Large Language Model Can Interpret Latent Space of Sequential Recommender

Anonymous ACL submission

Abstract

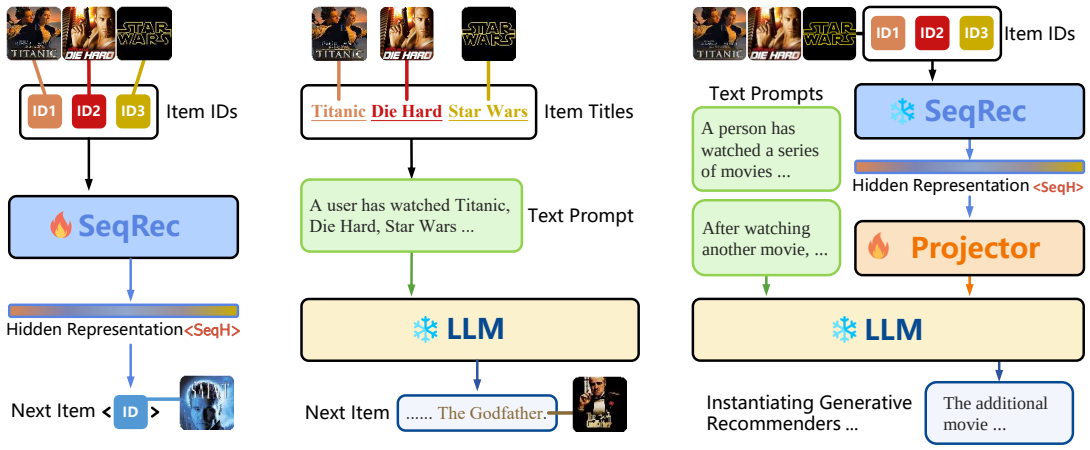
Sequential recommendation aims to predict the next item of interest for a user, based on her/his interaction history. In conventional sequential recommenders, a common approach is to model item sequences using discrete IDs, learning behavioral pattern representations. Inspired by recent success in empowering large language models (LLMs) to understand diverse modality data (*e.g.,* image, audio, 3D points), a compelling research question arises: "Can LLMs understand and utilize behavioral pattern representations from ID-based sequential recommenders?". To answer this, we propose a simple framework, RecInterpreter, which examines the capacity of open-source LLMs to decipher the representation space of sequential recommenders. Specifically, with the multi-modal 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, which encourages the LLM to generate textual narrations for items within the interaction sequence. Moreover, upon successfully interpreting sequential recommenders, the LLM can enhance its recommendation performance by tuning with the projected behavioral patterns. Empirical results showcase that our RecInterpreter enhances LLM to understand hidden representations from ID-based sequential recommenders and boost recommendation performance.

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 (Hidasi et al., 2016; Kang and McAuley, 2018; Covington et al., 2016). Scrutinizing leading sequential recommenders (Hidasi et al., 2016; Tang and Wang, 2018; Kang and McAuley, 2018; Yang et al., 2023; Beutel et al., 2018), we can summarize a typical pipeline: 1) assign discrete IDs to items and initialize learnable vectors (*aka.* item embeddings) to represent items, 2) learn the hidden representation of user behavioral pattern from each sequence of item embeddings, and 3) predict the next items that users are likely to consume based on the hidden representations, as Figure 1(a) shows. Such representations, derived from the ID-modeling paradigm, are able to encode the behavioral patterns of users, thus greatly facilitating the next-item recommendation.

With the meteoric rise of Large Language Models (LLMs) (*e.g.,* GPT4 (OpenAI, 2023a), LLaMA (Touvron et al., 2023a)), aligning diverse modalities with text can empower LLM to understand and reason about other modalities (Alayrac et al., 2022; Zhu et al., 2023; Ghosal et al., 2023; Maaz et al., 2023; Driess et al., 2023; Hong et al., 2023; Guo et al., 2023). Central to such an alignment is transforming the hidden representations from the modality-specific encoders (*e.g.,* images encoded by ViT (Dosovitskiy et al., 2021) or Stable Diffusion (Rombach et al., 2022)) into the text token embeddings of an LLM (Zhu et al., 2023; Ghosal et al., 2023). 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 (Geng et al., 2022; Lin et al., 2023a; Wu et al., 2023b; Fan et al., 2023; Ai et al., 2023; Bao et al., 2023b; Gao et al., 2023; Yuan et al., 2023; Li et al., 2023c) primarily focus on recasting the user-item interactions as text prompts and feeding them into LLM for recommendation or reranking, as Figure 1(b) illustrates. However, this paradigm shields LLMs from accessing or deciphering the hidden representations from recommender models.

