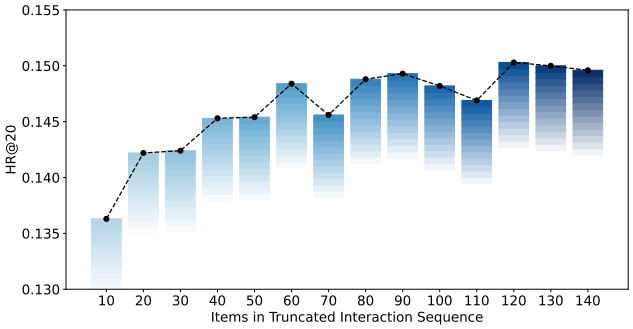


Figure 3: Performance of LLM-based recommender on Movielens-1M at different numbers of items in the truncated interaction sequence. The number of tokens in the historical item sequence ranges from 39 to 552, excluding the tokens of task description.

search over a token-level prefix tree constructed from all candidate item titles. This ensures that generated sequences correspond to valid items.

To investigate the impact of sequence length, we scale up K in preliminary experiments. Figure 3 reveals a key trend: performance improves with truncated sequence length. As the number of items in the truncated interaction sequence increases, recommendation accuracy — HR@20, improves substantially, suggesting that richer user histories enable more precise modeling of preferences. For instance, earlier interactions with animated films contribute to accurately predicting subsequent items like "Alice in Wonderland".

Despite the benefits of modeling long sequences, this baseline approach faces a critical limitation: excessive inference costs due to rapidly growing LLM prompt lengths. The length of the item sequence significantly influences the input prompt length, which is constrained by the context window length of LLMs [33] and the associated computational costs.

3 Method

In this section, we first present the cornerstone of our approach — the hierarchical patching strategy — and then describe the two-stage PatchRec framework.

3.1 Hierarchical Patching

To incorporate more interactions within constraints of LLM prompt length overhead, we hierarchically compress the representation of historical item sequences, as shown in Figure 4.

3.1.1 ***Item Patch Construction.*** In the context of LLM4Rec, the textual tokens of each item title naturally form a semantically coherent unit. To compress these tokens into a single token representation – termed an item patch – we adopt a simple yet effective method of average pooling. Specifically, we compute the mean of the token embeddings corresponding to the item's title as follows:

$$\mathbf{z}_j^i = \frac{1}{|\mathcal{T}_j|} \sum_{t \in \mathcal{T}_j} \mathbf{x}_t, \tag{2}$$

where $\mathbf{z}_j^i \in \mathbb{R}^d$ is the resulting item patch for the j-th item, $\mathcal{T}_j$ denotes the set of tokens in the item's title, and $\mathbf{x}_t$ is the embedding of token t. This average pooling approach compresses token-level information into a compact item-level representation. As shown in Figure 4, a movie title such as "The Bridges of Madison Country" is encoded into a single item patch. This reduces the input length while maintaining the core content of the item description.

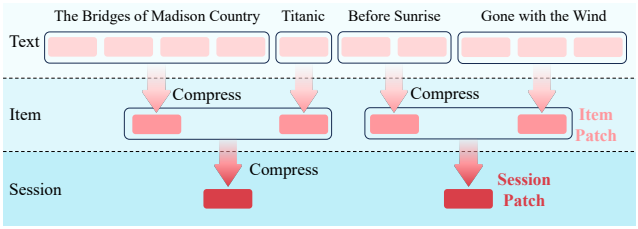


Figure 4: Hierarchical compression. The textual tokens of an item title are aggregated into a compact item patch. Then, several adjacent item patches are further compressed into a denser session patch.

3.1.2 ***Session Patch Construction.*** Since users often exhibit consistent interests over a period of time, consecutive interactions can be further compressed. To accomplish this, we partition a user's historical item sequence into sessions, where each session is formed by grouping a fixed number L of consecutive items [2]. We then obtain a session patch by averaging the embeddings of the item patches within each session:

$$\mathbf{z}_j^s = \frac{1}{|\mathcal{S}_j|} \sum_{t \in \mathcal{S}_j} \mathbf{z}_t^i, \tag{3}$$

where $\mathbf{z}_j^s \in \mathbb{R}^d$ denotes the session patch for the j-th session, $\mathcal{S}_j$ is the set of item patches associated with this session, and $\mathbf{z}_t^i$ represents the embedding of the t-th item patch. As depicted in Figure 4, every two adjacent item patches are further compressed into a denser session patch. This hierarchical compression strategy enables efficient handling of longer sequences while preserving the underlying user interest patterns.

3.1.3 ***Challenges of Patch Training.*** Based on our proposed item and session patch compression methods, we first examine two intuitive adaptations: substituting all text tokens in a sequence with item or session patches only, named Pure-Item and Pure-Session. While effectively reducing LLM input tokens, they ignore the temporal significance of user interactions and the LLM's inherent incomprehension of compression patterns, suggesting clear opportunities for improvement. As shown in Figure 2, these approaches achieve suboptimal performance-efficiency trade-offs, remaining below the Pareto frontier in the upper-right optimization space.

3.2 PatchRec

To accommodate both challenges of time-aware sequence modeling and compression pattern understanding, we propose **PatchRec**, a simple yet effective framework featuring two-stage training, as illustrated in Figure 5. To model the varying relevance of earlier versus later interacted items to a user's current interests, we employ a multi-grained patching strategy. Specifically, earlier interactions

[2]Alternative partitioning methods can also be employed, such as grouping items interacted with during a specific time window.

