Accepted: 02 December 2025
Revised: 14 September 2025
Received: 12 November 2024

RESEARCH-ARTICLE

Large Language Model Can Interpret Latent Space of Sequential Recommender

ZHENGYI YANG, University of Science and Technology of China, Hefei, Anhui, China

JIANCAN WU, University of Science and Technology of China, Hefei, Anhui, China

YANCHEN LUO, University of Science and Technology of China, Hefei, Anhui, China

JIZHI ZHANG, University of Science and Technology of China, Hefei, Anhui, China

YANCHENG YUAN, The Hong Kong Polytechnic University, Hong Kong, Hong Kong, Hong Kong

AN ZHANG, University of Science and Technology of China, Hefei, Anhui, China

View all

Open Access Support provided by:

University of Science and Technology of China

The Hong Kong Polytechnic University

Large Language Model Can Interpret Latent Space of Sequential Recommender

ZHENGYI YANG, University of Science and Technology of China, Hefei, China
JIANCAN WU, University of Science and Technology of China, China
YANCHEN LUO, University of Science and Technology of China, China
JIZHI ZHANG, University of Science and Technology of China, China
YANCHENG YUAN, The Hong Kong Polytechnic University, China
AN ZHANG, University of Science and Technology of China, China
XIANG WANG, University of Science and Technology of China, China
XIANGNAN HE*, MoE Key Lab of BIPC, University of Science and Technology of China, Hefei, China

Sequential recommendation aims to predict the next item of interest for a user, based on her/his interaction history. In conventional sequential recommenders, a common approach is to learn sequence representations based on ID embeddings of items, which can be leveraged to predict the subsequent items of interest. Clearly, the sequence representations encode user behavioral patterns, which are critical to recommendation. Inspired by recent success in empowering large language models (LLMs) to understand diverse modality (*e.g.,* image, audio), a compelling question arises: "Can LLMs understand and utilize representations from conventional recommenders?". To answer this, we propose RecInterpreter, which examines the capacity of LLMs to decipher the representation space of pretrained recommenders. Specifically, with the multimodal pairs (*i.e.,* interaction sequence representations and text narrations), RecInterpreter first uses a lightweight projector to map the representations into the token embedding space of the LLM, encouraging LLM to generate textual narrations for items within the sequence. Furthermore, upon interpreting recommenders, LLM can enhance its recommendation capabilities through fine-tuning with the projected representations, even without textual description of interaction sequences. Experiments showcase that RecInterpreter enhances LLMs to understand hidden representations from ID-based sequential recommenders and better accomplish recommendation task with the explicitly understanding of behavior patterns.

CCS Concepts: • **Information systems → Recommender System**.

Additional Key Words and Phrases: Recommendation, Large Language Models

1 INTRODUCTION

Sequential recommendation — predicting the next item of interest based on the historical interaction of a user — has been a fundamental task in both academia and industry [8, 23, 31]. Scrutinizing leading sequential

*Corresponding author.

Authors' Contact Information: Zhengyi Yang, University of Science and Technology of China, Hefei, China, yangzhy@mail.ustc.edu.cn; Jiancan Wu, University of Science and Technology of China, China, wujcan@gmail.com; Yanchen Luo, University of Science and Technology of China, China, luoyanchen@mail.ustc.edu.cn; Jizhi Zhang, University of Science and Technology of China, China, cdzhangjizhi@mail.ustc.edu.cn; Yancheng Yuan, The Hong Kong Polytechnic University, China, yancheng.yuan@polyu.edu.hk; An Zhang, University of Science and Technology of China, China, an_zhang@ustc.edu.cn; Xiang Wang, University of Science and Technology of China, China, xiangwang1223@gmail.com; Xiangnan He, MoE Key Lab of BIPC, University of Science and Technology of China, Hefei, China, xiangnanhe@gmail.com.

ACM 1558-2868/2025/12-ART
https://doi.org/10.1145/3786201

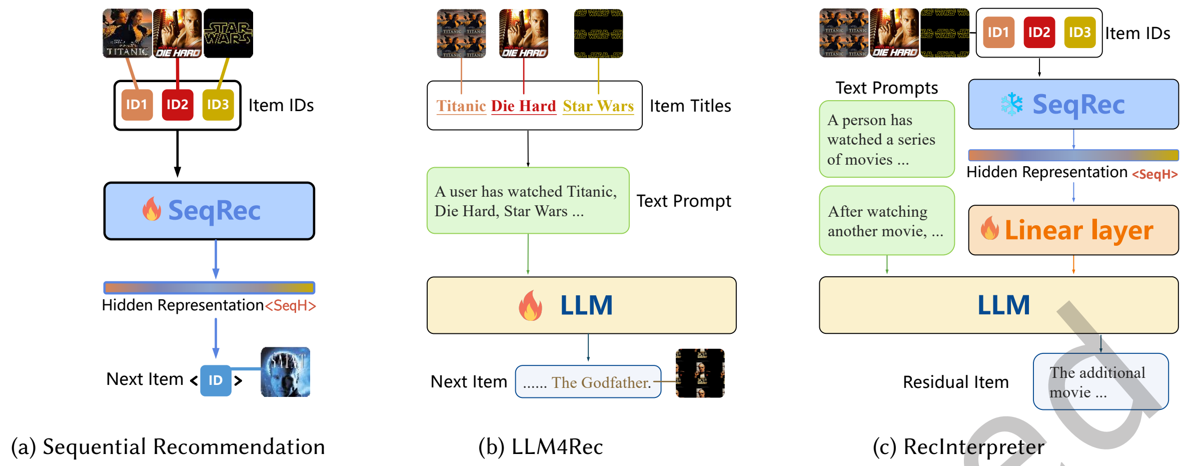
Item IDs
ID1
ID2
ID3
SeqRec
Hidden Representation<SeqH>
Next Item < ID >
(a) Sequential Recommendation
Titanic Die Hard Star Wars
Item Titles
A user has watched Titanic, Die Hard, Star Wars ...
Text Prompt
LLM
Next Item
...... The Godfather.
(b) LLM4Rec
Text Prompts
A person has watched a series of movies ...
After watching another movie, ...
Linear layer
Residual Item
The additional movie ...
(c) RecInterpreter

Fig. 1. Illustration of conventional sequential recommendation, LLM4Rec and RecInterpreter. Specifically, conventional sequential recommenders assign discrete IDs to items, learn sequence representations with sequences of item embeddings, and predict the next item matches user interest. Recent approaches leveraging LLMs for recommendation (LLM4Rec) mostly integrate the text description of user interaction sequence in the text prompt design and tuning the LLM for recommendation. In contrast to these approaches, our RecInterpreter first interprets hidden representations of behavioral patterns of conventional sequential recommenders with LLMs, then accomplishes recommendation task with the explicitly interpretation of behavior patterns. Flame 🔥 denotes tunable modules, while snowflake ❄ indicates frozen modules.

recommenders [5, 23, 31, 58, 73], we can summarize a typical pipeline: 1) assign discrete IDs to items and initialize learnable vectors (*aka.* item embeddings) to represent items, 2) hidden representations (*aka.* sequence embeddings) are learned from sequences of item embeddings to capture user behavioral patterns, and 3) predict the next items that users are likely to consume based on the hidden representations, as Figure 1 (*left* subfigure) shows. Such representations, derived from the ID-modeling paradigm, are able to encode the behavioral patterns of users, thereby significantly enhancing next-item prediction in sequential recommendation.

With the meteoric rise of Large Language Models (LLMs) (*e.g.,* GPT4 [48], LLaMA [60]), aligning diverse modalities with text can empower LLM to understand and reason about other modalities [2, 14, 19, 20, 24, 45, 81]. Central to such an alignment is transforming the hidden representations from the modality-specific encoders (*e.g.,* images encoded by ViT [13] or Stable Diffusion [55]) into the text token embeddings of an LLM [19, 81]. This allows for the LLM to reason over the input modality and generate the textual responses correspondingly. Although multi-modal comprehension is becoming a focal point of LLM, the capability to interpret hidden representations from sequential recommenders remains mostly unexplored. This is largely due to the current LLM-for-Recommendation (LLM4Rec) studies [1, 4, 15, 16, 18, 36, 38, 67, 74] predominantly focus on reformulating user-item interactions as text prompts for LLM-based recommendation or reranking, as shown in Figure 1 (*middle* subfigure). This approach, however, prevents LLMs from accessing or decoding the hidden representations within recommender models, thus failing to fully realize the potential of LLMs in recommendation systems.

Naturally, an compelling research question arises:

Can LLM Understand and Utilize Representations from Conventional Recommenders?

To answer this, we propose a simple framework, **RecInterpreter**, which examines the capacity of open-source LLM to decipher the representation space of sequential recommenders, and enhances recommendation performance accordingly. In terms of sequential recommenders, we harness the representative models trained solely on item ID sequences, including GRU4Rec [23], Caser [58], and SASRec [31]. Having the LLM and recommender frozen, the solution to bridge their gap is the alignment training [2, 47, 81] with the paired multimodal data (*i.e.,* representations of item ID sequences and text narrations). Following the leading alignment strategies [14, 47, 81], RecInterpreter employs two key designs: 1) train a lightweight projector to map the recommendation representations into the token embedding space of the LLM, and 2) inject these recommendation-specific tokens into a text prompt and ask the LLM for a textual elucidation. Specifically, as a bridge, the adapter fuses the spaces of the LLM and recommender into a joint token embedding space, wherein tokens represent both text and user behavioral patterns. Moreover, we simply set it as a MLP-based projection layer to train, where the model parameters of the recommenders and LLM are frozen. This lightweight design not only reaches convergence faster than training from scratch, but also inherits the reasoning capabilities of the LLM.

Having the recommendation-specific tokens, we first propose a sequence-recovery prompting task, which tries to explicitly recover the whole item sequence from the projected representations with text narrations. Here is an example of the prompt in the movie recommendation scenario:

Sequence-Recovery Prompt Example

"A user has watched a series of movies, which can be represented as <SeqH>. What movies has the user watched?"

where <SeqH> is the behavioral pattern representation encoded by a sequential recommender (*e.g.,* the hidden representation in Figure 1). While we empirically show that LLM could understand some interactions from the hidden representation, it is hard to recover all the interactions, since the hidden representation is highly compressed. To this end, we carefully craft a sequence-residual prompt tailored for sequential recommenders. This prompt is designed to guide LLM in identifying the residual item by comparing the representations before and after the sequence incorporates said residual.

Sequence-Residual Prompt Example

"A user has watched a series of movies, which can be represented as <SeqH1>. After watching another movie, the watching history can be represented as <SeqH2>. What is the additional movie the user watched?"

where <SeqH1> and <SeqH2> are the hidden representations before and after the sequence integrates with the residual.

Surprisingly, our empirical evaluations show that LLM exhibits a significant aptitude for deciphering the representations from sequential recommenders, especially following our instructions. Consequently, we may safely reach the conclusion that LLMs could be inspired to understand the representation space of sequential recommenders, which inherently encapsulate rich patterns of user behaviors. Moreover, since the linear projection is the only tunable component, it is affordable for online service providers to interpret their own recommenders with LLMs, which is flexible for them to investigate further how to utilize LLMs in their platforms.

To empower recommendation with RecInterpreter, we further tune LLM with LoRA [28] for recommendation, while keeping the sequential recommenders and the projection layer frozen. This strategy allows us to adapt the LLM's behavior specifically for recommendation tasks without disturbing the carefully learned representations from the sequential recommenders or the projection mechanisms. Note that we involve no textual narration of interactions (such as title sequence used in [37, 79]), and only use the projected behavioral pattern representation to describe user history. This streamlined approach represents a significant innovation in how we leverage LLMs for recommendation tasks. By eliminating the need for explicit textual descriptions, we not only reduce the computational complexity but also create a more direct and efficient path between user behavior patterns and recommendations. The projected representations serve as a sufficient and compact form of user history, containing the necessary information for generating accurate recommendations. Superior recommendation performance highlights the effectiveness of explicitly aligning sequential recommenders with LLM under the RecInterpreter framework, demonstrating that our approach not only simplifies the recommendation process but also enhances its effectiveness. This success validates our hypothesis that direct representation-based communication between recommender systems and LLMs can be more effective than approaches relying on intermediate textual descriptions.

Our contributions can be summarized as follows:

- We demonstrate that LLMs could explicitly interpret the behavioral patterns encoded in the sequence representations of pretrained sequential recommenders with our proposed sequence-recovery prompting and sequence-residual prompting.
- We propose RecInterpreter, which could not only understand the hidden representation of sequential recommenders, but also boost the recommendation performance of LLM-based recommenders with the explicitly understanding of behavior patterns.
- To examine the effectiveness of RecInterpreter, we conduct extensive experiments on four real-world datasets to showcase its capability of interpretering the hidden space of sequential recommenders and its potential to improve the performance of LLM-based recommenders.
- To show the impact of different components in RecInterpreter, we conduct plenty of ablation studies, including the efficiency analysis, projector layers, zero-shot performance and alignment strategies, which could properly verify the necessity of different components.

2 RELATED WORK

This section reviews the work on large language models (LLMs) and large multimodal models (LMMs), and then discusses the work on sequential recommendation, especially the integration of LLM.

2.1 Large Language Models and Large Multimodal Models

Over recent years, the field of language modeling has undergone intensive research focused on understanding and generating human language, leading to significant breakthroughs in Language Models (LMs) [11, 52]. The research community has subsequently explored the impact of scale by pushing boundaries in both model size and training data volume - now reaching unprecedented scales of billions of parameters trained on trillion-token datasets. This evolution has given rise to Large Language Models (LLMs), with examples such as GPT4 [48] and LLaMA [60], which have not only achieved superior performance but also exhibited unexpected capabilities including logical reasoning and the ability to follow instructions. The success of LLMs has extended to specialized fields, with domain-specific models emerging in sectors such as finance [69] and healthcare [56], combining field-specific expertise with the broader knowledge base of general-purpose LLMs. These developments have motivated our investigation into the applicability of LLMs within the recommendation domain.

Meanwhile, different modalities (including vision, video, audio and *etc.*), have been evolving their models rapidly to better accommodate different tasks [13, 32, 55]. With the rapid advancements of LLMs, it is promising to develop large multimodal models by integrating the archetecture or techniques in LLMs. More recently, researchers find that models of different modalities can be unified with LLM by making the hidden representations perceivable for LLM, leading to a promising direction, multimodal language models [2, 19, 45, 68, 81]. Along this research line, the pioneer work, Flamingao [2] demonstrates that the vision encoder NFNet [6] could be understood by LLM through inserting tunable gated cross-attention dense blocks among the layers of LLM, which has been proven effective in GPT-4Vision [49]. MiniGPT4 [81] further shows that a single linear layer is enough to make

LLaMA [60] interpret hidden representations encoded by ViT [13]. Similarly, TANGO [19] suggests that the audio backbone model HiFiGAN [32] can be unified by LLM. Besides, Video-ChatGPT [45] and VideoChat [34] imply that video encoders are also perceivable for LLM.