Naturally, an compelling research question



(a) Sequential Recommendation (b) LLM4Rec (c) RecInterpreter

Figure 1: Comparision of conventional sequential recommendation, LLM4Rec and RecInterpreter. Flame 🔥 denotes tunable modules, while snowflake ❄ indicates frozen modules.

arises: "**Can LLM Understand and Utilize Representations from 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. Here we select LLaMA2-7B (Touvron et al., 2023b) as a prime example of open-source LLMs, which offer access to its hidden states and support backpropagation. In terms of sequential recommenders, we harness the representative models trained solely on item ID sequences, including GRU4Rec (Hidasi et al., 2016), Caser (Tang and Wang, 2018), and SASRec (Kang and McAuley, 2018). Having the LLM and recommender frozen, one straightforward solution to bridge their gap is the alignment training (Alayrac et al., 2022; Zhu et al., 2023; Moon et al., 2023) with the paired multimodal data (*i.e.,* representations of item ID sequences and text narrations). Following the leading alignment strategies (Zhu et al., 2023; Driess et al., 2023; Moon et al., 2023), RecInterpreter has two key components: 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.

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).

To empower recommendation with RecInterpreter, we further tune LLM with LoRA (Hu et al., 2022) for recommendation, while keeping the sequential recommenders and the projection layer frozen. Note that we involve no textual narration of interactions (such as title sequence used in (Liao et al., 2023; Zhang et al., 2023d)), and only use the projected behavioral pattern representation to describe user history. Superior recommendation performance highlights the effectiveness of explicitly aligning sequential recommenders with LLM under the RecInterpreter framework.

2 Related Work

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

2.1 Multimodal Language Models

Recent advances in LLM have demonstrated remarkable few/zero-shot reasoning capabilities in Neural Language Processing tasks (OpenAI, 2023a;

Peng et al., 2023; Zhao et al., 2023). Meanwhile, different modalities (including vision, video, audio and *etc.*), have been evolving their models rapidly to better accommodate different tasks (Dosovitskiy et al., 2021; Rombach et al., 2022; Kong et al., 2020). 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 (Alayrac et al., 2022; Zhu et al., 2023; Ghosal et al., 2023; Maaz et al., 2023; Wu et al., 2023c).

Along this research line, the pioneer work, Flamingao (Alayrac et al., 2022) demonstrates that the vision encoder NFNet (Brock et al., 2021) 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 (OpenAI, 2023b). MiniGPT4 (Zhu et al., 2023) further shows that a single linear layer is enough to make LLaMA (Touvron et al., 2023a) interpret hidden representations encoded by ViT (Dosovitskiy et al., 2021). Similarly, TANGO (Ghosal et al., 2023) suggests that the audio backbone model HiFiGAN (Kong et al., 2020) can be unified by LLM. Besides, Video-ChatGPT (Maaz et al., 2023) and VideoChat (Li et al., 2023b) imply that video encoders are also perceivable for LLM.

2.2 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, such as Recurrent Neural Network (RNN) (Hidasi et al., 2016), Convolutional Neural Network (CNN) (Tang and Wang, 2018), and Transformer encoder (Kang and McAuley, 2018). Moreover, recent work also designs auxiliary learning tasks like causal inference (Zhang et al., 2021, 2023a), data augmentation (Xie et al., 2022), and robust learning (Yang et al., 2023).