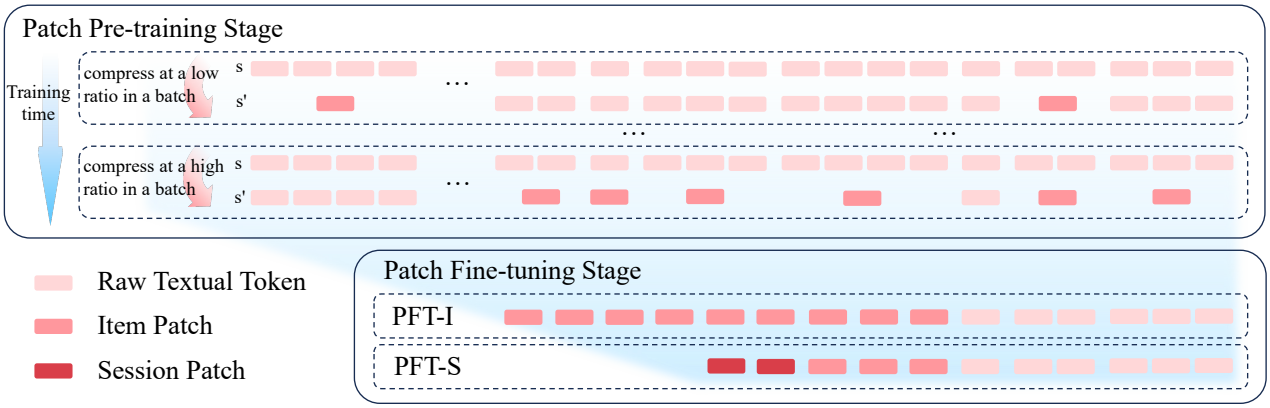


Figure 5: Two-stage training framework of PatchRec. In the patch pre-training stage, we augment each uncompressed sequence s with a compressed version s', where each item is independently and randomly compressed from raw textual tokens into an item patch, in order to build connections between item patches and textual item titles. During the training process, the degree of compression gradually increases from 0 to 1. In the patch finetuning stage, we fine-tune the LLMs with time-aware compressed sequences, allowing the LLM to become familiar with a mixed space of various compression granularities in downstream usage. For each sequence, interactions that occur earlier are compressed to a greater extent, while interactions that occur later are compressed to a lesser extent. PFT-S can achieve higher compression ratio with denser session patches.

are compressed at higher levels (*e.g.,* as item and even session-level patches) for capturing long-term preferences, while most recent interactions retain finer granularity (*e.g.,* textual tokens) for preserving semantic grounding. Since LLMs do not inherently understand the compression pattern from textual tokens to item patches, in the first stage, Patch Pre-training, we train the LLM to learn it by comparing item patches with their corresponding original textual tokens. In the second stage, Patch Fine-tuning, we further adapt the LLM to the final multi-grained compressed format.

3.2.1 ***Patch Pre-training.*** For each interaction sequence in a training batch, as shown in Figure 5, where items are represented by their textual titles, we augment the interaction sequence with a compressed version of it. In this compressed interaction sequence, the textual tokens of each item are independently and randomly compressed into an item patch with probability p. The training objective is formulated as:

$$P(Y|X;\Theta) = P(Y|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1}, \mathbf{z}^i_j, \mathbf{x}_{k+n+1}, \ldots, \mathbf{x}_N;\Theta), \quad (4)$$

where the j-th item' title tokens $\mathbf{x}_{[k:k+n]}$ are compressed into an item patch $\mathbf{z}^i_j$ as defined in Equation 2. The probability p gradually increases from 0 to 1 during training, following the schedule:

$$p = \frac{\tau}{T}, \quad (5)$$

where τ denotes the current training step, and T represents the total number of training steps. By employing p as a scheduler, the item representations in the compressed version progressively transition from textual tokens to item patches over the course of training.

Within each batch, interaction sequences in both raw and compressed formats are presented simultaneously. This approach establishes a clear correspondence for each interaction, effectively linking the textual tokens to their respective item patches.

3.2.2 ***Patch Fine-tuning.*** To incorporate the temporal decay of user interests in a sequence, we propose two fine-tuning strategies with different compression granularities: Patch Fine-tuning on Items (PFT-I) and Patch Fine-tuning on Sessions (PFT-S). As shown in Figue 5, they are designed to accommodate varying computational resource constraints.

PFT-I retains the most recent M items [3] from a truncated interaction sequence of K items, represented by their textual tokens. The earlier $(K - M)$ items are compressed into item patches. The training objective is formulated as:

$$P(Y|X;\Theta) = P(Y|\mathbf{z}^i_1, \mathbf{z}^i_2, \ldots, \mathbf{z}^i_{K-M}, \mathbf{x}_j, \ldots, \mathbf{x}_N;\Theta), \quad (6)$$

where $\mathbf{x}_j$ denotes the start textual token of the $(K - M + 1)$-th item and N is the total number of tokens across all K truncated items. PFT-I achieves a moderate compression ratio.

PFT-S divides the truncated historical item sequence into groups of items based on interaction time, with each group containing at most L adjacent items. The interactions in the latest group are retained as original textual tokens, while those in the second-latest group are compressed into item patches. All earlier groups are further compressed into the densest session patches. The training objective is formulated as:

$$P(Y|X;\Theta) = P(Y|\mathbf{z}^s_1, \mathbf{z}^s_2, \ldots, \mathbf{z}^i_1, \mathbf{z}^i_2, \ldots, \mathbf{x}_1, \mathbf{x}_2, \ldots, ;\Theta). \quad (7)$$

PFT-S achieves a higher compression ratio.