2.2 Conventional Sequential Recommendation

Sequential recommendation aims at inferring users' preferences based on their interaction sequences. Previous work has explored encoding the interaction sequences with different model architectures. RNN-based approaches like GRU4Rec [23] leverage recurrent structures to capture temporal dependencies, with gating mechanisms helping to address the vanishing gradient problem while modeling long-term dependencies. CNN-based methods such as Caser [58] employ convolutional operations to extract local and hierarchical patterns from interaction sequences, effectively capturing both short-term preferences and transition patterns. Transformer-based models like SASRec [31] utilize self-attention mechanisms to model complex item relationships and long-range dependencies without sequential compression. Recent advances have introduced sophisticated auxiliary learning objectives to enhance recommendation quality [51]. Causal inference frameworks [75, 78] attempt to disentangle confounding factors and identify true user preferences, while contrastive learning approaches through data augmentation [63, 72] help learn more robust and generalizable representations. Robust learning techniques [73] aim to maintain model performance under noisy conditions.

Analysis of prominent sequential recommendation systems [5, 23, 31, 58, 65] reveals a common methodological framework built on three key components. First, items are assigned unique discrete IDs and represented through learnable vectors (*aka.* item embeddings), serving as the fundamental building blocks that capture item characteristics in a continuous vector space. Second, the interaction sequence of item embeddings undergoes architectural-specific processing to generate hidden representations (*aka.* sequence embeddings) that capture complex temporal patterns, user preferences, and item transitions within the interaction history. Finally, these hidden representations are utilized to forecast users' future item interactions, typically through similarity computation or classification layers. This ID-based modeling approach has proven remarkably effective in encoding user behavioral patterns, leading to substantial improvements in next-item prediction accuracy within sequential recommendation. The behavioral patterns encoded in the sequence embeddings have emerged as the cornerstone of conventional sequential recommenders, serving as compressed yet informative representations of user preferences and interaction histories. Therefore, understanding and interpreting these patterns is crucial for both improving model performance and providing transparent recommendations.

2.3 LLM-based Sequential Recommendation

Recent advances in Large Language Models (LLMs) have sparked significant interest in their application to sequential recommendation. This integration has manifested in several distinct research directions, each offering unique approaches to leveraging LLM capabilities. One stream of research focuses on enhancing conventional recommendation systems using LLMs as auxiliary components. These studies primarily follow two strategies: using LLMs to extract relevant features [7, 17, 57, 64, 66, 71, 74], or generating semantic representations [25, 40, 43, 53, 54, 80] for both users and items. While this integration successfully introduces rich contextual knowledge into conventional recommendation frameworks, it falls short of fully utilizing LLMs' sophisticated reasoning capabilities. Another research direction explores the direct application of LLMs through in-context learning to evaluate their recommendation performance or enhance traditional recommendations [22, 27, 42, 76]. Recognizing LLMs' potential limitations in recommendation-specific knowledge during pre-training, several studies have proposed specialized tuning techniques to enhance their recommendation capabilities [4, 9, 12, 29, 35, 39, 39, 46, 77].

A promising approach has emerged in the integration of pre-trained recommenders for tuning LLMs, demonstrating effectiveness in enhancing LLM performance for recommendation tasks [36, 37, 79]. However, these methods typically maintain a barrier between LLMs and the hidden representations from frozen recommender models. This limitation leaves largely unexplored the potential of enabling LLMs to interpret and leverage these hidden representations for improved recommendations. While recent work such as ELM [59] proposes comprehensive understanding of item embeddings with LLM, RecInterpreter takes a different approach by focusing on improving LLM-based recommendation through better utilization of sequence embeddings from conventional recommenders. This distinction is significant as sequence embeddings encode complex user sequential behavior patterns, offering richer information than static item embeddings alone. The challenge of inspiring LLMs to interpret these hidden representations from sequential recommenders and improve recommendations accordingly remains largely unexplored, presenting an important research opportunity in the field.

2.4 Parameter-efficient Finetuning

Fine-tuning all parameters of pre-trained LLMs on task-specific data has traditionally been the primary approach for adapting LLMs to new tasks and domains. This conventional method involves updating the entire parameter space of the model during the adaptation process, which allows for comprehensive model optimization. However, this full fine-tuning approach is extremely time and resource intensive, often requiring substantial computational infrastructure and significant energy consumption, making it impractical for many real-world applications.

Therefore, plenty of parameter-efficient methods have been proposed to achieve comparable performance to full fine-tuning while only training a small subset of parameters [10, 28, 44]. These methods aim to strike a balance between adaptation effectiveness and computational efficiency by strategically updating only the most critical parameters. Among such methods, Low-Rank Adaptation (LoRA) has gained significant attention [28]. LoRA injects trainable low-rank decomposition matrices into each layer of the LLM's Transformer architecture. This technique is based on the observation that the parameter updates during fine-tuning often lie in a relatively low-dimensional subspace, making it possible to capture these updates efficiently through low-rank approximations. With far fewer trainable parameters than full fine-tuning, LoRA enables much faster training and smaller downstream model sizes, while still achieving strong performance.

More precisely, given a prompt-response sample $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ and $\mathbf{Y}$ are the input prompt and target response respectively, the autoregressive language modeling objective is:

$$\max \sum_{i=1}^{N} \log P_{_0+()} \left(\mathbf{Y}^i | \mathbf{X}, \mathbf{Y}^{[1:i-1]} \right), \tag{1}$$

where N is the number of tokens in the target response $\mathbf{Y}$, and $\mathbf{Y}^i$ represents the i^{th} token in $\mathbf{Y}$. The probability $P_{_0+()}$ is computed using both the frozen LLM parameters $_0$ and the low-rank adaptation matrices ().

The LoRA approach decomposes the weight updates into low-rank approximations. Specifically, for each weight matrix $W \in \mathbb{R}^{d \times k}$ in the original model, the update is expressed as:

$$W = AB, \tag{2}$$

where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$ are the low-rank decomposition matrices, and $r \ll \min(d, k)$ is the rank. The parameter set consists of these trainable LoRA matrices A and B for each weight matrix being adapted, while keeping the vast majority of the original LLM parameters $_0$ frozen.

The objective maximizes the log probability of generating each token in the target response, conditioned on both the input prompt $\mathbf{X}$ and all previously generated tokens $\mathbf{Y}^{[1:i-1]}$.

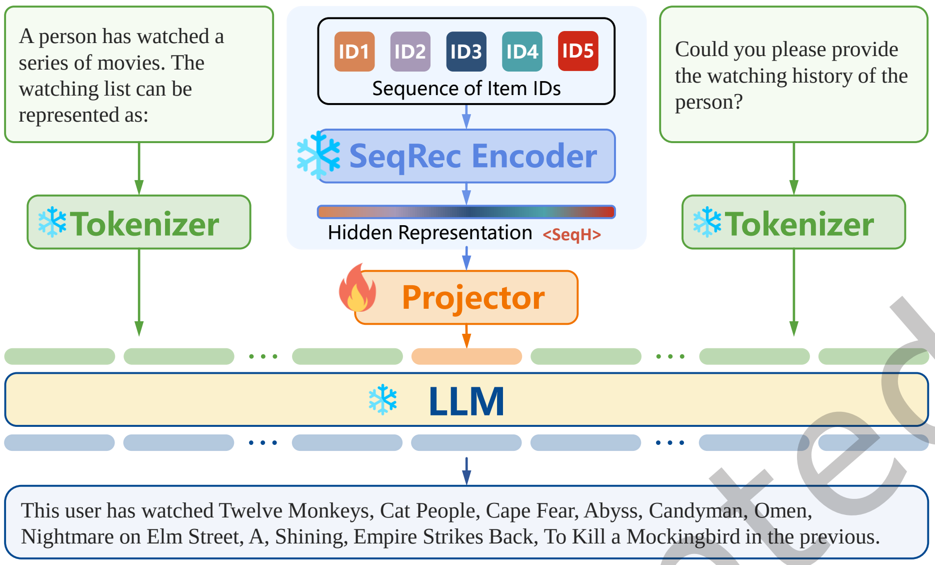
A person has watched a series of movies. The watching list can be represented as:
ID1 ID2 ID3 ID4 ID5
Sequence of Item IDs
SeqRec Encoder
Hidden Representation <SeqH>
Could you please provide the watching history of the person?
Tokenizer
Tokenizer
Projector
LLM
This user has watched Twelve Monkeys, Cat People, Cape Fear, Abyss, Candyman, Omen, Nightmare on Elm Street, A, Shining, Empire Strikes Back, To Kill a Mockingbird in the previous.

Fig. 2. Illustration of the sequence recovery framework. We provide the textual prompts and the hidden representation of the interaction sequence projected by the projection layer, targeting at inspiring LLM to recover the interactions with a textual response. Flame denotes tunable modules, while snowflake indicates frozen modules.

3 RECINTERPRETER FRAMEWORK

In this section, we outline our RecInterpreter framework to harness the capabilities of LLMs for comprehending conventional sequential recommenders. We first introduce a sequence-recovery prompting task, aiming to empower LLM to explicitly reconstruct the items in an interaction sequence with textual narrations based solely on its hidden representation. This task serves as a crucial step in transforming the ID-based sequence representations typically used in recommender systems into human-readable, interpretable descriptions that capture the underlying patterns and user preferences encoded in the sequence. Taking a step further, we propose a novel sequence-residual task, which guides LLM to pinpoint the residual item by contrasting the hidden representations before and after integrating this residual into the existing sequence. This approach enables the model to identify and understand the incremental changes in user preferences and behavioral patterns when new interactions are added to the sequence. By analyzing these representational differences, the LLM can better capture user behavior and the impact of individual items on the overall sequence representation. Based on the sequence-recovery and sequence-residual prompting, we propose a novel scheme to tune LLM for recommendation by leveraging the projected hidden representations, eliminating the need for explicit textual descriptions of user interactions. Finally, we discuss the difference between RecInterpreter and some recent studies.

3.1 Sequence-Recovery Prompting

To validate LLM's capability in understanding sequential recommenders, we draw inspiration from prior multi-modal alignment studies [2, 14, 34, 81] and propose a sequence recovery task. This task is designed to evaluate whether LLMs can accurately reconstruct and describe interaction sequences using natural language, working only with the hidden representations learned by conventional sequential recommenders. As illustrated in Figure 2, the process consists of several key components and steps:

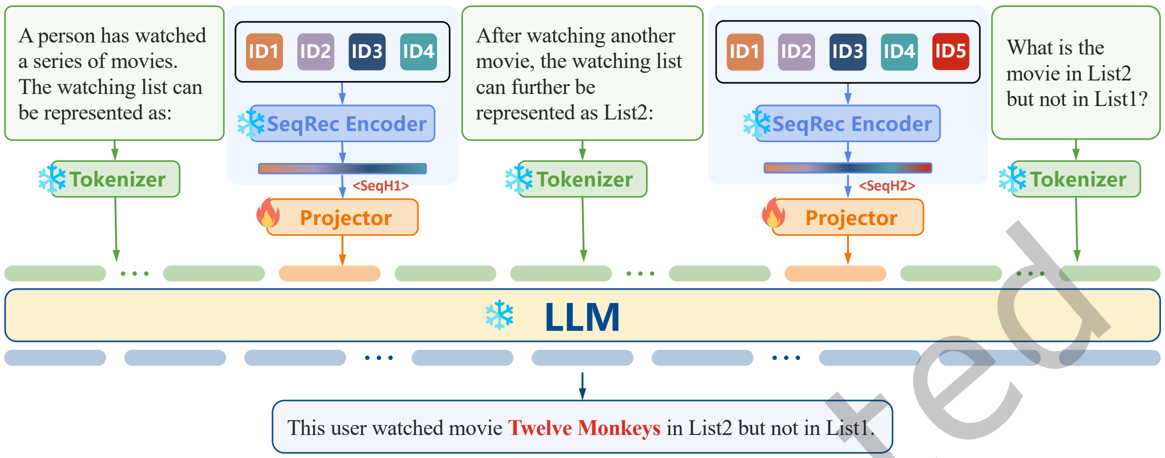
A person has watched a series of movies. The watching list can be represented as:
ID1 ID2 ID3 ID4
SeqRec Encoder
<SeqH1>
Projector
After watching another movie, the watching list can further be represented as List2:
ID1 ID2 ID3 ID4 ID5
SeqRec Encoder
<SeqH2>
Projector
What is the movie in List2 but not in List1?
Tokenizer
Tokenizer
Tokenizer
LLM
This user watched movie Twelve Monkeys in List2 but not in List1.

Fig. 3. Illustration of the sequence residual framework. We provide the task-specific textual prompts and the hidden representations before and after the sequence incorporates a residual item. The two hidden representations are projected by a shared projection Layer. Flame 🔥 denotes tunable modules, while snowflake ❄ indicates frozen modules.

Sequence Encoding via Sequential Recommenders. For an interaction sequence $\mathbf{s} = [s_1, s_2, \ldots, s_m]$ that involves the consumed items, we employ a well-trained sequential recommender, such as SASRec, to derive the hidden representation of the sequence. Formally, the interaction sequence **s** is first fed into the sequence encoder, and we can acquire the hidden representation through:

$$\mathbf{h_s} = \mathbf{Seq\text{-}Enc}(\mathrm{E_s}), \tag{3}$$

where $\mathbf{Seq\text{-}Enc}(\cdot)$ is the sequence encoder of conventional sequential recommender, and $\mathbf{h_s} \in \mathbb{R}^d$ is the d-dimentional representation of sequence s (*e.g.,* d is set as 64 or 256 in SASRec).

Representation Adaptation via Lightweight Projector. We train a lightweight projector to project the hidden representation $\mathbf{h_s}$ into the text token embedding space of LLaMA. Here we implement the projector as an MLP projection layer, whose design ensures that the input dimension aligns with d, while the output dimension matches LLM's token embedding size (*i.e.,* 4096 for LLaMA). Thus the hidden representation is transformed as:

$$\tilde{\mathbf{h}}_\mathbf{s} = \mathbf{Proj}_\theta(\mathbf{h_s}). \tag{4}$$

In this way, the projector serves as a bridge, integrating the spaces of LLM and the recommender system. This leads to a unified token embedding space, where tokens can signify either textual content or user interactions. The deeper exploration of such projectors, such as Q-former [33], is an avenue we plan to explore in future work.

Prompt Design for the Projector Training. Here we design the sequence-recovery prompts, which are composed of text tokens interleaved with the projected sequence representation $\tilde{\mathbf{h}}_\mathbf{s}$. Here is an example of the sequence-recovery prompt in the movie recommendation scenario:

Sequence-Recovery Prompt	
Input Prompt	A person has watched a series of movies. The watching list can be represented as: <SeqH>. Describe this watching history of the person in detail.
Target Response	This user has watched Twelve Monkeys, Cat People, Cape Fear, Abyss, Candyman, Omen, Nightmare on Elm Street, Shining, Empire Strikes Back, To Kill a Mockingbird in the previous.

where <SeqH> is the projected hidden representation (*i.e.,* $\tilde{\mathbf{h}}_\mathbf{s}$).