Advances in LLM have drawn increasing research attention to leveraging LLM for sequential recommendation. Some research directly employs in-context learning to assess the recommendation performance of LLM or enhance traditional recommendations, thereby measuring the recommendation capabilities of LLM (Hou et al., 2023; Zhang et al., 2023b; He et al., 2023; Liu et al., 2023; Yuan et al., 2023). Furthermore, some recent studies (Bao et al., 2023b,a) are concerned that LLM lacks recommendation-specific knowledge in the pre-training phase, thus proposing to leverage LLM to implement specific tuning techniques to enhance their recommendation performance (Wu et al., 2023a; Zhang et al., 2023c; Lin et al., 2023b; Bao et al., 2023a,b). Besides, several studies have proposed to integrate pre-trained recommenders for tuning LLM, which has shown effectiveness in enhancing the performance of LLM for recommendation tasks (Li et al., 2023c; Zhang et al., 2023d; Liao et al., 2023). However, these approaches mostly shield LLM from accessing or deciphering the hidden representations from frozen recommender models. Therefore, inspiring LLM to interpret hidden representations from sequential recommenders and improve recommendation accordingly remains largely unexplored.

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. Taking a step further, we propose a novel scheme to tune LLM for recommendation based on the sequence-recovery prompting. Finally, we discuss the difference between RecInterpreter and some recent preprint studies.

3.1 Sequence-Recovery Prompting

To validate LLM's capability in understanding sequential recommenders, we draw inspiration from prior multi-modal alignment studies (Alayrac et al., 2022; Zhu et al., 2023; Li et al., 2023b; Driess et al., 2023) and propose a sequence recovery task. This objective is to encourage LLM to explicitly reconstruct the items in an interaction sequence with textual narrations, based solely on its hidden representation acquired from conventional sequential recommenders, as depicted in Figure 2. Next, we elaborate on our steps.

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 $\mathbf{s}$ is first fed into the sequence encoder, and

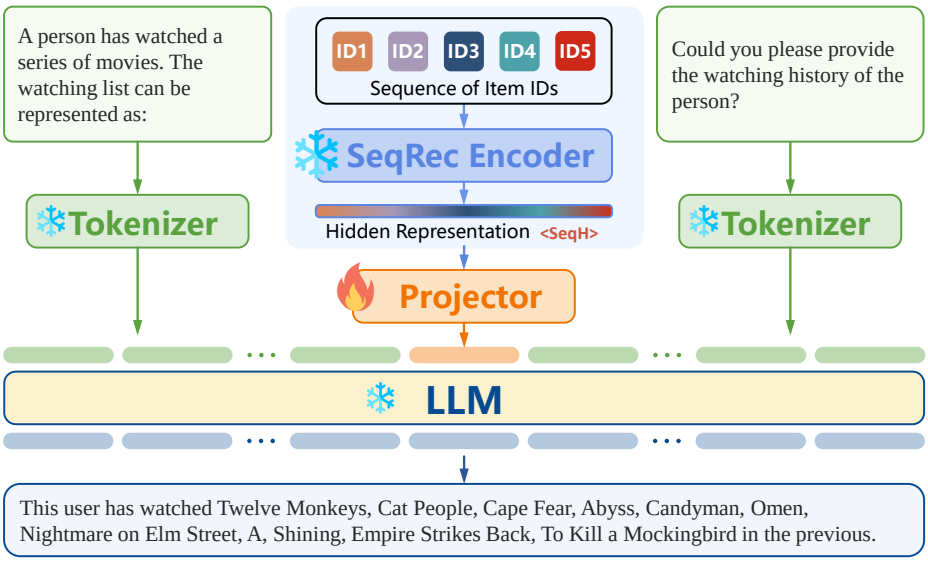


Figure 2: Illustration of the sequence recovery framework. We provide the textual prompts and the hidden representation of the interaction sequence projected by an MLP layer, targeting at inspiring LLM to recover the interactions with a textual response. Flame 🔥 denotes tunable modules, while snowflake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

we can acquire the hidden representation through:

$$\mathbf{h_s} = \textbf{Seq-Enc}(\mathbf{E_s}), \tag{1}$$

where **Seq-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:

$$\tilde{\mathbf{h}}_\mathbf{s} = \textbf{Proj}_\theta(\mathbf{h_s}). \tag{2}$$

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 (Li et al., 2023a), 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 $\mathbf{X}$ (Touvron et al., 2023a):

$$\max_\theta \sum_{i=1}^{N} \log P(\mathbf{Y}^i | \mathbf{X}, \mathbf{Y}^{[1:i-1]}), \tag{3}$$

where θ denotes the parameters of the projection layer $\textbf{Proj}_\theta(\cdot)$, 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 RecInterpreter for Recommendation

After training the projector, LLM is able to explicitly interpret the hidden representation of behavioral patterns encoded by sequential recommenders. 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 with LoRA.