Fine-tuning the LLM on these multi-grained compressed interaction sequences enables it to effectively capture time-aware user interest patterns, striking a balance between representation effectiveness and computational efficiency.

[3] M is a predefined hyperparameter, which controls the overall compression ratio.

Table 1: Statistics of datasets. Int.: Interactions. SL.: Items per user sequence. TT.: Title tokens per item.

Dataset	# User	# Item	# Int.	# Max SL.	# Avg. SL.	# Avg. TT.
MovieLens-1M	6,037	3,883	575,281	1,435	95.28	3.94
Goodreads	2,090	8,793	162,539	430	77.77	9.82
MovieLens-100K	682	1,682	51,824	378	75.99	3.86

Table 2: Performance comparison with 100 items in the truncated user sequence. CR is short for compression ratio. Bold and underlined indicate the best and the second-best results, respectively. PatchRec improves recommendation accuracy with less computational demands. *(p-value $\ll$ 0.05)

	MovieLens-1M*					Goodreads*					MovieLens-100K				
Model	**HR@10**	**N@10**	**HR@20**	**N@20**	**CR**	**HR@10**	**N@10**	**HR@20**	**N@20**	**CR**	**HR@10**	**N@10**	**HR@20**	**N@20**	**CR**
GRU4Rec	0.0623	0.0298	0.1106	0.0419	-	0.0401	0.0219	0.0592	0.0267	-	0.0661	0.0307	0.1211	0.0445	-
Caser	0.0384	0.0180	0.0712	0.0262	-	0.0116	0.0057	0.0220	0.0083	-	0.0501	0.0234	0.0910	0.0336	-
SASRec	0.0594	0.0286	0.1035	0.0397	-	0.0404	0.0229	0.0569	0.0271	-	0.0605	0.0290	0.1069	0.0406	-
LinRec	0.0622	0.0296	0.1063	0.0407	-	0.0251	0.0125	0.0398	0.0161	-	0.0619	0.0283	0.1048	0.0393	-
Mamba4Rec	0.0648	0.0303	0.1116	0.0421	-	0.0281	0.0146	0.0450	0.0188	-	0.0667	0.0309	0.1086	0.0415	-
MoRec	0.0341	0.0164	0.0616	0.0233	-	0.0122	0.0061	0.0236	0.0089	-	0.0383	0.0184	0.0762	0.0279	-
TALLRec	0.0933	0.0396	0.1482	0.0491	1.00	0.0563	0.0236	0.0770	0.0249	1.00	0.0986	0.0443	0.1680	**0.0611**	1.00
PatchRec-I	**0.1058**	**0.0455**	**0.1616**	**0.0525**	3.44	0.0733	0.0289	0.0976	0.0293	6.81	0.0967	0.0416	**0.1738**	0.0574	3.38
PatchRec-S	0.0967	0.0408	0.1526	0.0496	**9.23**	**0.0748**	**0.0295**	**0.1013**	**0.0301**	**13.62**	**0.1016**	**0.0450**	0.1680	0.0554	**3.85**

4 Experiments

In this section, we present a comprehensive comparison of PatchRec against various baseline methods across real-world datasets, demonstrating the effectiveness and efficiency of PatchRec under different levels and granularities of compression. Additionally, we analyze the mechanisms through which our approach achieves its superior performance and its performance in user groups with more long-term behaviors.

4.1 Experimental Settings

Datasets. We conduct extensive experiments on three real-world benchmark datasets: MovieLens-1M, Goodreads, and MovieLens-100K. The MovieLens datasets are widely used benchmarks, sourced from a movie recommendation platform [4], while the Goodreads dataset is derived from the largest online community for readers and book recommendations [5].

To ensure data quality, we apply filtering based on rating thresholds of 3 and 5 for MovieLens and Goodreads, respectively. For MovieLens-100K, we retain only users with more than 20 interactions. For Goodreads, we exclude users who have interacted with fewer than 50 items and items that have received fewer than 10 interactions from distinct users. To prevent information leakage and balance computational overhead during training and evaluation, we partition the interactions in each dataset into training, validation, and test splits, following a temporal split ratio of 48:1:1 based on timestamps. The dataset statistics are summarized in Table 1.

Implementations. We adopt Llama-3.2-1B-Instruct [7] as the backbone LLM, fine-tuning all its parameters. Each training stage consists of one epoch and requires approximately 12 NVIDIA A800 GPU hours. To balance computational efficiency and resource allocation, we use a training batch size of 64 for MovieLens datasets and 32 for Goodreads, and a test batch size of 16 for all three datasets. The training employs a cosine learning rate scheduler with a warmup ratio of 0.05, and the learning rates are set to 8e-6,5e-5,1e-5 for MovieLens-1M, Goodreads, and MovieLens-100K, respectively. For each interaction, the latest K historical interactions are selected as the truncated user sequence. All experiments access item titles that appear within their respective datasets to ensure fair evaluation.