It is worth noting that the prompt involves two key components: 1) the input prompt, which contains the projected hidden representation of the sequence $\tilde{\mathbf{h}}_\mathbf{s}$; and 2) the target response, which offers the detailed textual narration of $\tilde{\mathbf{h}}_\mathbf{s}$. Within the autoregressive framework of LLM, we calculate the training objective by regressing the target prompt $\mathbf{Y}$ based on the condition of the input prompt $\mathbf{X}$ [60]:

$$\max_{\theta} \sum_{i=1}^{N} \log P(\mathrm{Y}^i | \mathrm{X}, \mathrm{Y}^{[1:i-1]}), \tag{5}$$

where θ denotes the parameters of the projection layer $\mathbf{Proj}_\theta(\cdot)$, N is the number of tokens in the target prompt and $\mathbf{Y}^i$ is the i-th token in $\mathbf{Y}$.

During the training, we provide both the input prompt and the target response with the objective of learning to generate descriptions for the projected sequence embedding $\tilde{\mathbf{h}}_\mathbf{s}$. During the inference, we only provide the input prompt containing the projected sequence embedding $\tilde{\mathbf{h}}_\mathbf{s}$, acquiring the output text as LLM's understanding of $\tilde{\mathbf{h}}_\mathbf{s}$.

3.2 Sequence-Residual Prompting

It is challenging for LLM to understand all items from a simple hidden representation of the interaction sequence, since the datasets of recommendation are usually very sparse. Although we empirically show that LLaMA can understand the interactions to a large extent under the sequence-recovery framework, we would also like to refine the framework to encourage LLaMA to understand sequential recommender more delicately. Drawing inspiration from Flamingo [2], which suggests that LLMs could better process images if the hidden representations of two similar images are provided at the same time with their differences, we propose to inspire LLaMA to understand sequential recommenders by identifying the residual item based on hidden representations before and after a sequence integrates the residual, as illustrated in Figure 3. Then we elaborate on the sequence-residual prompting step by step:

Sequence Encoding via Sequential recommenders. Given an interaction sequence $\mathbf{s} = [s_1, s_2, \ldots, s_m]$, we could design a circumstance, that a user has interacted with $[s_1, s_2, \ldots, s_{m-1}]$ and then interacts with a residual item s_m. The sequential recommender could encode $\mathbf{s}^1 = [s_1, s_2, \ldots, s_{m-1}]$ and $\mathbf{s}^2 = [s_1, s_2, \ldots, s_m]$ as $\mathbf{h}_s^1$ and $\mathbf{h}_s^2$ respectively:

$$\mathbf{h}_{\mathbf{s}^1} = \mathbf{Seq\text{-}Enc}(\mathbf{E}^1) \quad \mathbf{and} \quad \mathbf{h}_{\mathbf{s}^2} = \mathbf{Seq\text{-}Enc}(\mathbf{E}^2), \tag{6}$$

where $\mathbf{E}^1$ and $\mathbf{E}^2$ are the vactorized sequence of $\mathbf{s}^1$ and $\mathbf{s}^2$.

Representation Adaptaion via Lightweight Adapter. We also employ a linear projection layer as the lightweight adapter, which could project $\mathbf{h}_{\mathbf{s}^1}$ and $\mathbf{h}_{\mathbf{s}^2}$ to be $\tilde{\mathbf{h}}_{\mathbf{s}^1}$ and $\tilde{\mathbf{h}}_{\mathbf{s}^2}$:

$$\tilde{\mathbf{h}}_{\mathbf{s}^1} = \mathbf{Proj}_\theta(\mathbf{h}_{\mathbf{s}^1}) \quad \mathbf{and} \quad \tilde{\mathbf{h}}_{\mathbf{s}^2} = \mathbf{Proj}_\theta(\mathbf{h}_{\mathbf{s}^2}), \tag{7}$$

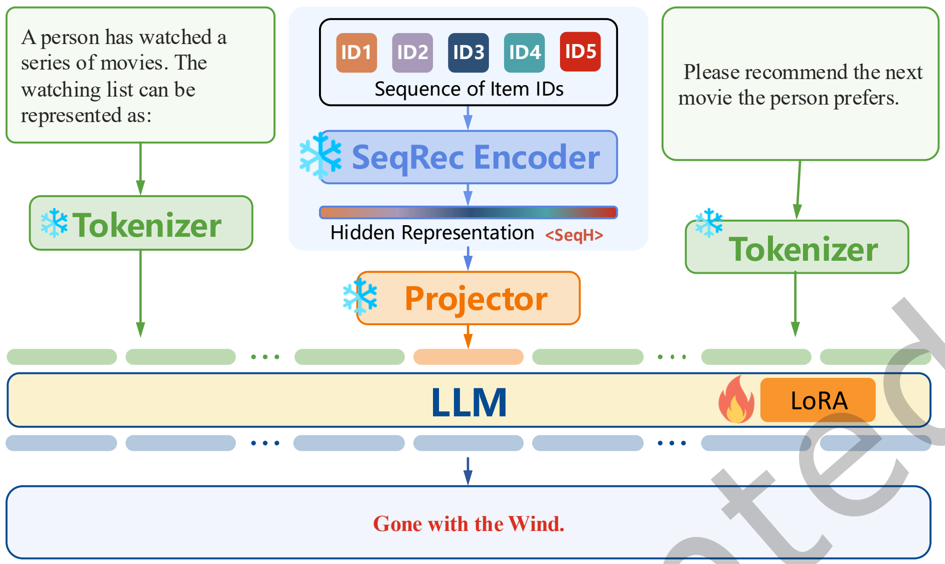
A person has watched a series of movies. The watching list can be represented as:
Tokenizer
ID1 ID2 ID3 ID4 ID5
Sequence of Item IDs
SeqRec Encoder
Hidden Representation <SeqH>
Projector
Please recommend the next movie the person prefers.
Tokenizer
LLM
LoRA
Gone with the Wind.

Fig. 4. Illustration of leveraging RecInterpreter for recommendation. We provide the textual prompts and the hidden representation of the interaction sequence after projection, targeting at letting LLM predict the correct next item from candidates in the prompt. Note that we only employ the projected hidden representation <SeqH> to denote the user interaction history in the prompt, instead of integrating the text description of user interactions. Flame 🔥 denotes tunable modules, while snowflake ❄ indicates frozen modules.

where the parameters of linear layer are shared by $\mathbf{h}_{s^1}$ and $\mathbf{h}_{s^2}$.

Prompt Design for the Adapter Training. Here we design more delicate sequence-residual prompts, which inspire LLaMA to identify the residual item s_m by comparing $\tilde{\mathbf{h}}_{s^1}$ and $\tilde{\mathbf{h}}_{s^2}$. Here is an example of the sequence-residual prompt in the movie recommendation scenario:

Sequence-Residual Prompt	
Input Prompt	A person has watched a series of movies. The watching list can be represented as List1: <SeqH1>. After watching another movie, the watching list can further be represented as List2: <SeqH2>. What is the movie in List2 but not in List1?
Target Response	This user watched movie Twelve Monkeys in List2 but not in List1.

where <SeqH1> and <SeqH2> would be replaced with $\tilde{\mathbf{h}}_{s^1}$ and $\tilde{\mathbf{h}}_{s^2}$, and Twelve Monkeys is the residual item in the example.

Similar to the sequence-recovery prompting, we provide both the input prompt and target response during the training phase, and only the input prompt during the inference phase.

3.3 RecInterpreter for Recommendation

After training the projector, LLM is able to explicitly interpret the hidden representation of behavioral patterns encoded by sequential recommenders. This capability enables the model to transform complex numerical ID embeddings into comprehensible natural language descriptions, providing insights into how the recommender system understands and represents user behavior patterns. Next, we demonstrate how the explicit projection could benefit the recommendation task, with the key idea of leveraging the projected representation to describe user behaviors and tuning LLM for recommendation. The framework of leveraging RecInterpreter for Recommendation is illustrated in Figure 4.

Prompt Design for Recommendation. To empower LLM for recommendation after explicitly deciphering sequential recommenders, we involve the projected representation $\tilde{\mathbf{h}}_\mathbf{s}$ in the prompt design for recommendation. We employ the all-ranking setting adopted from BIGRec [3], which first performs instruction-tuning on the LLM to restrict its output from language space to recommendation space. Then, it grounds the LLM's outputs to actual items by calculating the L2 distance between item embeddings and the embedding of the LLM's generated output.

Here is an example of the prompt in the movie recommendation scenario:

Recommendation Prompt	
Input Prompt	A person has watched a series of movies. The watching list can be represented as: <SeqH>. Please suggest the next movie this person is likely to watch.
Target Response	Gone with the Wind

where <SeqH> is the projected hidden representation (*i.e.,* $\tilde{\mathbf{h}}_\mathbf{s}$), which can be acquired through either the sequence-recovery prompting or the sequence-residual prompting.

It worth noting that we only employ the projected hidden representation <SeqH> to denote the user interaction history in the prompt, instead of integrating the text description of user interactions like some other studies do [37, 79]. This design choice represents a significant departure from conventional approaches and offers several advantages. The projected representation serves as a compact yet informative encoding of user behavior patterns, eliminating the need for verbose textual descriptions while maintaining the essential information needed for recommendation tasks. The reason is that the projected hidden representation <SeqH> is supposed to contain the behavioral patterns after the sequence-recovery prompting or sequence-residual prompting, which is informative enough to represent user interaction for recommendation task. This approach is particularly effective because the sequence-recovery and sequence-residual prompting processes have already trained the LLM to understand and interpret these representations in a meaningful way. By leveraging these learned representations directly, we can achieve more efficient and streamlined recommendation processing while maintaining the model's ability to capture complex user behaviors and preferences.

Then we adopt LoRA [28] to tune LLM for recommendation, the objective can be formulated as:

$$\max \sum_{i=1}^{N} \log P_{0+()} \left(\mathbf{Y}^i | \mathbf{X}, \mathbf{Y}^{[1:i-1]} \right), \tag{8}$$

where $_0$ and are parameters of LLM and parameters of low-rank matrics decomposition introduced by LoRA, respectively. Note that only the parameters introduced by LoRA are tunable, and the sequential recommender, the projection layer, and the LLM are frozen in the training process for recommendation task.

3.4 Discussion

It is worth mentioning that several recent studies have proposed to integrate pre-trained recommenders into LLM for recommendation, which has shown effectiveness in enhancing the performance of LLM in recommendation tasks [36, 37, 79]. These approaches represent progress in combining the strengths of traditional recommender systems with the powerful language understanding capabilities of LLMs. Nevertheless, these studies directly train the projection layer with the objective of optimizing recommendation tasks. Besides, their prompt designs typically contain both the hidden representation and textual information (*e.g.,* title sequence) of historical interactions to describe user history. This dual-input approach, while comprehensive, may introduce redundancy and computational overhead into the recommendation process. More importantly, these methods typically leverages item- or user-level alignment by directly projecting item or user embeddings into the token embedding space of LLMs. Although these methods have achieved certain effectiveness, they have overlooked the information encoded within sequential encoders that process user behavior sequences. This information is crucial for understanding user behaviors and preferences in sequential recommendation.

In contrast, RecInterpreter explicitly inspires LLM to interpret hidden behavioral pattern representation with textual narrations, as detailed in Section 3.1, which enables LLM to understand the hidden representations in the first place. This step is crucial as it establishes a strong semantic connection between the sequence embeddings of numerical ID representations and their meaning in natural language, creating a more robust basis for subsequent recommendation tasks. In contrast to item- or user-level alignment, we adopt the sequence-level alignment, where the sequence embeddings are generated by pretrained sequential recommenders. This step is crucial to fully leverage the information of user behavior patterns learned by the sequential recommender. Our approach ensures that the LLM develops a deep understanding of the behavioral patterns before being tasked with making recommendations. More importantly, in the recommendation prompt of RecInterpreter, we only utilize the projected hidden representation $\tilde{\mathbf{h}}_\mathbf{s}$ to describe user historical interactions, involving no textual information. This streamlined approach demonstrates that once the LLM has been properly trained to understand these representations, additional textual descriptions become unnecessary. The projected representations alone contain sufficient information for generating accurate recommendations, leading to a more efficient and elegant solution. These differences set RecInterpreter distinctly apart from prior studies, offering a novel paradigm that prioritizes deep understanding of behavioral patterns while maintaining recommendation efficiency.

4 EXPERIMENT

In this section, we conduct extensive experiments to evaluate and validate our proposed RecInterpreter framework. The primary objective is to demonstrate how RecInterpreter explicitly enables LLMs to understand sequential recommenders through the interpretation of hidden representations. Furthermore, we investigate how this enhanced understanding can be leveraged to improve overall recommendation performance through our framework. To ensure a comprehensive evaluation, we employ three real-world datasets that represent diverse recommendation scenarios and user behaviors. These datasets provide varied interaction patterns and item characteristics, allowing us to assess the generalizability of our approach across different domains. We compare RecInterpreter against several competitive baselines, including both traditional sequential recommenders and state-of-the-art methods, to establish its effectiveness. For clarity, we present our experimental analysis by addressing the following research questions:

- **RQ1:** How could LLM interpret the hidden space of conventional recommenders with the proposed sequence-recovery prompting in RecInterpreter?
- **RQ2:** How could LLM interpret the hidden space of conventional recommenders with the proposed sequence-residual prompting in RecInterpreter?

Table 1. Statistics of datasets.

Dataset	Movie	Steam	Book	CD
#sequences	943	11,938	6,031	123,876
#items	1,682	3,581	4,500	89,370
#interactions	100,000	274,726	220,100	1,552,764

- **RQ3:** How does the recommendation performance of the proposed leveraging RecInterpreter for recommendation compared to traditional and LLM-based recommenders?

Through addressing these research questions, we aim to provide a thorough understanding of RecInterpreter's capabilities in bridging the gap between sequential recommenders and language models, while demonstrating its practical value in enhancing recommendation performance.

4.1 Experimental Setings

4.1.1 Datasets. We use four datasets from real-world recommendation scenarios: Movie, Steam, Book and CD. The statistics of datasets are illustrated in Table 1.

- **Movie**[1] is a well-known dataset for movie recommendation, containing users' rating history. We preserve titles as the textual descriptions of movies. A notable characteristic of the MovieLens dataset is its high-quality and reliable rating data, making it a well-adopted benchmark in recommendation system research.
- **Steam** [31] dataset contains user reviews of video games on the Steam Store. The Steam dataset is unique in that it reflects user interaction behaviors in the gaming domain. Game titles typically encode crucial information about game genres and themes, providing valuable semantic cues for recommendation systems.
- **Book** is collected from a social book cataloging website[2], where users can search, rate, and provide reviews of various books. The platform covers a diverse range of book categories, including fiction, biography, science fiction, mystery, and more. These characteristics make the Book dataset particularly suitable for evaluating recommendation algorithms in the literary domain.
- **CD** comes from the well-known Amazon Review Benchmark (2023, latest version) [26]. We select the CDs_and_Vinyl (*abbr.* CD) dataset, which is of different category from the other datasets. From Table 1 we can see that the scale of CD dataset is much larger than other datasets, containing millions of interactions.