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. Here we employ a reranking task to prompt LLM, which aims to select the ground-truth next items from the candidate set. 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 from the provided options: Get Shorty, Gone with the Wind, Candyman, . . . , The Saint.
Target Response	This person prefers Gone with the Wind among the options.

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

Then we adopt LoRA (Hu et al., 2022) (please refer to Appendix A.4 for detailed information of LoRA) to tune LLM for recommendation, the objective can be formulated as:

$$\max_{\Theta} \sum_{i=1}^{N} \log P_{\Phi_0+\Delta\Phi(\Theta)} \left(\mathbf{Y}^i | \mathbf{X}, \mathbf{Y}^{[1:i-1]} \right), \quad (4)$$

where Φ_0 and Θ 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.

3.3 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 (Li et al., 2023c; Zhang et al., 2023d; Liao et al., 2023) (arxiv paper, not published yet). 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. In contrast, RecInterpreter first 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. 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. These two differences set RecInterpreter distinctly apart from prior studies.

4 Experiment

In this section, we conduct experiments to demonstrate our approach to explicitly inspire LLM to understand sequential recommenders through interpreting the hidden representations. Then we show how to facilitate recommendation performance with the RecInterpreter framework.

4.1 Experimental Setings

4.1.1 Datasets

We use three datasets from real-world recommendation scenarios: Movie, Steam, and Book. The statistics of datasets are illustrated in Table 1. The detailed information about the datasets is demonstrated in Appendix A.1.

4.1.2 Implementation Details

The implementation details are shown in A.3.