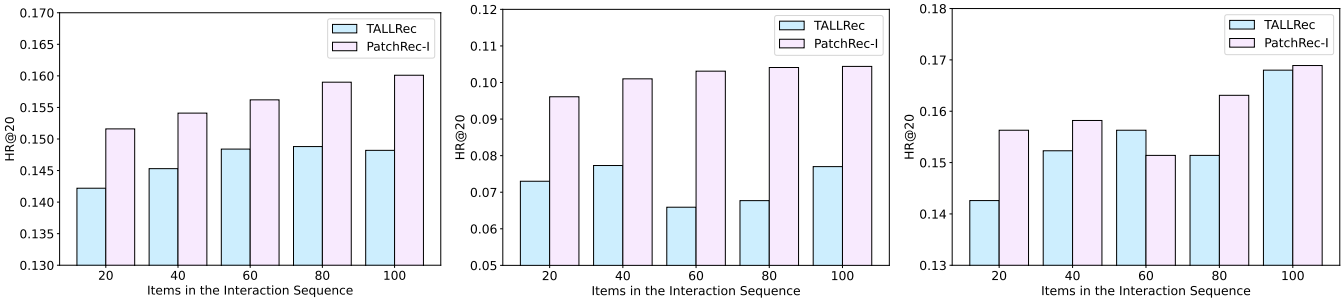
Evaluation. To evaluate recommendation performance, we use two key metrics: HitRatio@10/20 and NDCG@10/20. Additionally, to assess efficiency, we introduce the **Compression Ratio**, defined as the ratio of the number of tokens in the historical item sequence before compression to the number after compression. A higher compression ratio indicates greater efficiency.

4.2 Main Performance Comparison

4.2.1 ***Baselines.*** We denote the models trained using patch pre-training and PFT-I as **PatchRec-I**, and those trained with patch pre-training and PFT-S as **PatchRec-S**. They are compared against traditional recommender models, efficient long-term sequential recommender models, and LLM-based recommenders.

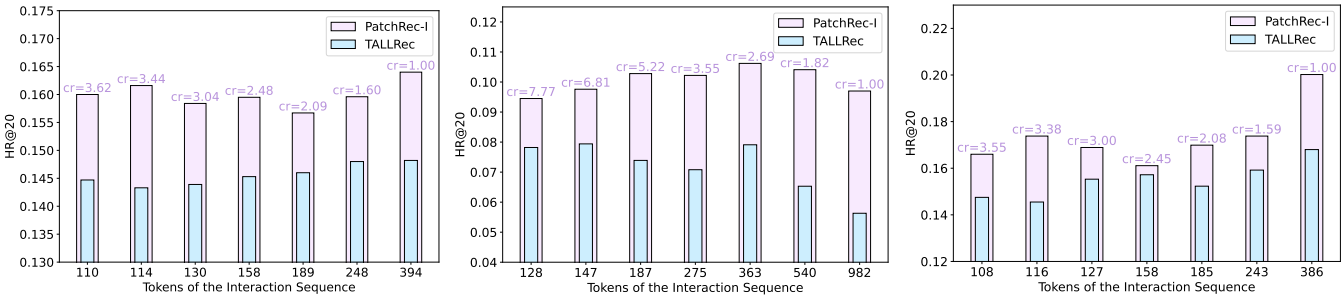
- **Traditional Recommender Models.** We select three sequential recommender models as representatives of different mechanisms for capturing user behavior patterns: GRU4Rec[14] (RNN-based), Caser[34] (CNN-based) and SASRec[20] (attention-based) [6].
- **Efficient Long-term Recommender Models.** LinRec [26] and Mamba4Rec [25] are chosen as representatives of efficient long-term sequential recommendation methods. LinRec leverages linear attention mechanisms, while Mamba4Rec employs Mamba blocks [12] for efficient modeling of long-term user behaviors.

[4] https://movielens.org/
[5] https://www.goodreads.com/
[6] These models are implemented following [38].



(a) MovieLens-1M (b) Goodreads (c) MovieLens-100K

Figure 6: Performance comparison between PatchRec-I with TALLRec with the same item numbers in interaction sequence. The compression ratios of PatchRec-I are 2.27, 3.06, and 2.25 for MovieLens-1M, Goodreads, and MoviLens-100K, respectively. Impressively, with improved efficiency, PatchRec-I does not compromise recommendation accuracy.



(a) MovieLens-1M (b) Goodreads (c) MovieLens-100K

Figure 7: Performance comparison between PatchRec-I with TALLRec with comparable numbers of tokens in the interaction sequence. The item number in the interaction sequence is 100 for PatchRec-I, with various compression ratios (cr). For PatchRec-I, the latest-M (M=3, 5, 10, 20, 30, 50, 100) items are represented with textual titles, and earlier items are compressed into item patches. PatchRec-I delivers superior recommendation performance compared to TALLRec at the same computational cost.

- **LLM-based Recommenders.** We include two representative methods that incorporate LLMs in sequential recommendation: (1) MoRec [7] [42] enhances traditional recommenders by incorporating textual features encoded by LLMs. (2) TALLRec [3] applies SFT on truncated interaction sequences without compression. [8]

4.2.2 ***Results***. The truncation lengths of all the methods compared in Table 2 are 100 (*i.e.,* modeling the latest-100 items for each sequence). For PatchRec-I, we keep the latest-M (M=5) interactions as items' textual titles and compress the early interactions into item patches. For PatchRec-S, we use a group size L of 5 for MovieLens-1M and Goodreads, and 20 for MovieLens-100K; so the latest-L interactions are retained as textual tokens, the L interactions in the second latest group are compressed into item patches, and the earlier interactions are compressed into session patches.

The results in Table 2 demonstrate that both PatchRec-I and PatchRec-S outperform all traditional recommender, efficient long-term sequential recommender, and LLM-based recommender baselines across all four metrics on MovieLens-1M and GoodReads. And on MovieLens-100K, with comparable recommendation accuracy, PatchRec achieves a much short input sequence.