Since tuning the projection layer requires backpropagation from LLaMA, the training phase is more time-consuming than conventional recommenders, and the dataset size should not be too large. Based on this consideration, we select the MovieLens100K version. This scale ensures sufficient interaction information while keeping the training time manageable. Like previous studies, we maintain the standard practice of keeping users with at least 20 reviews to ensure reliable user behavior patterns. We first apply an interaction threshold by removing users who have fewer than 20 reviews, keeping consistent with the MovieLens preprocessing. Then, to further reduce the computational burden, we randomly sample one third of the users and one third of the games while preserving all interactions between these selected entities. This sampling strategy maintains the natural density and distribution patterns of the original dataset while reducing its scale. For the Book dataset collected from Goodreads, we implement similar density-based filtering by removing both users and books with fewer

[1]https://grouplens.org/datasets/movielens/
[2]https://www.goodreads.com/

than 20 interactions. This bilateral filtering ensures sufficient interaction data for both user and item representations, making the learned patterns more reliable and generalizable. For CD datasets, the pre-processing involves implementing a 5-core filtering mechanism, which removed users and items having fewer than 5 interactions.

For all datasets, we first sort all sequences in chronological order based on the timestamp of the last interaction in each sequence, and then split the data into training, validation, and testing data at the ratio of 8:1:1. This temporal splitting approach ensures that no future interactions leak into the training set, which is crucial for maintaining the integrity of the evaluation process as highlighted in literature [30]. When constructing instances for different tasks, we employ the following strategies:

- For the sequence-recovery prompting task: Given a user interaction sequence, we ensure the sequence length is at least 3 and at most 10 items (padding shorter sequences and truncating longer ones). We then apply a sliding window approach to create multiple training instances from each sequence. For example, from a sequence 'ABCDEFGHI', we can generate several training instances such as 'ABC', 'ABCD', 'ABCDE', and so on, with each subsequence and its corresponding text description forming a training point.
- For the sequence-residual prompting task: We create pairs of consecutive subsequences from each sequence, where each pair consists of a sequence and the same sequence with one additional item. For example, from a sequence 'ABCDEFGHI', we create training pairs such as ('AB', 'ABC'), ('ABC', 'ABCD'), and so on. The model is trained to identify the residual item by comparing the representations of these pairs.
- For the recommendation task: We adopt the standard next-item prediction format, where for each subsequence, the model predicts the next item. For instance, from a sequence 'ABCDEFGHI', we generate training examples like 'AB'$\longrightarrow$'C', 'ABC'$\longrightarrow$'D', and so on, with a maximum sequence length of 10.

This construction method ensures that our evaluation properly assesses the model's ability to understand sequential patterns while preventing data leakage between training and evaluation sets. Unlike approaches that simply designate the last few items of each user's sequence for evaluation, our strict temporal split maintains the chronological integrity of the recommendation task and better reflects real-world application scenarios.

4.1.2 Baselines.

- **GRU4Rec** [23] an RNN-based sequential recommender, which leverages GRU to encode users' interaction sequences. This pioneering approach introduced the application of recurrent neural networks to sequential recommendation, utilizing the GRU's ability to capture temporal dependencies.
- **Caser** [58] is a CNN-based sequential recommender. We apply one vertical filter and 16 horizontal filters with heights {2, 3, 4}. This model innovatively treats user-item interaction sequences as images and applies convolutional operations to capture local features. The vertical and horizontal convolutional filters are designed to capture point-level and union-level sequential patterns respectively.
- **SASRec** [31] a self-attention based sequential recommender that captures long-range dependencies in user-item interactions. By leveraging the transformer architecture, this model effectively models the relationships between items in a sequence, overcoming the distance limitations of traditional sequential models. The self-attention mechanism allows the model to dynamically focus on relevant historical interactions when making predictions.
- **ChatRec** is adapted from [16]. We preserve item titles and users' interaction sequences as users' profiles to accommodate for the datasets and settings in our experiment. Besides, we select GPT4 as the backbone LLM for ChatRec. This adaptation maintains the conversational nature of the original model while optimizing it for our specific experimental context, leveraging the advanced capabilities of GPT4 for more nuanced understanding of user preferences.

- **MoRec** [74] enhances the conventional recommenders by encoding the item's modality features (*e.g.,* text features). We adopt BERT as the text encoder and SASRec as the recommender backbone, consistent with the officially provided implementation. This multi-modal approach combines the semantic understanding capabilities of BERT with the sequential modeling power of SASRec, enabling richer item representations and more informed recommendations.
- **BIGRec** [3] constructs recommendation corpus by transferring interaction sequences into textual prompts, and tunes LLMs for recommendation with the corpus of specific domains. BIGRec grounds the LLM's outputs to actual items by calculating the L2 distance between item embeddings and the embedding of the LLM's generated output.
- **LLaRA** [37] integrates modality information learned by conventional recommenders into the tuning process of LLMs for recommendation. Besides, LLaRA develops a curriculum learning scheme to progressively warm up the training complexity. This approach combines multi-modal information with adaptive learning strategies, gradually increasing the difficulty of the training process to achieve better model convergence and performance.
- **CoLLM** [79] directly encodes collaborative information from pretrained traditional collaborative models, and then tunes a mapping module to align it with the LLM's input text token space for recommendations.
- **RecLora** [82] incorporates a personalized LoRA module for each user and a long-short modality retriever to retrieve history lengths for different modalities.

The compared methods we selected can be mainly categorized into the following two types:

- **Conventional recommenders** include **GRU4Rec**, **Caser** and **SASRec**. These are traditional recommenders that use historical interaction within specific domains to provide recommendations. These models have established strong baselines in the recommendation field through their specialized architectures for processing sequential user interactions.
- **LLM-based recommenders** can also be divided into several subcategories. We select **ChatRec** as the representative in-context learning based method, **BIGRec** and **RecLora** as the representative prompt tuning based method, **LLaRA** and **CoLLM** as the representative modality alignment based method. These methods represent different strategies for incorporating the powerful language understanding and generation capabilities of LLMs into recommendation. For all LLM-based recommenders and our proposed RecInterpreter, we all utilize LLaMA2-7B [61] as the backbone LLM for fair comparison.

4.1.3 Implementation details. We implement all approaches with Python 3.10, PyTorch 2.0.0, and transformers 4.32.0 in Nvidia A40 GPU. We preserve the last 10 interactions as the historical sequence. For sequences with less than 10 interactions, we would pad them to 10 with a padding token.

We first train the sequential recommenders (GRU4Rec [23], Caser [58], and SASRec [31]) on the training datasets. These models are selected as they represent different architectural paradigms in sequential recommendation: RNN-based, CNN-based, and Transformer-based approaches respectively. We use Adam optimizer due to its adaptive learning rate properties and superior performance in deep learning tasks. The learning rate is tuned as 0.001 and the batch size is set as 256. We adopt L2 regularization for all models to prevent overfitting, with the coefficient extensively searched in [1e-3, 1e-4, 1e-5, 1e-6, 1e-7]. The embedding size is searched in [16, 64, 256, 1024] to explore the trade-off between model capacity and computational efficiency. After training, the sequential recommenders are frozen to provide consistent feature extraction for subsequent stages.

We would utilize the frozen encoders in the pre-trained sequential recommenders to obtain the hidden representations of interaction sequences. Let L and D denote the length of the sequences and the dimension of the item embeddings respectively. For Caser, the size of the hidden representation is $1 \times (D + n_f \times s_f)$, where n_f and s_f are the number and size of convolutional kernels respectively [58], we directly employ an MLP layer to transfer the hidden representation to be the size of token embedding of LLaMA. For GRU4Rec and SASRec, they adopt sequence-to-sequence models (RNN or Transformer encoder) as sequence encoders, and the size of hidden representations is $L \times D$ [23, 31]. Therefore, we first acquire the linear combination of the hidden representations by employing a convolutional filter of size $L \times 1$ to acquire a $1 \times D$ representation, and then adopt the MLP projector similar to Caser.

In the training phase of sequence-recovery prompting, we employ a linear projection MLP layer as the projector. We defines two special tokens: '<Seq>', '</Seq>' to wrap the projected embeddings to indicate the start and end of interpretation. The training process uses AdamW optimizer with a carefully designed warmup schedule: starting at 0.0001 for the 1st epoch, linearly increasing to 0.0005 by the 5th epoch, then maintaining this rate. This schedule helps stabilize early training while allowing for effective optimization later. The L2 regularization coefficient is searched in [1e-4, 1e-5, 1e-6] to prevent overfitting while maintaining model expressiveness. We select LLaMA2-7B [61] as our base LLM, balancing model capacity with computational constraints. The maximum generated token length is capped at 50 - this limit was empirically determined to accommodate multiple items while preventing hallucination and redundant generation. The training times (2, 6, and 5 hours per epoch for Movie, Steam, and Book datasets respectively) reflect the varying complexity of each domain. Through empirical observation, 20 epochs typically achieve convergence while avoiding overfitting.

For the LoRA tuning phase, we employ AdamW optimizer with a comprehensive hyperparameter search: learning rates in [1e-4, 1e-5, 1e-6] and L2 coefficients in [1e-6, 1e-7, 1e-8]. The LoRA rank is set to 8, providing a good balance between adaptation capacity and parameter efficiency. This configuration allows for effective fine-tuning.

To ensure robust evaluation and statistical significance, all experiments are conducted 3 times with different random seeds, and we report the averaged results.

4.1.4 Metrics. The evaluation of RecInterpreter includes the performance of interpreting sequential recommenders and recommendation performance:

- To evaluate whether LLM could interpret the interacted items encoded in the hidden representations, we calculate the number of interacted items RecInterpreter can correctly generate with text narrations after the sequence-recovery prompting. Then we calculate the ratio of correct recovery among all interaction sequences for sequence-recovery task. For the sequence-residual task, we compute prediction precision of the residual item by comparing the model's prediction with the actual residual item. This metric directly measures how well the model can identify incremental changes in the sequence representation.
- To evaluate the recommendation performance, we adopt a all-ranking task following prior study [3]. Specifically, for each sequence, we rank all candidate items based on the model output. For LLM-based recommender, the ranking is based on the L2 distance between item embeddings and the embedding of the LLM's generated output. As for the recommendation metrics, we adopt two widely-used top-K recommendation metrics: Hit Ratio (HR) and Mean Reciprocal Rank (MRR) [70], where HR only considers whether the ground-truth item in the top-K result, and MRR further takes the rank of the ground-truth item in the top-K list.

4.2 Sequence-Recovery Result (**RQ1**)

The straightforward approach to show whether LLM could understand the hidden representations of sequential recommenders is to let LLM recover the items encoded in the hidden representations with textual descriptions. This approach serves as a direct evaluation of the LLM's capability to decode and interpret the complex patterns embedded within the ID-based sequence representations that sequential recommenders typically generate. By requiring the LLM to produce natural language descriptions of the encoded items, we can assess both the accuracy of its understanding and the quality of its interpretations. In the testing data, each interaction sequence contains several movies, video games, or books, and the sequence recovery task is to recover these items based on the

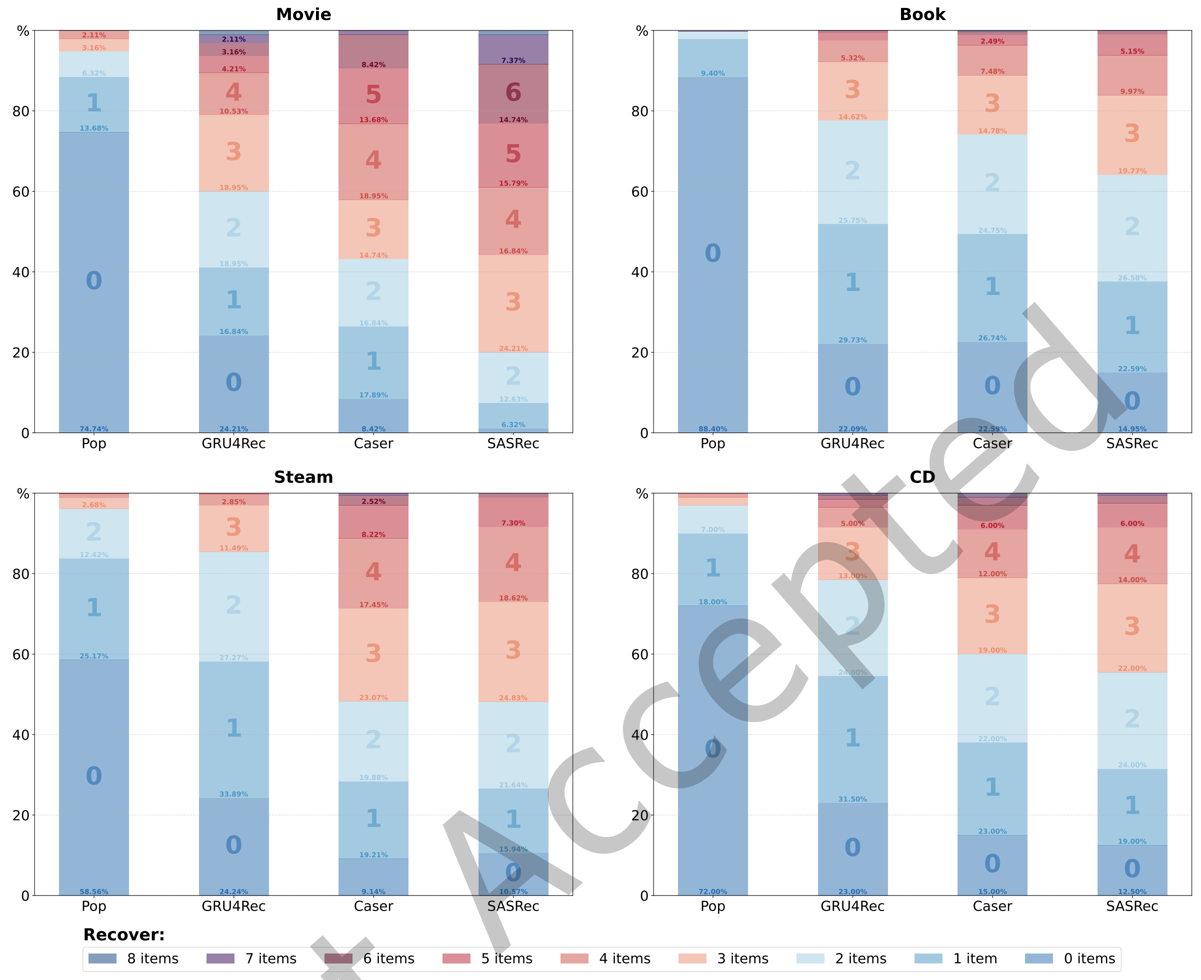

Fig. 5. The distribution of the number of recovered items on the Movie, Steam, Book and CD datasets, with GRU4Rec, Caser, and SASRec as the sequential recommenders. 'Pop' denotes that the 10 most popular movies or games in the training data are provided as the sequence recovery results. The average number of items in the test sequences is 10.00 for the Movie dataset, 8.89 for the Steam dataset, 9.88 for the Book dataset, and 9.95 for the CD dataset.

hidden representations of the sequential recommenders. This setup creates a challenging environment where the LLM must demonstrate its ability to understand abstract sequence representations back to specific items across different entertainment domains. It is worth noting that sequence recovery is not a trivial task due to: 1) The large number of possible items makes the recovery process particularly challenging, as the model must distinguish between thousands of potential candidates; and 2) The sequence embeddings are highly compressed with several concrete items encoded in one embedding representation, which means that the LLM must understand not only individual items but also their relationships within the sequence. We illustrate the result in Figure 5, and we present the observed cases in the inference phase in Figure 6. More cases on all the datasets are shown in Table 10 and Table 11 in Appendix B.

From Figure 5 we can observe that:

- If we naively provide the most popular items in the training data as the recovery of interaction sequences, they can hardly match the real interacted items. In the Movie dataset, among 74.74% of the test samples, the popularity-based recovery strategy can not recover any items. Similarly, in the Steam dataset, the ratio of recovering 0 items based on popularity is 58.56%. These results suggest that the sequence recovery task is quite challenging, which a simple heuristic of popularity can hardly handle. This finding underscores the complexity of the task and the need for more sophisticated approaches that can capture the nuanced patterns in user interactions beyond simple popularity metrics, which can be better accomplished by our proposed sequence-recovery prompting in RecInterpreter.
- In general, with our designed sequence recovery framework, LLM shows the capability of understanding hidden representations of sequential recommenders. Notably, in the Movie dataset, LLM could recover more than 5 items from the hidden representations of Caser and SASRec for over 35% of all test samples, and the percentage of recovering more than 3 items from the hidden representations could reach 80%. Therefore, we could safely draw the conclusion that the hidden representations of interaction sequences encoded by sequential recommender are also perceivable for LLM, just as the hidden representations of images, audios, and videos. This finding establishes an important parallel between recommendation systems and other modal domains, suggesting a universal capability of LLMs in understanding various forms of encoded information including the behavioral patterns in recommender systems.
- In the comparison of the Movie, Steam, Book and CD datasets, we can observe that LLM shows a better understanding of the Movie data than the Steam, Book and CD data. One reason is that the item titles in the Steam, Book and CD datasets are more complex than the movie titles in the Movie dataset. For example, the movie titles in the Movie dataset are quite simple and clear containing only English words. However, the game titles in the Steam dataset are more complicated, where plenty of the game titles contain the version or provider information, such as *"Swords and Sorcery - Underworld - Definitive Edition"*. Besides, some of the titles contain other languages than English, such as Chinese or Japanese. Another possible reason is that the movie-related corpus (such as reviews and comments) might appears more in the training data of LLM, which makes it easier for LLM to achieve the best recovery performance on movie data.
- In the comparison of different sequential recommenders, we can observe that LLM can better understand Caser, and SASRec than GRU4Rec. The reason comes from their different model architectures. Specifically, Caser adopts CNN to capture the sequential patterns, and CNN has displayed the impressive capability of encoding the global information [21]. The Transformer encoder employed by SASRec is one of the widely adopted sequence-to-sequence architectures [62], which is also the fundamental component in the framework of LLM [60]. However, RNN may suffer from issues such as vanishing gradient and exploding gradient [50], thus presenting obstacles for LLM to well understand its hidden representations. These architectural differences and their impact on model performance highlight the importance of choosing appropriate sequential modeling approaches for recommendation systems that need to interface with LLMs.

Position Order Analysis. To further evaluate the LLM's capability in understanding not only the items but also their sequential relationships, we analyze whether the correctly recovered items maintain their relative position order as in the original sequence. For each pair of correctly recovered items, we check if their relative order is consistent with the original sequence, and compute the accuracy across all such pairs. As shown in Table 2, the position order accuracy ranges from 71.43% to 82.17% across different datasets and recommenders, indicating that LLM not only recognizes individual items but also largely preserves their sequential arrangement. This suggests that the hidden representations encode not just item identities but also meaningful sequential patterns that LLM

Table 2. Position order analysis of correctly recovered items with different pretrained recommenders

Dataset	GRU4Rec	Caser	SASRec
Movie	78.65%	82.17%	81.93%
Steam	74.28%	77.36%	79.82%
Book	71.43%	75.89%	76.52%
CD	72.09%	74.62%	75.38%

Table 3. Distribution of correctly recovered items across sequence positions. Values represent the percentage of correctly recovered items falling within each position range.

Dataset	GRU4Rec			Caser			SASRec		
	Early (1-3)	Middle (4-6)	Late (7-10)	Early (1-3)	Middle (4-6)	Late (7-10)	Early (1-3)	Middle (4-6)	Late (7-10)
Movie	22.4%	31.8%	**45.8%**	33.2%	32.5%	34.3%	31.5%	34.8%	33.7%
Steam	19.1%	25.3%	**55.6%**	34.7%	30.9%	34.4%	33.4%	32.1%	34.5%
Book	24.7%	32.4%	**42.9%**	32.8%	33.6%	33.6%	30.8%	34.9%	34.3%
CD	16.8%	24.2%	**59.0%**	31.4%	34.8%	33.8%	32.9%	33.2%	33.9%

can interpret. Notably, Caser and SASRec consistently outperform GRU4Rec in maintaining position accuracy, further confirming their superior capability in representing sequential information in a way that is interpretable to LLMs.

Table 4. Popularity bias in failed recovery cases with different pretrained recommenders

Dataset	GRU4Rec	Caser	SASRec
Movie	63.82%	58.45%	61.29%
Steam	62.56%	59.31%	60.18%
Book	65.14%	61.53%	63.67%
CD	64.39%	60.28%	56.95%

Position-aware Recovery Pattern Analysis. To provide deeper insights into the position accuracy patterns, we conduct a comprehensive analysis of correctly recovered items across different sequence positions. We categorize sequence positions into three groups: early positions (1-3), middle positions (4-6), and late positions (7-10), and examine the distribution of correctly recovered items across these position ranges. The result is shown in Table 3.

The results reveal distinct architectural-specific patterns that provide insights into how different sequential recommenders encode temporal information. GRU4Rec exhibits a distinctive "recency bias" where over 42% of correctly recovered items come from late positions (7-10) across all datasets, while only 16-25% come from early positions (1-3). This pattern aligns with the inherent characteristics of recurrent neural networks, where the hidden state primarily captures recent information due to the vanishing gradient problem. The sequential processing nature of RNNs means that earlier items in the sequence have weaker influence on the final hidden representation, making them less likely to be correctly recovered by the LLM. In contrast, both Caser and SASRec demonstrate more balanced recovery distributions across all positions. This balanced pattern reflects the architectural advantages of their respective designs. Caser's sliding window approach of CNNs can capture patterns across different temporal scales uniformly, resulting in more consistent position coverage in the recovered sequences. Similarly, SASRec benefits from the self-attention mechanism that can directly model relationships between all positions without the sequential bottleneck inherent in RNNs.

While quantifying item types proves challenging due to the subjective nature of categorization, several potential factors may influence position maintenance patterns. Item popularity likely plays a role, as well-known items may benefit from richer semantic associations in the LLM's pre-training corpus, potentially leading to more stable positional encoding. Additionally, items within similar semantic categories may exhibit different position recovery behaviors, though systematic quantification of these effects remains difficult without domain-specific taxonomies.

These position-sensitive patterns have important implications for applications where sequential order matters. In e-commerce scenarios, product browsing sequences where early items indicate initial intent while later items reflect refined preferences require architecture-specific consideration for temporal weighting. Content consumption platforms such as video streaming with episode dependencies, educational systems with prerequisite relationships, and social media platforms where temporal dynamics influence engagement all benefit from understanding these architectural biases. The position recovery analysis demonstrates that RecInterpreter not only recovers items but also preserves meaningful temporal relationships, making it particularly valuable for applications requiring position-aware recommendations where the choice between RNN-based recency weighting versus balanced temporal coverage becomes crucial for system design.

Failure Analysis. To better understand the limitations of our approach, we conducted a detailed analysis of failure cases in the sequence recovery task. As shown in Table 4, popularity bias emerges as the primary cause of recovery errors. Approximately 60% of incorrectly recovered items across all datasets are among the top 10% most popular items in the training set. This suggests that when LLM is uncertain about which item to recover, it tends to default to popular items that appeared frequently during training. The effect is most pronounced with GRU4Rec, which shows the highest popularity bias across all datasets, particularly in the Book dataset (65.14%).

Beyond popularity bias, we observed other error patterns that are less amenable to quantitative analysis. Some errors stem from semantic similarity between items, such as confusing different installments within the same series. For example, in the Book dataset, the model occasionally mixes up different volumes in the Harry Potter series or other sequential works. Similarly, in the Movie dataset, sequels or films with similar themes or actors are sometimes interchanged. These errors suggest that while LLM can broadly understand the domain and genre encoded in the hidden representations, it may struggle with fine-grained distinctions between semantically similar items. Such cases are difficult to systematically categorize and quantify, hence we do not present statistical results for these observations.

4.3 Sequence-Residual Result (**RQ2**)

We have shown that LLM could understand the interactions from the hidden representation, but it is hard to recover all items, since the number of candidates is very large and the hidden representation is highly compressed. Therefore, we design the sequence-residual prompting framework, *i.e.,* inspiring LLM to identify the residual item by comparing the representations before and after the sequence incorporates the said residual. Instead of attempting to recover all items simultaneously, it focuses on identifying the incremental change in the representation caused by a single item addition. This targeted approach reduces the complexity of the task significantly. Moreover, by comparing representations before and after the incorporation of a residual item, the model can learn to detect subtle patterns and changes that characterize specific items' impact on the sequence representation.

We illustrate the result in Table 5, from which we can observe:

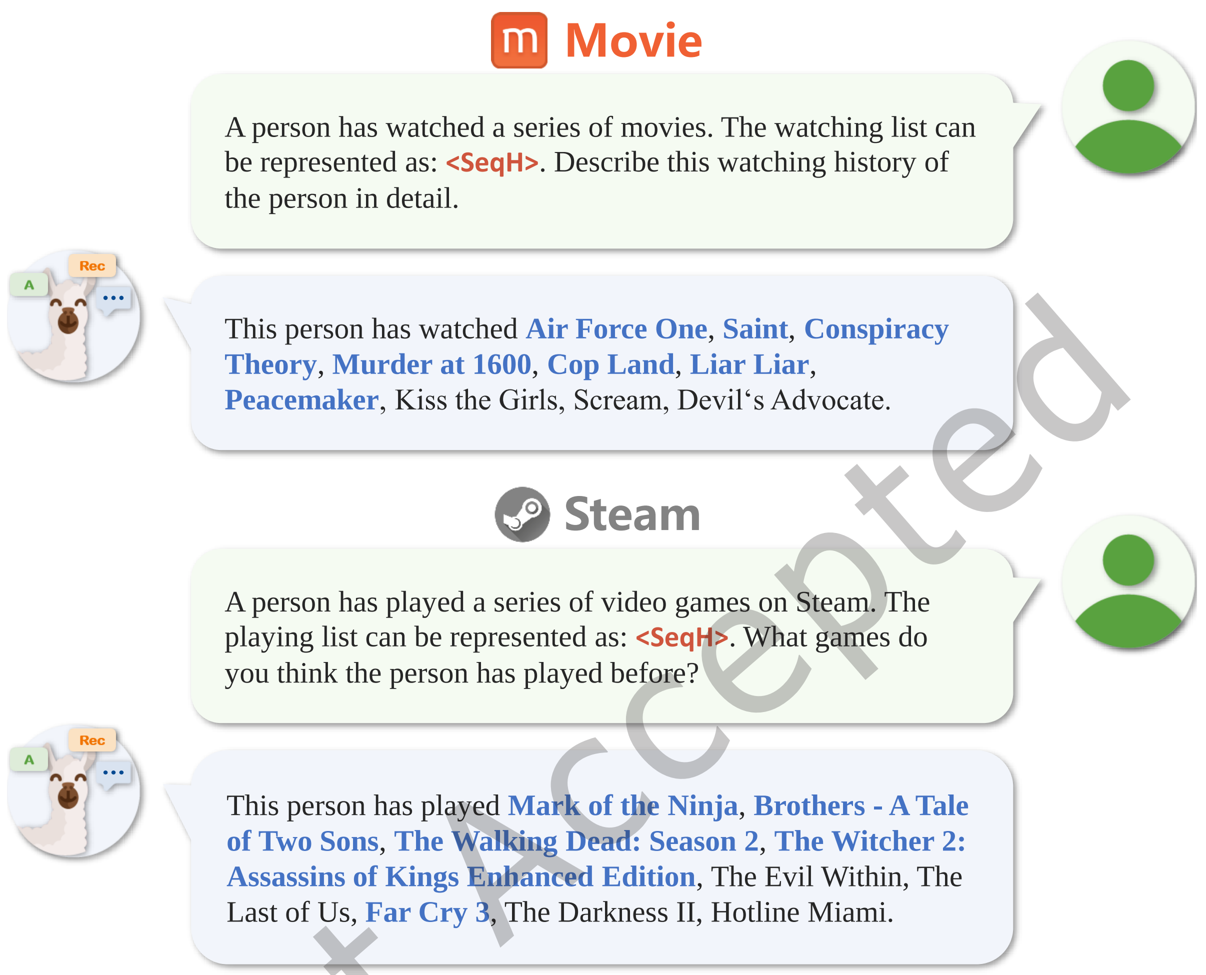
Movie
A person has watched a series of movies. The watching list can be represented as: <SeqH>. Describe this watching history of the person in detail.
Rec
A
This person has watched Air Force One, Saint, Conspiracy Theory, Murder at 1600, Cop Land, Liar Liar, Peacemaker, Kiss the Girls, Scream, Devil's Advocate.
Steam
A person has played a series of video games on Steam. The playing list can be represented as: <SeqH>. What games do you think the person has played before?
This person has played Mark of the Ninja, Brothers - A Tale of Two Sons, The Walking Dead: Season 2, The Witcher 2: Assassins of Kings Enhanced Edition, The Evil Within, The Last of Us, Far Cry 3, The Darkness II, Hotline Miami.

Fig. 6. Two cases of the sequence recovery task in Movie and Steam datasets. <SeqH> denotes the hidden representation of the interaction sequence after the projection layer. The blue text in the response denotes the correctly recovered movies and games only from the hidden representation.

- Among the conventional recommenders evaluated on the sequence-residual task, Caser and SASRec demonstrate superior accuracy in residual item identification compared to GRU4Rec. This performance difference aligns with our observations from the sequence-recovery framework and can be attributed to their more sophisticated architectural designs. Specifically, Caser leverages convolutional neural networks (CNN) while SASRec employs a Transformer encoder architecture, both of which prove more effective at capturing sequential patterns compared to the recurrent neural network (RNN) structure used in GRU4Rec. These findings highlight the importance of architectural design in sequential recommendation systems and demonstrate how proper architectures can lead to better representation learning, ultimately resulting in improved performance in identifying the residual item with LLMs.

Table 5. The result of sequence residual on the Movie, Steam, Book and CD datasets. Note that GRU4Rec, Caser, and SASRec denote that we leverage them as the conventional sequential recommender and employ the sequence-residual prompting on top of them. We calculate the accuracy of correctly identifying the residual item in the test data as the evaluation metrics.