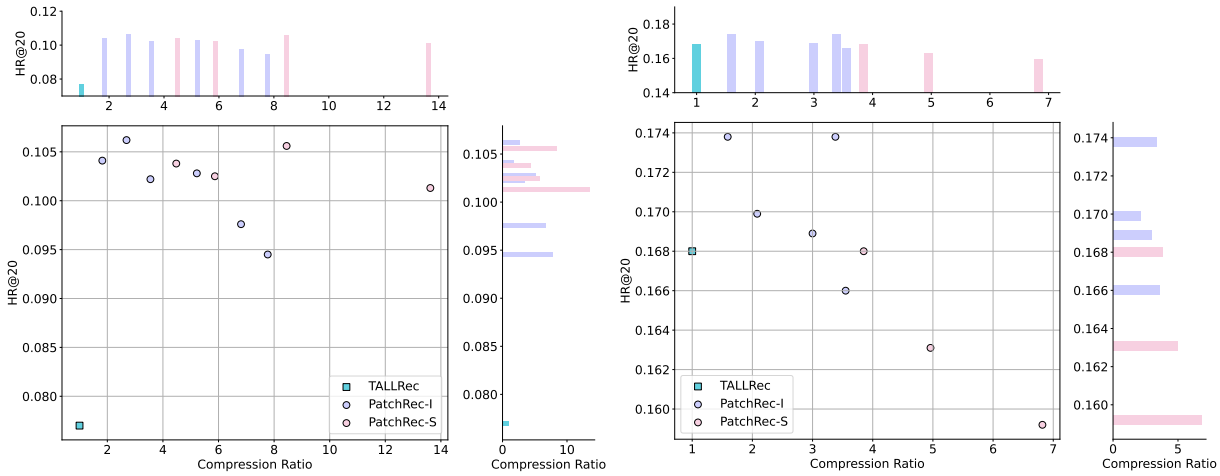
Notably, PatchRec-I achieves HR@20 improvements of 9.04%, 26.75%, and 3.45% over TALLRec, with compression ratios of 3.44, 6.81, and 3.38 on MovieLens-1M, Goodreads, and MovieLens-100K, respectively. Similarly, PatchRec-S achieves HR@20 improvements of 2.97% and 31.56% on MovieLens-1M and Goodreads, respectively, with higher compression ratios of 9.23 and 13.62. On MovieLens-100K, PatchRec-S achieves comparable performance to TALLRec with a compression ratio of 3.85.

These results demonstrate that both PatchRec-I and PatchRec-S significantly reduce the number of LLM input tokens without compromising recommendation performance. Overall, the experimental findings highlight the effectiveness of our approach in simultaneously improving recommendation accuracy and reducing computational overhead.

4.3 Analysis of PatchRec-I

To further assess the performance of PatchRec-I in details, we compare it with TALLRec under two distinct experimental settings across three datasets, addressing the following research questions:

[7] We adopt BERT as the encoder for items' textual metadata and SASRec as the recommender backbone.
[8] We adapt TALLRec to the all-ranking task with constrained beam search in item title space, consistent with PatchRec



(a) Goodreads (b) MovieLens-100K

Figure 8: The performance-efficiency trade-off of TALLRec, PatchRec-I, and PatchRec-S on Goodreads and MovieLens-100K dataset. In the scatter plot, points that are positioned further towards the upper right corner indicate better performance and higher efficiency. Both PatchRec-I and PatchRec-S provide a better performance-efficiency trade-off compared to TALLRec.

- **RQ1:** Does compression result in a loss of recommendation performance when processing identical interaction sequences?
- **RQ2:** For a fixed inference computational cost, does compression improve recommendation performance?

4.3.1 ***Performance Comparison under the Same Items in the Interaction Sequence (RQ1).*** We maintain the same truncation length for interaction sequences, with item counts of 20, 40, 60, 80, and 100, and compare HR@20 of PatchRec-I with TALLRec to examine whether the compression of early items in PatchRec-I leads to any loss of recommendation performance. The results are shown in Figure 6.

On the MovieLens-1M and Goodreads datasets, PatchRec-I consistently outperforms TALLRec, with relative gains exceeding 5% and 30%, respectively. This improvement may be attributed to the increased information density in the compressed item patches, which mitigates the influence of noisy interactions. On the MovieLens-100K dataset, PatchRec-I achieves performance comparable to TALLRec, indicating that the proposed compression strategy does not compromise recommendation accuracy.

Furthermore, under a fixed compression ratio, we observe that as the truncation length (*i.e.,* the items in the interaction sequence) increases across the three datasets, the recommendation performance of PatchRec-I generally improves, unlike TALLRec, which doesn't show steady improvement. This trend suggests that, for PatchRec-I, incorporating more historical interactions enhances the ability to model long-term user behaviors effectively.

4.3.2 ***Performance Comparison under Comparable Numbers of Tokens (RQ2).*** We compare the recommendation performance of PatchRec-I with TALLRec with comparable token counts for LLM input to evaluate the impact of compression on recommendation performance at an equivalent computational cost. The results are shown in Figure 7.

In all experiments across the three datasets, PatchRec-I processes 100 items, which are represented as item patches or raw textual titles. However, due to varying compression ratios, the number of tokens required differs. For each bar in the figures, both PatchRec-I and TALLRec input the same number of tokens. Despite this parity in token usage, PatchRec-I consistently achieves higher HR@20 than TALLRec, with relative improvements of at least 7%, 20%, and 2%, and reaching up to 13%, 72%, and 19% on MovieLens-1M, Goodreads, and MovieLens-100K, respectively.