Dataset	GRU4Rec	Caser	SASRec
Movie	54.71%	80.23%	94.58%
Steam	18.23%	55.67%	54.50%
Book	15.05%	47.24%	50.47%
CD	16.50%	47.36%	52.20%

- LLaMA also understands the Movie dataset better than the Steam, Book and CD datas *w.r.t.* the sequence-residual task. The reason also comes from that the titles of video games are more complicated than the titles of movies. Besides, another potential reason can be that the movie-related corpus exists more in the training data of LLM, which LLM achieves better performance on Movie data.
- Overall, RecInterpreter demonstrates remarkable comprehension capabilities in the sequence-residual task across all datasets, particularly excelling in identifying items that deviate from the primary sequential patterns. This strong performance indicates that our framework not only captures the main sequential dependencies but also exhibits sophisticated understanding of subtle variations and anomalies within user interaction sequences. Such capability is crucial for real-world recommendation scenarios where user behaviors often contain both consistent patterns and occasional deviations, further validating the effectiveness of our approach in modeling complex sequential dynamics.

4.4 Recommendation Result (**RQ3**)

In this section, we compare the recommendation performance of RecInterpreter with several baseline methods, including: 1) Representative traditional sequential recommenders: GRU4Rec [23], Caser [58], and SASRec [31]. 2) Recent LLM-based recommenders: ChatRec [16], MoRec [74], BIGRec [3], LLaRA [37], CoLLM [79], and RecLora [82]. These baselines represent the evolution of sequential recommendation methods, from traditional neural architectures to modern LLM-based approaches, providing a comprehensive evaluation framework for our proposed RecInterpreter model.

The results are shown in Table 6, from which we can observe that:

- Generally, RecInterpreter achieves the best recommendation performance compared to all baselines, under both the sequence-recovery and sequence-residual promptings. The performance gains are consistent across different datasets, highlighting the effectiveness of our approach. This notable improvement in performance can be attributed to RecInterpreter's novel design that explicitly guides the large language model to interpret hidden behavioral pattern representations through textual descriptions using our carefully designed sequence-recovery prompting. Unlike previous approaches that either rely solely on traditional sequential modeling or use language models in a more general way, our method creates a direct bridge between the sequential patterns learned by recommendation models and the linguistic understanding capabilities of LLMs. Through this explicit prompting mechanism, the language model develops a comprehensive understanding of the behavioral patterns that sequential recommenders have encoded. This synergistic combination of explicit pattern interpretation and efficient parameter tuning sets RecInterpreter apart from existing methods and explains its consistent performance advantages across different experimental settings.

Table 6. Recommendation performance comparison between RecInterpreter and baselines on four datasets. Bold indicates the best performance and the underline indicates the best among baselines. Rec_g, Rec_c, and Rec_s denote RecInterpreter with GRU4Rec, Caser, and SASRec as sequential recommenders respectively. We adopt a all-ranking task following prior study [3]. Experiments are conducted 3 times and the average is reported.

		Movie		Steam		Book		CD	
		MRR@20	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20	HR@20
Conventional	GRU4Rec	0.0614	0.1533	0.0441	0.1104	0.0433	0.1082	0.0141	0.0352
	Caser	0.0627	0.1392	0.0477	0.1193	0.0454	0.1136	0.0147	0.0368
	SASRec	0.0450	0.1125	0.0436	0.1091	0.0423	0.1059	0.0144	0.0361
LLM-based	ChatRec	0.0301	0.0753	0.0379	0.0947	0.0409	0.1022	0.0095	0.0238
	MoRec	0.0431	0.1078	0.0415	0.1036	0.0371	0.0927	0.0103	0.0257
	BIGRec	0.0630	0.1387	0.0492	0.1232	0.0511	0.1278	0.0148	0.0371
	LLaRA	<u>0.0716</u>	<u>0.1679</u>	<u>0.0525</u>	<u>0.1313</u>	<u>0.0537</u>	<u>0.1341</u>	<u>0.0156</u>	<u>0.0392</u>
	CoLLM	0.0703	0.1663	0.0513	0.1298	0.0526	0.1329	0.0145	0.0377
	RecLora	0.0683	0.1549	0.0511	0.1279	0.0527	0.1317	0.0152	0.0379
Seq-Recovery	Rec_g	0.0889	0.2222	0.0590	0.1475	0.0571	0.1428	0.0168	0.0421
	Rec_c	**0.0944**	**0.2360**	0.0601	0.1504	0.0547	0.1368	0.0173	0.0432
	Rec_s	0.0933	0.2333	0.0576	0.1441	**0.0583**	**0.1457**	0.0177	0.0441
Seq-Residual	Rec_g	0.0927	0.2319	0.0592	0.1479	0.0576	0.1441	0.0171	0.0428
	Rec_c	0.0940	0.2350	**0.0606**	**0.1515**	0.0565	0.1413	**0.0179**	**0.0447**
	Rec_s	0.0932	0.2331	0.0578	0.1445	0.0558	0.1395	0.0175	0.0438

- When combined with different sequential recommenders, RecInterpreter displays different recommendation performances. Our experimental results show that RecInterpreter achieves superior performance when integrated with Caser and SASRec compared to its combination with GRU4Rec. This performance pattern mirrors what we observed in the sequence recovery task, and can be attributed to the architectural advantages of these models, *i.e.,* Caser and SASRec employ more advanced model architectures than GRU4Rec. Therefore, Caser and SASRec could better encode the behavioral patterns of users in their learning process.
- Among the LLM-based recommendation approaches, our experimental results reveal several key insights regarding their performance characteristics. In-context learning approaches demonstrate limited effectiveness in recommendation tasks, primarily due to the nature of LLM pre-training. As noted in previous studies [3, 37, 79, 82], the pre-training process of large language models typically includes minimal exposure to recommendation-specific tasks or relevant corpus. Therefore, it is challenging for LLM to perform recommendation tasks without fine-tuning. MoRec's performance falls somewhat short of expectations in our experiments. This can be attributed to our experimental setup where only item titles are used for constructing representations to ensure fair comparison across all approaches. As suggested in the original MoRec paper [74], the model's full potential may require additional modality information,

such as user profiles and item images, to leverage its multimodal architecture effectively. Furthermore, LLaRA's superior performance over BIGRec suggests that integrating conventional recommenders can further improve LLM's performance for recommendation tasks. In summary, more powerful backbone models, additional modality data, and integration of supplementary behavioral patterns seem to boost the performance of LLM-based recommenders.
- Comparing the approaches that utilize parameter tuning (BIGRec, LLaRA, CoLLM, RecLoRA and RecInterpreter), our experiments demonstrate RecInterpreter's superior performance. An analysis of their methodological differences provides insights into the sources of this performance advantage. BIGRec adopts a straightforward approach by utilizing only textual information, specifically the sequence of item titles, to describe user historical behavior in its prompt design. While this method leverages the language model's natural text processing capabilities, it may not fully capture the complex patterns in user behavior sequences. LLaRA advances this approach by incorporating both textual information and mapped hidden representations obtained from sequential recommenders in its prompt design. However, LLaRA's integration of hidden representations with the language model remains implicit, as it optimizes only the recommendation objective as shown in Equation 8. This implicit integration may limit the model's ability to fully leverage the behavioral patterns encoded in these representations. RecInterpreter introduces a different approach by explicitly teaching the language model to understand hidden representations through the proposed sequence-recovery prompting. This novel mechanism enables the model to operate directly with projected hidden representations when describing user behavior history, effectively discarding the need for raw textual information. By establishing this explicit bridge between behavioral patterns and language understanding, RecInterpreter achieves a more comprehensive interpretation of user preferences. The significant performance improvements observed in our experiments validate both the theoretical rationale behind this approach and its practical superiority in recommendation tasks.

4.5 Efficiency analysis

In this section, we analyze the computational efficiency of RecInterpreter compared to baseline methods. Table 7 presents the training time in GPU hours for all methods across four datasets (We exclude ChatRec since ChatRec utilizes in-context learning and does not require training the model). Several key observations can be made from these results:

- Conventional sequential recommenders (GRU4Rec, Caser, and SASRec) are extremely efficient, requiring only 1-3 GPU hours for training across all datasets. This efficiency stems from their relatively small parameter size and specialized architectures optimized for sequential recommendation tasks. However, as shown in Table 6, these models achieve significantly lower performance compared to LLM-based approaches. As LLM-enhanced recommender, MoRec also has low training time as it primarily focuses on modality enhancement without extensive LLM tuning.
- Among LLM-based recommenders, BIGRec demonstrates moderate efficiency with training times approximately half of other LLM-tuning methods. LLaRA and CoLLM exhibit higher computational demands, mainly because they directly input embeddings into the LLM for fine-tuning without the interpretable projection mechanism that RecInterpreter employs. This direct input approach not only increases computational cost but also results in lower performance compared to our method, as shown in Table 6.
- Our RecInterpreter framework involves a two-phase training procedure: (1) training the projector layer (either Sequence-Recovery or Sequence-Residual) and (2) recommendation fine-tuning. While this two-phase approach results in longer total training times compared to other LLM-based methods (approximately 1-2 times longer), it brings substantial performance improvements. The computational investment is justified

Table 7. Training efficiency comparison between RecInterpreter and baselines on four datasets. All times are reported in GPU hours. Rec_g, Rec_c, and Rec_s denote RecInterpreter with GRU4Rec, Caser, and SASRec as sequential recommenders respectively. For RecInterpreter, we report the time for both training phases: projector training and recommendation fine-tuning.

		Training Time (GPU Hours)			
		Movie	Steam	Book	CD
Conventional	GRU4Rec	1.2	1.8	1.6	2.5
	Caser	1.3	2.0	1.8	2.7
	SASRec	1.4	2.1	1.9	2.8
LLM-based	MoRec	1.8	2.5	2.3	3.2
	BIGRec	15.4	43.2	31.8	195.3
	LLaRA	29.8	83.5	61.4	376.5
	CoLLM	25.6	72.1	53.2	325.7
	RecLora	24.9	70.4	51.8	317.2
Seq-Recovery	Rec_g (Projector)	35.1	158.6	92.8	629.6
	Rec_c (Projector)	40.2	181.5	106.2	720.6
	Rec_s (Projector)	37.8	170.8	99.5	677.7
	Rec_g (Recommendation)	7.0	31.7	18.6	125.9
	Rec_c (Recommendation)	8.0	36.3	21.3	144.2
	Rec_s (Recommendation)	7.6	34.1	20.0	135.4
Seq-Residual	Rec_g (Projector)	26.5	102.7	65.7	468.2
	Rec_c (Projector)	30.3	117.5	75.1	535.8
	Rec_s (Projector)	28.4	110.1	70.4	503.6
	Rec_g (Recommendation)	9.3	39.5	23.1	153.6
	Rec_c (Recommendation)	10.1	41.5	35.0	177.2
	Rec_s (Recommendation)	9.7	38.1	24.1	160.7

by the significant gains in recommendation quality, with RecInterpreter consistently outperforming all baselines across all datasets.

- When comparing our two prompting strategies, we observe an interesting time-performance trade-off. The Sequence-Recovery approach requires more GPU hours in the first phase (projector training) but less time in the recommendation fine-tuning phase compared to Sequence-Residual. This is because the more comprehensive understanding of user behavior patterns learned during Sequence-Recovery prompting allows for more efficient adaptation during the recommendation phase. Conversely, Sequence-Residual requires less time for projector training but more time for recommendation fine-tuning, as the model needs to compensate for the more focused nature of the residual understanding.

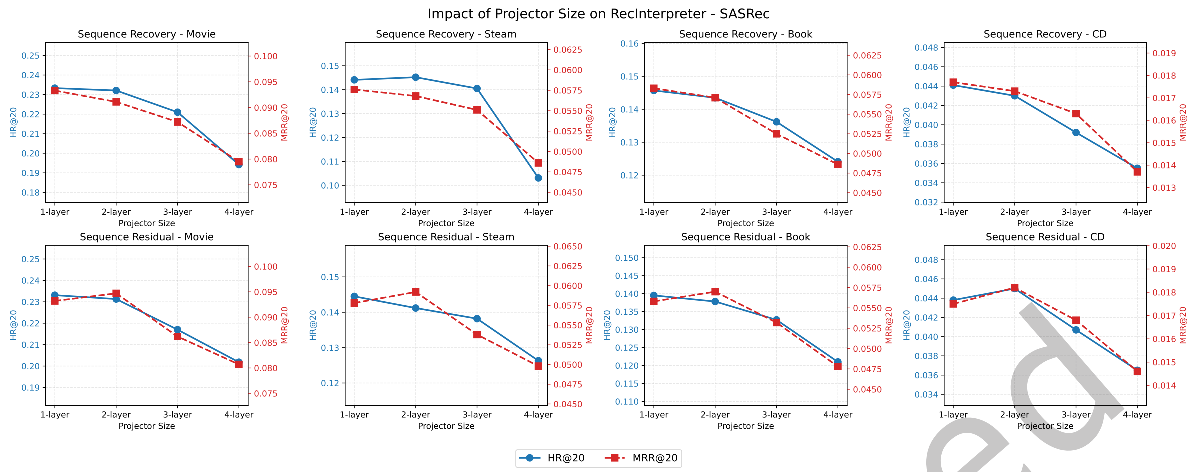

Fig. 7. Impact of projector architecture on RecInterpreter (SASRec) performance across four datasets. The figure shows both HR@20 (blue, left y-axis) and MRR@20 (red, right y-axis) metrics for different projector sizes ranging from single-layer to four-layer MLPs. Top row shows results with Sequence-Recovery prompting, while bottom row shows results with Sequence-Residual prompting. Performance generally peaks with simpler architectures (1-2 layers) and gradually declines with more complex architectures (3-4 layers), suggesting that one-layer lightweight projector is sufficient for effective latent space interpretation.

Overall, while RecInterpreter demands higher computational resources than conventional recommenders, the investment yields substantial performance gains. Compared to other LLM-based methods, RecInterpreter's two-phase approach offers a favorable balance between computational efficiency and recommendation quality. The explicit interpretation mechanism not only improves performance but also enhances explainability, providing valuable insights into user preferences that are absent in other approaches.

4.6 In-depth Analysis

To investigate the impact of different components in RecInterpreter and to shed light on the reasons why the proposed RecInterpreter outperforms baselines, we conduct in-depth analysis with experiments in this section, including the projector size, other alignment strategy, and the cold-start ability.

4.6.1 Ablation on the Projector Size. The projector, which maps sequence representations from recommender systems into the LLM token embedding space, is a critical component in our framework. We investigate how different projector sizes influence RecInterpreter's performance. Figure 7 presents the results of experiments with increasingly complex MLP sizes (1-layer: A simple linear projection [1024 ⟶ 4096]; 2-layer: [1024 ⟶ 2048 ⟶ 4096]; 3-layer: [1024 ⟶ 2048 ⟶ 2072 ⟶ 4096]; 4-layer: [1024 ⟶ 1024 ⟶ 2048 ⟶ 2072 ⟶ 4096]). All configurations use GELU activation between layers. We evaluate these architectures across four datasets (Movie, Steam, Book, and CD) for both sequence-recovery and sequence-residual prompting strategies.

The results reveal a consistent pattern: simpler projector architectures (1-2 layers) consistently outperform more complex ones across all datasets. For instance, in the Movie dataset with Sequence-Recovery prompting, the 1-layer projector achieves an HR@20 of 0.2333, while performance gradually decreases with 2-layer (0.2321), 3-layer (0.2210), and 4-layer (0.1942) architectures. Similar trends are observed in MRR@20 metrics and across all datasets. This finding is particularly interesting considering that deeper neural networks typically have greater

Table 8. Zero-shot recommendation performance without second-stage SFT.

		Movie		Steam		Book		CD	
		MRR@20	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20	HR@20
Baseline		0.0148	0.0370	0.0116	0.0290	0.0133	0.0333	0.0035	0.0088
Seq-Recovery	Rec_g	0.0262	0.0655	0.0194	0.0485	0.0211	0.0528	0.0064	0.0161
	Rec_c	0.0348	0.0870	0.0231	0.0578	0.0215	0.0538	0.0069	0.0173
	Rec_s	0.0336	0.0840	0.0241	0.0603	0.0236	0.0590	0.0074	0.0186
Seq-Residual	Rec_g	0.0250	0.0626	0.0183	0.0459	0.0194	0.0485	0.0062	0.0155
	Rec_c	0.0325	0.0812	0.0225	0.0563	0.0205	0.0514	0.0067	0.0167
	Rec_s	0.0316	0.0790	0.0232	0.0581	0.0224	0.0561	0.0072	0.0179

representational capacity. The performance degradation with deeper projectors can be attributed to several factors. First, overfitting becomes more likely with increased parameter count, especially given the relatively limited size of recommendation datasets. Second, information can be lost through multiple transformation layers, potentially diluting the essential behavioral patterns encoded in the sequence representations. Third, deeper networks face optimization challenges that may prevent them from learning optimal mappings between representation spaces.

Based on these comprehensive results, we adopt the 1-layer projector [1024 $\longrightarrow$ 4096] in our main experiments, as it offers the best balance between performance and computational efficiency. This finding aligns with recent research in multimodal LLMs [41, 81], where simpler projection mechanisms have been found to be effective for aligning different representation spaces with token embeddings of language models.

4.6.2 Zero-shot Performance without the Second-stage SFT. To further investigate the effectiveness of our proposed RecInterpreter framework, we conduct a zero-shot evaluation experiment where we only implement the sequence representation interpretation phase (either Sequence-Recovery or Sequence-Residual) without performing the second-stage supervised fine-tuning for recommendation. This experiment aims to verify whether the enhanced understanding of user behavior patterns through our interpretation mechanisms alone can improve recommendation performance, even without explicit optimization for the recommendation objective.

In this zero-shot setting, we directly use the trained projector to map sequence representations into the LLM's token embedding space, and then prompt the LLM to generate recommendations based only on its understanding of these projected representations. As a baseline, we compare against a straightforward approach where LLaMA2-7B is prompted with the raw textual titles of the user's historical interaction sequence, which is a common practice in prompt-based recommendation methods.Table 8 presents the performance comparison between our zero-shot RecInterpreter variants and the baseline approach across four datasets. The results reveal several interesting findings:

- RecInterpreter variants significantly outperform the title-based prompting baseline across all datasets. This substantial improvement demonstrates that the understanding of hidden behavioral patterns encoded in sequence representations is indeed more effective than directly leveraging raw textual descriptions of items for recommendation tasks.
- We observe that Sequence-Recovery consistently outperforms Sequence-Residual in the zero-shot setting. This suggests that the more comprehensive understanding of user preferences gained through recovering

Table 9. Performance comparison of different alignment strategies using SASRec as the sequential recommender.

	Movie		Steam		Book		CD	
	MRR@20	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20	HR@20
Title-based	0.0630	0.1387	0.0492	0.1232	0.0511	0.1278	0.0148	0.0371
Item-level	0.0552	0.1210	0.0448	0.1102	0.0472	0.1156	0.0130	0.0325
Sequence-level	0.0933	0.2333	0.0576	0.1441	0.0583	0.1457	0.0177	0.0441

entire interaction sequences provides a stronger foundation for recommendation tasks compared to the more focused residual item identification approach. This finding aligns with our intuition that a holistic understanding of user behavior is particularly valuable when no explicit recommendation fine-tuning is performed.

The zero-shot performance is, as expected, substantially lower than our fully-trained RecInterpreter variants presented in Table 6. The substantial performance gap between zero-shot (HR@20: 0.0840 on Movie dataset) and fully-trained versions (HR@20: 0.2333) is expected and reasonable, given that the latter undergoes dedicated recommendation task fine-tuning while the former relies solely on interpretation capabilities without task-specific adaptation.

The true value of this zero-shot experiment lies in its validation purpose: compared to the baseline title-based prompting approach, our method demonstrates significant improvements across all datasets, providing compelling evidence that the sequence interpretation capabilities developed in the alignment phase are indeed effective. This validates the core contribution of our sequence-level alignment strategy - that LLMs can meaningfully understand hidden behavioral patterns encoded by sequential recommenders. The complete RecInterpreter framework combines this interpretation capability with subsequent recommendation task fine-tuning, which explains the substantial performance gains shown in our main results. This two-stage design ensures both interpretability and recommendation effectiveness, with the zero-shot results confirming that the first stage successfully bridges the representation gap between sequential recommenders and LLMs.

4.6.3 Alignment Strategies for Sequence Recovery. A critical design choice in RecInterpreter is our sequence-level alignment approach, which differs from conventional alignment strategies in LLM-based recommendation. To validate the effectiveness of this approach, we conduct an ablation study comparing three different alignment strategies:

- **Title-based**: Directly using item titles as textual prompts for the LLM without embedding alignment, which is BIGRec [3].
- **Item-level**: Aligning individual item embeddings to the LLM's token space, then using these aligned embeddings for recommendation.
- **Sequence-level(Ours)**: Aligning entire sequence embeddings to the LLM's token space through sequence recovery.

As shown in Table 9, sequence-level alignment significantly outperforms both title-based and item-level alignment strategies across all datasets. These results highlight a limitation in existing LLM-based recommendation methods. Conventional approaches typically leverage item- or user-level alignment by directly projecting item or user embeddings into the token embedding space of LLMs. Although these methods have achieved certain effectiveness, they have overlooked the critical information encoded within sequential encoders that process user behavior sequences. This information is crucial for understanding user behaviors and preferences in sequential recommendation.

The inferior performance of item-level alignment compared to even title-based alignment is particularly revealing. It suggests that naively projecting individual item embeddings into the LLM's token space may actually disrupt the semantic coherence that exists in plain text descriptions. This finding challenges the assumption that embedding projection alone is sufficient for effective recommendation. Therefore, Studies such as LLaRA [37] and CoLLM [79] incorporates both item title and embedding in the prompts to achieve better performance than naively item title. In contrast, RecInterpreter first explicitly inspires LLM to interpret hidden behavioral pattern representations with textual narrations, as detailed in Section 3.1, which enables the LLM to understand the hidden representations in the first place. This foundational step is crucial as it establishes a strong semantic connection between the sequence embeddings of numerical ID representations and their meaning in natural language, creating a more robust basis for subsequent recommendation tasks. Unlike item- or user-level alignment, our sequence-level alignment leverages sequence embeddings generated by pretrained sequential recommenders. This approach is crucial to fully capture the rich information of user behavior patterns learned by specialized sequential models. By aligning at the sequence level, we preserve the temporal dynamics, item relationships, and evolving user preferences that are encoded in these representations.

The consistent performance gains across diverse datasets demonstrate that sequence-level alignment represents a more effective paradigm for bridging sequential recommenders and large language models. This alignment strategy allows RecInterpreter to combine the strengths of both worlds: the specialized sequential pattern recognition capabilities of traditional recommenders and the rich semantic understanding of LLMs.

5 CONCLUSION AND FUTURE WORK

We present RecInterpreter, a novel framework designed to enable LLMs to comprehend and interpret conventional sequential recommenders. Our approach is inspired by recent breakthroughs in multi-modal language models, which demonstrate that LLMs can effectively perceive hidden representations from modality-specific encoders through straightforward projection mechanisms. Building on this insight, RecInterpreter introduces sequence-recovery prompting and sequence-residual prompting that facilitate LLMs' understanding of sequential recommenders' hidden representations. The sequence-recovery prompting enables LLMs to reconstruct user interaction sequences from hidden representations, while the sequence-residual prompting further challenges LLMs to identify residual items in incomplete sequences. Furthermore, we develop an innovative method for fine-tuning LLMs for recommendation tasks, leveraging the insights gained through these prompting techniques. Notably, after successful interpretation of sequential recommenders, RecInterpreter enables LLMs to enhance recommendation performance even without requiring textual descriptions of interaction sequences, suggesting that the projected representations serve as a sufficient and compact form of user history, containing the necessary information for generating accurate recommendations.

Despite its demonstrated effectiveness, RecInterpreter has certain limitations that warrant acknowledgment. First, our current implementation employs simple MLP layer for projection, which may not fully capture the complexity of the representation space. Second, the maximum sequence length is limited to 10 items due to computational constraints, which may not reflect real-world scenarios where users have extensive interaction histories. If computational resources allow, future work should explore longer sequence handling through strategies such as hierarchical representation learning, sliding window approaches, or attention-based sequence compression techniques. Third, the scale of datasets used in our experiments, while sufficient for validation, could benefit from expansion to better demonstrate the framework's capabilities. We anticipate that these limitations can be addressed in future research through the incorporation of more sophisticated adaptation mechanisms, such as Q-former [33], and the utilization of larger-scale datasets. As an effort to enable explicit LLM understanding of recommendation modality, RecInterpreter opens up numerous promising research directions. Online service providers could implement RecInterpreter within their existing recommendation systems, exploring new applications of LLMs in this context. Additionally, the research community can investigate alternative frameworks beyond sequence-recovery to enhance LLMs' comprehension of recommendation modality. This initial work serves as a foundation for future innovations in combining language models with recommendation systems.

ACKNOWLEDGMENTS

This research is supported by the National Science and Technology Major Project (2023ZD0121102) and the National Natural Science Foundation of China (U24B20180, 62525211, 62302321).

REFERENCES

[1] Qingyao Ai, Ting Bai, Zhao Cao, Yi Chang, Jiawei Chen, Zhumin Chen, Zhiyong Cheng, Shoubin Dong, Zhicheng Dou, Fuli Feng, et al. 2023. Information Retrieval Meets Large Language Models: A Strategic Report from Chinese IR Community. *AI Open* 4 (2023), 80–90.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. In *NeurIPS*.

[3] 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).

[4] 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.

[5] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *WSDM*. 46–54.

[6] Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. 2021. High-Performance Large-Scale Image Recognition Without Normalization. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. 1059–1071.

[7] Zheng Chen. 2023. PALR: Personalization Aware LLMs for Recommendation. *CoRR* abs/2305.07622 (2023).

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. 191–198.

[9] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. *CoRR* abs/2205.08084 (2022).

[10] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *CoRR* abs/2305.14314 (2023).

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[12] Chenlu Ding, Jiancan Wu, Yancheng Yuan, Jinda Lu, Kai Zhang, Alex Su, Xiang Wang, and Xiangnan He. 2025. Unified Parameter-Efficient Unlearning for LLMs. In *ICLR*. OpenReview.net.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*. OpenReview.net.

[14] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. 2023. PaLM-E: An Embodied Multimodal Language Model. In *ICML (Proceedings of Machine Learning Research, Vol. 202)*. 8469–8488.

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

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

[17] Binzong Geng, Zhaoxin Huan, Xiaolu Zhang, Yong He, Liang Zhang, Fajie Yuan, Jun Zhou, and Linjian Mo. 2024. Breaking the Length Barrier: LLM-Enhanced CTR Prediction in Long Textual User Behaviors. *CoRR* abs/2403.19347 (2024).

[18] 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.

[19] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. 2023. Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model. *CoRR* abs/2304.13731 (2023).

[20] Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, and Pheng-Ann Heng. 2023. Point-Bind & Point-LLM: Aligning Point Cloud with Multi-modality for 3D Understanding, Generation, and Instruction Following. *CoRR* abs/2309.00615 (2023).

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[22] 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. *arXiv preprint arXiv:2308.10053* (2023).

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

[24] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 2023. 3D-LLM: Injecting the 3D World into Large Language Models. *CoRR* abs/2307.12981 (2023).

[25] Yupeng Hou, Zhankui He, Julian J. McAuley, and Wayne Xin Zhao. 2023. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *WWW*. ACM, 1162–1171.

[26] 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).

[27] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *arXiv preprint arXiv:2305.08845* (2023).

[28] 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 *ICLR*. OpenReview.net.

[29] Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2024. GenRec: Large Language Model for Generative Recommendation. In *ECIR (3) (Lecture Notes in Computer Science, Vol. 14610)*. Springer, 494–502.

[30] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A Critical Study on Data Leakage in Recommender System Offline Evaluation. *ACM Trans. Inf. Syst.* 41, 3 (2023), 75:1–75:27.

[31] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. 197–206.

[32] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. In *NeurIPS*.

[33] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *ICML (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 19730–19742.

[34] Kunchang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023. VideoChat: Chat-Centric Video Understanding. *CoRR* abs/2305.06355 (2023).

[35] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Personalized Prompt Learning for Explainable Recommendation. *ACM Trans. Inf. Syst.* 41, 4 (2023), 103:1–103:26.

[36] 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. *CoRR* abs/2312.02443 (2023).

[37] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, and Xiang Wang. 2024. LLaRA: LLaRA: Large Language-Recommendation Assistant.

[38] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2024. How Can Recommender Systems Benefit from Large Language Models: A Survey. *ACM Trans. Inf. Syst.* (2024).

[39] 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.

[40] Hao Liu, Lei Guo, Lei Zhu, Yongqiang Jiang, Min Gao, and Hongzhi Yin. 2024. MCRPL: A Pretrain, Prompt, and Fine-tune Paradigm for Non-overlapping Many-to-one Cross-domain Recommendation. *ACM Trans. Inf. Syst.* 42, 4 (2024), 97:1–97:24.

[41] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. In *NeurIPS*.

[42] Junling Liu, Chaoyong Liu, Renjie Lv, Kangdi Zhou, and Yan Bin Zhang. 2023. Is ChatGPT a Good Recommender? A Preliminary Study. *ArXiv* abs/2304.10149 (2023). https://api.semanticscholar.org/CorpusID:258236609

[43] Qidong Liu, Xian Wu, Xiangyu Zhao, Yejing Wang, Zijian Zhang, Feng Tian, and Yefeng Zheng. 2024. Large Language Models Enhanced Sequential Recommendation for Long-tail User and Item. *CoRR* abs/2405.20646 (2024).

[44] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *CoRR* abs/2110.07602 (2021).

[45] Muhammad Maaz, Hanoona Abdul Rasheed, Salman H. Khan, and Fahad Shahbaz Khan. 2023. Video-ChatGPT: Towards Detailed Video Understanding via Large Vision and Language Models. *CoRR* abs/2306.05424 (2023).

[46] Wenyu Mao, Jiancan Wu, Weijian Chen, Chongming Gao, Xiang Wang, and Xiangnan He. 2025. Reinforced Prompt Personalization for Recommendation with Large Language Models. *ACM Trans. Inf. Syst.* 43, 3 (2025), 72:1–72:27.

[47] Seungwhan Moon, Andrea Madotto, Zhaojiang Lin, Tushar Nagarajan, Matt Smith, Shashank Jain, Chun-Fu Yeh, Prakash Murugesan, Peyman Heidari, Yue Liu, et al. 2023. AnyMAL: An Efficient and Scalable Any-Modality Augmented Language Model. *CoRR* abs/2309.16058 (2023).