These results demonstrate that across all three datasets, PatchRec-I delivers superior recommendation performance compared to TALLRec at the same computational cost, underscoring the efficiency and effectiveness of the proposed compression method.

4.4 Analysis of PatchRec-S

PatchRec-S further compresses the earliest item patches into session patches, thereby achieving even higher compression ratio. To validate its effectiveness, we compare the compression ratio and HR@20 of three methods — TALLRec, PatchRec-I, and PatchRec-S — while maintaining the same number of 100 items in the interaction sequence across different compression ratios on three datasets. The results are shown in Figure 2 and Figure 8.

The figures show that the distribution of PatchRec-I and PatchRec-S points is positioned further towards the upper-right corner compared to TALLRec across all three datasets. Specifically:

- PatchRec-S consistently outperforms TALLRec in HR@20 across various compression ratios on the MovieLens-1M and Goodreads datasets. Even at compression ratios as high as 9.23 and 13.62 for MovieLens-1M and Goodreads, respectively, PatchRec-S maintains superior HR@20 compared to TALLRec.
- PatchRec-I points are generally positioned higher on the performance axis, while PatchRec-S points are shifted further to the right, reflecting greater efficiency.

Table 3: The performance comparison of PatchRec-I/S between with and without patch pre-training on MovieLens-1M dataset. The patch pre-training stage improves recommendation performance.

Method	HR@10	N@10	HR@20	N@20
PatchRec-I				
w/o Patch Pre-training	0.1014	0.0439	0.1550	0.0504
w/ Patch Pre-training	**0.1058**	**0.0455**	**0.1616**	**0.0525**
PatchRec-S				
w/o Patch Pre-training	0.0920	0.0386	0.1436	0.0463
w/ Patch Pre-training	**0.0967**	**0.0408**	**0.1526**	**0.0496**

Table 4: Performance comparison between TALLRec, patch pre-training, and data augmentation with dropout on MovieLens-1M dataset. The enhancements achieved by compression are not simply due to the random exclusion of items.

Method	HR@10	N@10	HR@20	N@20
TALLRec	0.0933	0.0396	0.1482	0.0491
Dropout	0.0936	0.0402	0.1480	0.0492
Patch Pre-training	**0.1026**	**0.0431**	**0.1610**	**0.0532**

These results indicate that both PatchRec-I and PatchRec-S provide a better performance-efficiency trade-off compared to TALLRec. Furthermore, the distinction between the two methods demonstrates their complementary strengths: PatchRec-I prioritizes performance, while PatchRec-S emphasizes efficiency.

4.5 Effectiveness of the Patch Pre-training Stage

In this section, we ablate to demonstrate the benefits of the patch pre-training stage.

4.5.1 ***Impact of the Patch Pre-training Stage.*** To evaluate the importance of the patch pre-training stage, we compare the recommendation performance of PatchRec-I and PatchRec-S under two settings: (1) Starting from LLM checkpoints trained with the patch pre-training stage, which is denoted as w/ patch pre-training. (2) Directly initialized from the Llama-3.2-1B-Instruct checkpoint without patch pre-training, denoted as w/o patch pre-training. The evaluation is conducted on the MovieLens-1M dataset, and the results are shown in Table 3. From the table, we can observe that both PatchRec-I and PatchRec-S with the patch pre-training stage consistently outperforms those without that stage.

This demonstrates the critical role of the patch pre-training stage in improving performance. During this stage, the inclusion of both uncompressed textual tokens (*i.e.,* item titles) and compressed item patches within the same batch enables the LLM to effectively learn the patterns of item-level compression. This adaptation allows the LLM to transition seamlessly from the textual token space to the compressed item patch space, laying a robust foundation for the subsequent multi-grained patch fine-tuning stage.

4.5.2 ***On Data Augmentation.*** To examine the performance improvements attributed to the compression operations in the patch pre-training stage, we compare our compression approach against a baseline method that directly drops the items compressed during the patch pre-training data augmentation process.

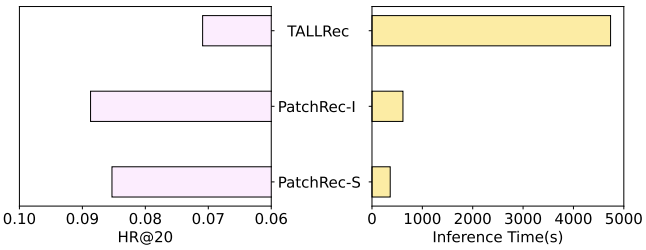


Figure 9: Performance (*i.e.,* HR@20) and efficiency (*i.e.,* inference time) comparison of TALLRec, PatchRec-I, and PatchRec-S for long sequences (*i.e.,* users with at least 500 behaviors) on MovieLens-1M. PatchRec enhances both performance and efficiency in modeling long-term user behaviors.

The experimental results, as summarized in Table 4, reveal that patch pre-training consistently outperforms both the TALLRec (*i.e.,* no-compression) and dropout baselines. This demonstrates that the enhancements achieved by our compression strategy are not simply due to the random exclusion of items. Our compression design retains critical patterns while increasing the effective information density of the input. These findings highlight the strength of our method in balancing efficiency and effectiveness, ensuring that compression enhances.

4.6 Performance and Efficiency for Long-term User Behaviors

To further evaluate the performance and efficiency of PatchRec when applied to even longer sequences, we analyze user groups with more than 500 interactions in MovieLens-1M. This experiment was conducted using a Nvidia A100 GPU with 40 GB of memory. For each test interaction, we maximize the number of historical items visible to all three methods (*i.e.,* TALLRec, PatchRec-I, and PatchRec-S) to fully utilize the available GPU memory, which simulates the pragmatism in real-world recommender systems that make full use of all available computing sources and data. Specifically, we truncate the Latest-K (K=150, 350) historical items for both TALLRec and PatchRec-I, whereas PatchRec-S is capable of accommodating all historical items (1435 at most) within the 40 GB GPU memory constraints.

In the case of PatchRec-I, the latest 20 interactions are retained as textual titles, while earlier interactions are compressed into item patches. For PatchRec-S, the parameter L, which denotes the number of items in a session, is set to 20. The experimental results are shown in Figure 9. The figure demonstrates that PatchRec-I achieved a 25.11% improvement in HR@20 with only 12.99% inference time, compared to TALLRec. Similarly, PatchRec-S achieved a 20.31% increase in HR@20 relative to TALLRec, with a even higher 13.09-fold reduction in inference time. These results indicate that our proposed methods significantly enhance both performance and efficiency in modeling long-term user behaviors, implying that our PatchRec is capable of modeling practical historical behavior lengths via LLM4Rec.

5 Related Work

5.1 LLM for Recommendation

With LLMs demonstrating vast world knowledge and impressive reasoning capabilities across various domains [5, 7, 11, 29, 32, 37], their potential for recommendation tasks has garnered significant attention from researchers [17, 43]. Existing work on LLM4Rec can be roughly classified into two categories: (1) LLMs as enhancers: LLMs augment traditional recommender models by processing textual features associated with users and items [15, 42], and (2) LLMs as recommenders: LLMs directly generate recommendations for users' next interactions [3, 8, 10, 31].

The primary focus of this paper is on the latter — LLMs as recommenders — where the core challenge is converting user-item interaction sequences into a format suitable for language modeling. Current item indexing approaches in LLM4Rec primarily fall into two types: (1) Text-based representations, where items are indexed using textual information, such as item titles [2, 3], descriptions [21], or learned semantic IDs [18, 19]; (2) Collaborative-incorporated representations, which consider collaborative signals [23, 46] in item embeddings.

For both aforementioned types, each item is typically represented by multiple tokens. Due to the quadratic computational complexity $O(n^2)$ of self-attention mechanism in LLMs *w.r.t.* sequence length [9], using multiple tokens to represent one item in a user's interaction sequence becomes computationally demanding. This motivates us to explore more efficient item and even sequence representations, aiming to improve the performance-efficiency trade-off.

5.2 Long Sequence Modeling in LLM4Rec

Recent efforts have made significant strides in addressing the context length limitations of LLMs [6, 27, 30, 33, 44]. One prominent line of work focuses on context compression, wherein older or less critical information is compressed into a more compact representation. Early method [4] trains LLMs to summarize or encode lengthy preceding text into compact vectors that can be prefixed to the prompt. Similarly, ICAE [9] introduces an in-context autoencoder that encodes long contexts and then reconstructs necessary details when needed. More recent studies [28, 41] explore prompt compression, demonstrating that selectively retaining only task-relevant facts can preserve performance while substantially reducing input length. Broadly, these compression-based techniques aim to prevent earlier information from overwhelming the model's capacity as context grows. In contrast to these approaches, PatchRec learns concise embeddings at multiple granularities, enabling efficient modeling of extended histories without relying on explicit textual compression or summarization.

The integration of LLMs into recommendation systems introduces further challenges, particularly the restriction on modeling long user behavior sequences due to limited context length and computational resource constraints [13, 16, 35, 36]. One solution [1, 47] enhances LLMs with external long-term memory modules that summarize user interaction histories and maintain dynamically updated profiles. Rather than inputting every past interaction, MemoryBank [47] maintains a running summary that the model can consult when needed. PURE [1] introduces a "Review Extractor" and "Profile Updater" continuously distill evolving user histories into a compact form. Another notable approach, ReLLA [24] tackles the challenge by retrieving the top-K interacted items from a user's history based on semantic relevance to the target item. Distinct from these memory- and retrieval-based methods, PatchRec directly trains LLMs to internalize long-term historical signals through aggregated embeddings, eliminating the need for explicit summarization or retrieval mechanisms.

[9]The computational complexity of vanilla attention mechanisms is $O(n^2)$. Recent studies proposed methods to reduce it to approximately $O(n)$

6 Limitation

While PatchRec shows a strong performance-efficiency trade-off, several limitations remain, especially for real-world deployment:

- **Inference Latency.** Despite compression, reliance on LLMs still incurs high inference costs, limiting real-time applicability. Future work can explore model distillation, quantization, or early-exit strategies to reduce latency.
- **Compression Design.** PatchRec is simple and effective, but more expressive designs — such as advanced patching or innovative grouping techniques — could be further investigated.
- **Adaptive Granularity.** We currently use a fixed multi-grained compression scheme. Dynamically adjusting granularity based on resource constraints or user patterns is a promising direction.

Addressing these limitations will be key to making PatchRec more practical and scalable for real-world recommendation systems.