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

[49] OpenAI. 2023. GPT-4V(ision) System Card. Retrieved September 25, 2023 from https://cdn.openai.com/papers/GPTV_System_Card.pdf

[50] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML (3)*, Vol. 28. 1310–1318.

[51] Ruihong Qiu, Zi Huang, Tong Chen, and Hongzhi Yin. 2022. Exploiting Positional Information for Session-Based Recommendation. *ACM Trans. Inf. Syst.* 40, 2 (2022), 35:1–35:24.

[52] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[53] 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*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[54] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2023. Representation Learning with Large Language Models for Recommendation. *CoRR* abs/2310.15950 (2023).

[55] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *CVPR*. IEEE, 10674–10685.

[56] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Schärli, Aakanksha Chowdhery, Philip Andrew Mansfield, Blaise Agüera y Arcas, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2022. Large Language Models Encode Clinical Knowledge. *CoRR* abs/2212.13138 (2022).

[57] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. Large Language Models Enhanced Collaborative Filtering. *CoRR* abs/2403.17688 (2024).

[58] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[59] Guy Tennenholtz, Yinlam Chow, Chih-Wei Hsu, Jihwan Jeong, Lior Shani, Azamat Tulepbergenov, Deepak Ramachandran, Martin Mladenov, and Craig Boutilier. 2024. Demystifying embedding spaces using large language models. In *ICLR*.

[60] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[61] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023).

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[63] Chenyang Wang, Weizhi Ma, Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2023. Sequential Recommendation with Multiple Contrast Signals. *ACM Trans. Inf. Syst.* 41, 1 (2023), 11:1–11:27.

[64] Xin Wang, Hong Chen, Zirui Pan, Yuwei Zhou, Chaoyu Guan, Lifeng Sun, and Wenwu Zhu. 2024. Automated Disentangled Sequential Recommendation with Large Language Models. *ACM Trans. Inf. Syst.* (2024).

[65] Zhidan Wang, Lixin Zou, Chenliang Li, Shuaiqiang Wang, Xu Chen, Dawei Yin, and Weidong Liu. 2024. Toward Bias-Agnostic Recommender Systems: A Universal Generative Framework. *ACM Trans. Inf. Syst.* (2024).

[66] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. LLMRec: Large Language Models with Graph Augmentation for Recommendation. In *WSDM*.

[67] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A Survey on Large Language Models for Recommendation. *arXiv preprint arXiv:2305.19860* (2023).

[68] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2023. NExT-GPT: Any-to-Any Multimodal LLM. *CoRR* abs/2309.05519 (2023).

[69] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. *CoRR* abs/2303.17564 (2023).

[70] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI*. AAAI Press, 346–353.

[71] Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. *CoRR* abs/2306.10933 (2023).

[72] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *ICDE*. 1259–1273.

[73] 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.

[74] 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.

[75] An Zhang, Fangfu Liu, Wenchang Ma, Zhibo Cai, Xiang Wang, and Tat-Seng Chua. 2023. Boosting Differentiable Causal Discovery via Adaptive Sample Reweighting. *CoRR* abs/2303.03187 (2023).

[76] Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Is ChatGPT Fair for Recommendation? Evaluating Fairness in Large Language Model Recommendation. In *RecSys*. ACM, 993–999.

[77] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. *CoRR* abs/2305.07001 (2023).

[78] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR*. 11–20.

[79] 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.

[80] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting Large Language Models by Integrating Collaborative Semantics for Recommendation. In *ICDE*. IEEE, 1435–1448.

[81] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. *CoRR* abs/2304.10592 (2023).

[82] Jiachen Zhu, Jianghao Lin, Xinyi Dai, Bo Chen, Rong Shan, Jieming Zhu, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Lifelong Personalized Low-Rank Adaptation of Large Language Models for Recommendation. *CoRR* abs/2408.03533 (2024).

A PROMPTS

A.1 Sequence Recovery Prompts

1. A user has viewed/played/read/listened to several movies/video games/books/music albums. Their viewing/gaming/reading /listening history can be represented as: <Seq><SeqHere></Seq>. Could you list all the movies/video games/books/music albums this user has watched/played/read/listened to?

2. A person has watched/played/read/listened to a series of movies/video games/books/music albums. The watching/gaming /reading/listening list can be represented as: <Seq><SeqHere></Seq>. Describe this watching/gaming/reading/listening history of the person in detail.

3. A person has watched/played/read/listened to a series of movies/video games/books/music albums. The watching/gaming /reading/listening list can be represented as: <Seq><SeqHere></Seq>. Please provide a detailed description of this watching/gaming/reading/listening history of the person.

4. A user has a movie watching/video game playing/book reading/music album listening history that can be represented as: <Seq><SeqHere></Seq>. What movies/video games/books/music albums has this person watched/played/read/listened to based on this representation?

5. The following representation <Seq><SeqHere></Seq> describes a person's movie watching/video game playing/book reading/music album listening history. Please enumerate all the movies/video games/books/music albums they have watched/played/read/listened to.

6. A viewer's/gamer's/reader's/listener's movie/video game/book/music album history is encoded as: <Seq><SeqHere></Seq>. Please decode this representation and list all the movies/video games/books/music albums they have watched/played/read/listened to.

7. The sequence <Seq><SeqHere></Seq> represents a person's movie watching/video game playing/book reading/music album listening history. Can you translate this into a list of movie/game/book/album titles they've watched/played/read/listened to?

8. A cinephile's/gaming enthusiast's/bibliophile's/music enthusiast's watching/playing/reading/listening history is represented by <Seq><SeqHere></Seq>. Please identify all the movies/video games/books/music albums this person has watched/played/read/listened to.

9. I have a representation of someone's movie watching/video game playing/book reading/music album listening history: <Seq><SeqHere></Seq>. Please convert this into a complete list of the movies/video games/books/music albums they've watched/played/read/listened to.

10.The encoded movie/gaming/reading/listening history <Seq><SeqHere></Seq> represents all films/video games/books/music albums watched/played/read/enjoyed by a person. Please decode this and detail their complete viewing/playing/reading/listening history.

A.2 Sequence Residual Prompts

1. A person has watched/played/read/listened to a series of movies/video games/books/music albums. The watching/gaming /reading/listening list can be represented as List1: <Seq1><SeqHere1></Seq1>. After watching/playing/reading/listening to another movie/video game/book/music album, the watching/gaming/reading/listening list can further be represented as List2: <Seq2><SeqHere2></Seq2>. What is the movie/video game/book/music album in List2 but not in List1?

2. A person has watched/played/read/listened to a series of movies/video games/books/music albums. The watching/gaming/reading/listening list can be represented as List1: <Seq1><SeqHere1></Seq1>. Then the person watched/played/read /listened to one more movie/video game/book/music album, and the watching/gaming/reading/listening list can further be represented as List2: <Seq2><SeqHere2></Seq2>. What is the movie/video game/book/music album in List2 but not in List1?

3. A user's movie/video game/book/music album history can be represented as List1: <Seq1><SeqHere1></Seq1>. Later, after watching/playing/reading/listening to an additional movie/video game/book/music album, their history is represented as List2: <Seq2><SeqHere2></Seq2>. Please identify the movie/video game/book/music album that was added.

4. Initially, a person's movie watching/video game playing/book reading/music album listening history is encoded as List1: <Seq1><SeqHere1></Seq1>. After some time, their updated history is encoded as List2: <Seq2><SeqHere2></Seq2>. Which movie/video game/book/music album was added to their history?

5. The representation List1: <Seq1><SeqHere1></Seq1> shows a user's movie/video game/book/music album history. The representation List2: <Seq2><SeqHere2></Seq2> shows their updated history after watching/playing/reading/listening to one more movie/video game/book/music album. What is this new addition?

6. A viewer's/gamer's/reader's/listener's history changed from List1: <Seq1><SeqHere1></Seq1> to List2: <Seq2><SeqHere2> </Seq2> after they watched/played/read/listened to another movie/video game/book/music album. Can you determine which movie/video game/book/music album was added?

7. Compare these two representations of a person's movie watching/video game playing/book reading/music album listening history: List1: <Seq1><SeqHere1></Seq1> and List2: <Seq2><SeqHere2></Seq2>. List2 includes one additional movie/video game/book/music album. What is it?

8. A cinephile/gaming enthusiast/bibliophile/music enthusiast expanded their collection from List1: <Seq1><SeqHere1></Seq1> to List2: <Seq2><SeqHere2></Seq2>. Which movie/video game/book/music album was newly added to their collection?

9. I have two representations of someone's movie/video game/book/music album history: List1: <Seq1><SeqHere1></Seq1> and an updated List2: <Seq2><SeqHere2></Seq2>. Please identify the movie/video game/book/music album that appears in List2 but not in List1.

10. The encoded history List1: <Seq1><SeqHere1></Seq1> was updated to List2: <Seq2><SeqHere2></Seq2> after a person watched/played/read/listened to an additional movie/video game/book/music album. What is this new movie/video game/book/music album they've experienced?

A.3 Recommendation Prompts

1. A person has watched/played/read/listened to a series of movies/video games/books/music albums. The watching/gaming /reading/listening list can be represented as: <SeqH>. Please suggest the next movie/video game/book/music album this person is likely to watch/play/read/listen to.

2. Based on this representation of a person's movie watching/video game playing/book reading/music album listening history: <SeqH>, what would be the most suitable next movie/video game/book/music album to recommend?

3. A user's movie/video game/book/music album preferences can be encoded as: <SeqH>. What movie/video game/book/music album would you recommend they watch/play/read/listen to next?

4. I have a representation <SeqH> of someone's movie/video game/book/music album history. Could you recommend one movie/video game/book/music album they might enjoy watching/playing/reading/listening to next?

5. The following representation <SeqH> describes a viewer's/gamer's/reader's/listener's history. What movie/video game/book/music album do you think would be their ideal next choice?

6. Given this encoded pattern of a person's movie watching/video game playing/book reading/music album listening behavior: <SeqH>, what would be your top recommendation for their next movie/video game/book/music album?

7. A cinephile's/gaming enthusiast's/bibliophile's/music enthusiast's taste can be represented as: <SeqH>. Based on this, what movie/video game/book/music album should they experience next?

8. The sequence <SeqH> represents a person's movie/video game/book/music album preferences. What would be the most appropriate next movie/video game/book/music album to recommend to them?

9. Looking at this user's movie/video game/book/music album history representation: <SeqH>, which movie/video game/book/music album would complement their taste and be a good next recommendation?

10. A person with this movie watching/video game playing/book reading/music album listening pattern <SeqH> is looking for their next movie/video game/book/music album. What would you suggest they watch/play/read/listen to next?

B SEQUENCE RECOVERY CASES

Table 10. Examples of sequence recovery cases on Movie and Steam datasets.

Input Prompt	**Target Response**
A person has watched a series of movies. The watching list can be represented as: ⟨SeqH⟩. Describe this watching history of the person in detail.	This user has watched **Fargo**, **Jerry Maguire**, Antonia's Line, **Secrets and Lies**, Swingers, **Leaving Las Vegas**, **Emma**, Donnie Brasco, Cold Comfort Farm.
A user has viewed several movies. Their viewing history can be represented as: ⟨SeqH⟩. Could you list all the movies this user has watched?	This user has watched **Peacemaker**, **Conspiracy Theory**, Picture Perfect, **Mimic**, **Air Force One**, **Devil's Own**, **Murder at 1600**, Hoodlum, Beautician and the Beast, Kull the Conqueror.
The sequence ⟨SeqH⟩ represents a person's movie watching history. Can you translate this into a list of movie titles they've watched?	This user has watched **Abyss**, **Alien 3**, **Cat People**, **Nightmare on Elm Street**, **Cape Fear**, Body Snatchers, **Omen**, Jaws, **Shining**.
A person has played a series of video games. The gaming list can be represented as: ⟨SeqH⟩. Describe this gaming history of the person in detail.	This user has played **Don't Starve Together**, Half-Life, **The Binding of Isaac: Rebirth**, LEGO The Hobbit, **ShellShock Live**, Unturned, **Left 4 Dead**, Fallout 3: Game of the Year Edition, **Fallout 4**.
A user has played several video games. Their gaming history can be represented as: ⟨SeqH⟩. Could you list all the games this user has played?	This user has played Grand Theft Auto IV, **Source Filmmaker**, WAD Creator, **Brawlhalla**, Blockworld E2, **AdVenture Capitalist**, **Garry's Mod**, Robocraft, **Trove**.
The sequence ⟨SeqH⟩ represents a person's video game playing history. Can you translate this into a list of game titles they've played?	This user has played Getting Over It with Bennett Foddy, **Unturned**, Goliath, **Trove**, America's Army: Proving Grounds, **The Elder Scrolls V: Skyrim Special Edition**, AdVenture Capitalist, **DARK SOULS III**, Paladins.

Table 11. Examples of sequence recovery cases on Book and CD datasets.

Input Prompt	Target Response
A person has read a series of books. The reading list can be represented as: ⟨SeqH⟩. Describe this reading history of the person in detail.	This user has read **Six of Crows**, The Dream Thieves, **Heir of Fire**, **Queen of Shadows**, Blue Lily, Lily Blue, **Crooked Kingdom**, The Raven King, **Empire of Storms**, A Court of Mist and Fury.
A user has read several books. Their reading history can be represented as: ⟨SeqH⟩. Could you list all the books this user has read?	This user has read City of Glass, **Inheritance**, The Power of Six, **Brisingr**, Babe: Pig in the City, **Eragon**, The Grand Inquisitor, The Buccaneer, Cornwall.
The sequence ⟨SeqH⟩ represents a person's book reading history. Can you translate this into a list of book titles they've read?	This user has read **City of Fallen Angels**, **City of Lost Souls**, **City of Heavenly Fire**, The Fault in Our Stars, **Lady Midnight**, Beautiful Creatures, Clockwork Angel, **Throne of Glass**, Mockingjay.
A person has listened to a series of music albums. The listening list can be represented as: ⟨SeqH⟩. Describe this listening history of the person in detail.	This user has listened to **Essential Borodin**, **Prokofiev: Symphony No. 6**, **Mahler: Symphony No. 7**, Mozart: Complete Piano Sonatas, **Vivaldi: Concertos**, **Shostakovich: Symphonies Nos. 4 & 10**, Beethoven: The Complete Symphony Collection, **Abbey Road Anniversary Deluxe**.
A user has listened to several music albums. Their listening history can be represented as: ⟨SeqH⟩. Could you list all the albums this user has listened to?	This user has listened to **Keep on Smiling / Dixie Rock**, Led Zeppelin IV, **The Byrds: Original Album Classics**, **Best Of Johnny Winter**, The Doors: Greatest Hits, **Blues Breakers With Eric Clapton**, The Rolling Stones: Hot Rocks.
The sequence ⟨SeqH⟩ represents a person's music album listening history. Can you translate this into a list of album titles they've listened to?	This user has listened to **Haydn: Complete Symphonies**, Beethoven: Symphony No. 9, **Mahler: Symphony No. 5**, **Mahler: Symphony No. 9**, Brahms: The Complete Symphonies, **Mahler: Symphony No. 10**, **Leonard Slatkin conducts Vaughan Williams**, Tchaikovsky: The Symphonies.