7 Conclusion

In this paper, we introduce PatchRec, a simple yet effective compression framework designed for multi-grained modeling of users' extensive historical behaviors in LLM-based recommendation (LLM4Rec). To address the significant challenges posed by LLM constraints and the temporal dynamics of user interactions — mostly untouched in prior works — PatchRec leverages a hierarchical and time-aware compression strategy, enabling a deeper and more nuanced understanding of user preferences. By adopting a two-stage training process, PatchRec effectively adapts LLMs to operate within a multi-grained sequence representation space. Extensive experiments on benchmark datasets demonstrate that PatchRec not only dramatically reduces the input token length for LLMs but also consistently improves recommendation performance.

This work paves the way for future research into more efficient and effective ways to harness the power of LLMs in sequential recommendation. We hope PatchRec will inspire further exploration into efficient LLM-based sequential recommenders for lifelong user sequence modeling.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (92270114, U24B20180, 62121002) and the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001).

References

[1] Seunghwan Bang and Hwanjun Song. 2025. LLM-based User Profile Management for Recommender System. *arXiv preprint arXiv:2502.14541* (2025).
[2] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnan He, and Qi Tian. 2023. A Bi-Step Grounding Paradigm

for Large Language Models in Recommendation Systems. *CoRR* abs/2308.08434 (2023).

[3] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. In *RecSys*. ACM, 1007–1014.

[4] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 3829–3846.

[5] DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[6] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. LongRoPE: extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*. 11091–11104.

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, and et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024).

[8] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. *CoRR* abs/2303.14524 (2023).

[9] Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. [n. d.]. In-context Autoencoder for Context Compression in a Large Language Model. In *The Twelfth International Conference on Learning Representations*.

[10] 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 *RecSys*. ACM, 299–315.

[11] Gemini Team Google. 2023. Gemini: A Family of Highly Capable Multimodal Models. *CoRR* abs/2312.11805 (2023).

[12] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752* (2023).

[13] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 720–730.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR (Poster)*.

[15] 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 *KDD*. ACM, 585–593.

[16] 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.

[17] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian J. McAuley, and Wayne Xin Zhao. 2024. Large Language Models are Zero-Shot Rankers for Recommender Systems. In *ECIR (2) (Lecture Notes in Computer Science, Vol. 14609)*. Springer, 364–381.

[18] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to Index Item IDs for Recommendation Foundation Models. In *SIGIR-AP*. ACM, 195–204.

[19] Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, Suhang Wang, Jiawei Han, and Xianfeng Tang. 2024. Language Models as Semantic Indexers. In *ICML*. OpenReview.net.

[20] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. IEEE Computer Society, 197–206.

[21] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian J. McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In *KDD*. ACM, 1258–1267.

[22] Jiayi Liao, Xiangnan He, Ruobing Xie, Jiancan Wu, Yancheng Yuan, Xingwu Sun, Zhanhui Kang, and Xiang Wang. 2024. RosePO: Aligning LLM-based Recommenders with Human Values. *CoRR* abs/2410.12519 (2024).

[23] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. LLaRA: Large Language-Recommendation Assistant. In *SIGIR*. ACM, 1785–1795.

[24] Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. ReLLa: Retrieval-enhanced Large Language Models for Lifelong Sequential Behavior Comprehension in Recommendation. In *WWW*. ACM, 3497–3508.

[25] Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. 2024. Mamba4Rec: Towards Efficient Sequential Recommendation with Selective State Space Models. *CoRR* abs/2403.03900 (2024).

[26] Langming Liu, Liu Cai, Chi Zhang, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Yifu Lv, Wenqi Fan, Yiqi Wang, Ming He, Zitao Liu, and Qing Li. 2023. LinRec: Linear Attention Mechanism for Long-term Sequential Recommender Systems. In *SIGIR*. ACM, 289–299.

[27] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173.

[28] Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems* 36 (2023), 19327–19352.

[29] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).

[30] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. [n. d.]. Compressive Transformers for Long-Range Sequence Modelling. In *International Conference on Learning Representations*.

[31] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. In *NeurIPS*.

[32] Chenze Shao, Fandong Meng, and Jie Zhou. 2024. Patch-Level Training for Large Language Models. arXiv:2407.12665 [cs.CL] https://arxiv.org/abs/2407.12665

[33] Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing* 568 (2024), 127063.

[34] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. ACM, 565–573.

[35] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI*. ijcai.org, 6332–6338.

[36] Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. 2025. R^3 Mem: Bridging Memory Retention and Retrieval via Reversible Compression. *arXiv preprint arXiv:2502.15957* (2025).

[37] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, and et al. 2024. Qwen2 Technical Report. *CoRR* abs/2407.10671 (2024).

[38] Zhengyi Yang, Xiangnan He, Jizhi Zhang, Jiancan Wu, Xin Xin, Jiawei Chen, and Xiang Wang. 2023. A Generic Learning Framework for Sequential Recommendation with Distribution Shifts. In *SIGIR*. ACM, 331–340.

[39] 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. *CoRR* abs/2310.20487 (2023).

[40] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Networks. In *IJCAI*. ijcai.org, 3926–3932.

[41] Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. CompAct: Compressing Retrieved Documents Actively for Question Answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 21424–21439.

[42] 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. In *SIGIR*. ACM, 2639–2649.

[43] 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. *CoRR* abs/2311.02089 (2023).

[44] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems* 33 (2020), 17283–17297.

[45] Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems* (2023).

[46] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. CoLLM: Integrating Collaborative Embeddings Into Large Language Models for Recommendation. *IEEE Trans. Knowl. Data Eng.* 37, 5 (2025), 2329–2340.

[47] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19724–19731.