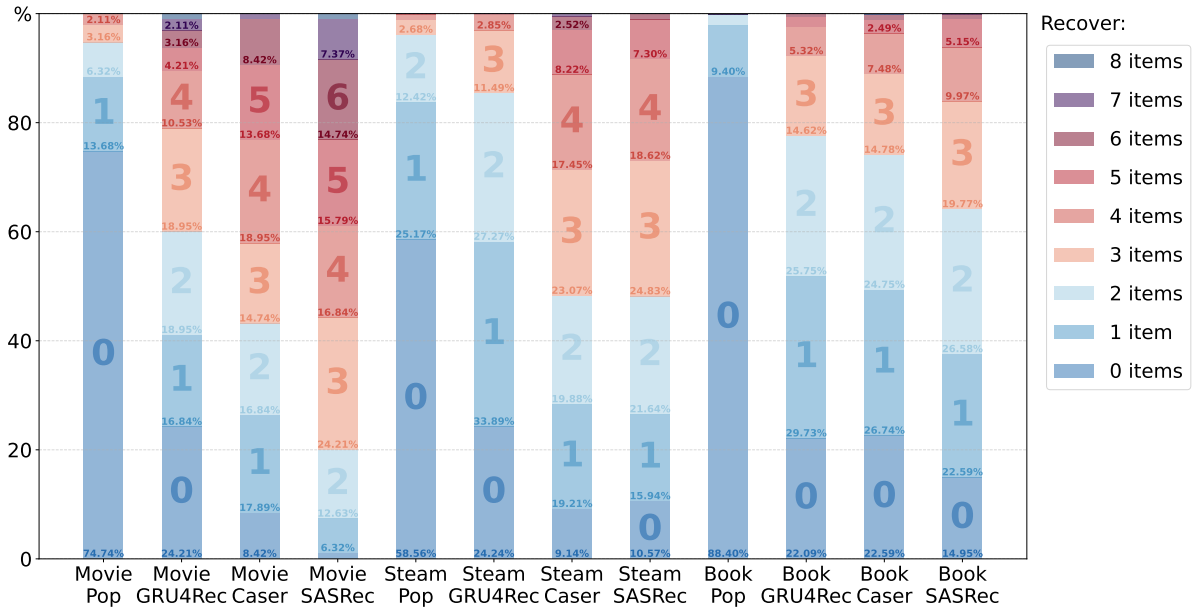


Figure 3: The distribution of the number of recovered items on the Movie, Steam, and Book datasets, with GRU4Rec, Caser, and SASRec as the sequential recommenders. 'Pop' denotes that the 10 most popular movies or games in the training data are provided as the sequence recovery results. The average number of items in the test sequences is 10.00 for the Movie dataset, 8.89 for the Steam dataset, and 9.88 for the Book dataset.

Table 1: Statistics of datasets.

Dataset	Movie	Steam	Book
#sequences	943	11,938	6,031
#items	1,682	3,581	4,500
#interactions	100,000	274,726	220,100

4.1.3 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.

- To evaluate the recommendation performance, we adopt a re-ranking task following prior study (Liao et al., 2023). Specifically, for each sequence, we randomly select 20 non-interacted items to construct the candidate set, ensuring the inclusion of the correct subsequent item, and we evaluate whether they could identify the correct item from this candidate set, and the performance is evaluated using the HitRatio (HR) metric. With appropriate prompting, LLM-based recommenders can generate a single candidate item as required.

4.2 Sequence-Recovery Result

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. 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 hidden representations of the sequential recommenders. It is worth noting that sequence recovery is not a trivial task, since each dataset contains thousands of items and the hidden representation is highly compressed. We illustrate the result in Figure 3, and we present the observed cases from the Movie and Steam datasets in the inference phase in Figure 4.

From Figure 3 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.

- In general, with our designed sequence recovery framework, LLLM 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.

- In the comparison of the Movie, Steam, and Book datasets, we can observe that LLM shows a better understanding of the Movie data than the Steam and Book data. One reason is that the item titles in the Steam and Book 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.

- 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 (He et al., 2016). The Transformer encoder employed by SASRec is one of the widely adopted sequence-to-sequence architectures (Vaswani et al., 2017), which is also the fundamental component in the framework of LLM (Touvron et al., 2023a). However, RNN may suffer from issues such as vanishing gradient and exploding gradient (Pascanu et al., 2013), thus presenting obstacles for LLM to well understand its hidden representations.

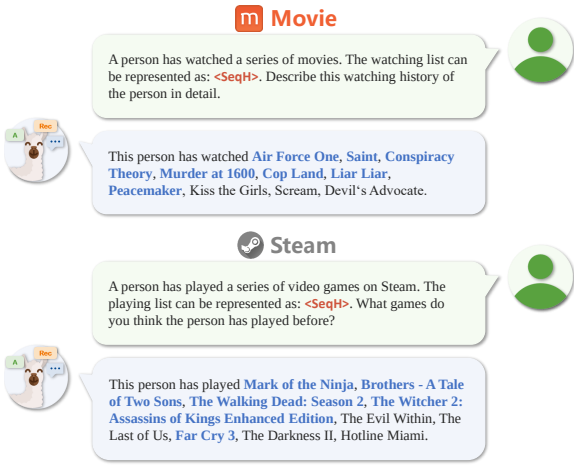


Figure 4: Two cases of the sequence recovery task in Movie and Steam datasets. <SeqH> denotes the hidden representation of the interaction sequence after the projection layer. The blue text in the response denotes the correctly recovered movies and games only from the hidden representation.

4.3 Recommendation Result

In this section, we compare the recommendation performance of RecInterpreter with several baseline methods, including: 1) Representative traditional sequential recommenders: GRU4Rec (Hidasi et al., 2016), Caser (Tang and Wang, 2018), and SASRec (Kang and McAuley, 2018). 2) Recent LLM-based recommenders: ChatRec (Gao et al., 2023), MoRec (Yuan et al., 2023), TallRec (Bao et al., 2023b), and LLaRA (Liao et al., 2023). More detailed information about these baselines is shown in Appendix A.2.

The results are shown in Table 2, from which we can observe that:

- Generally, RecInterpreter achieves the best recommendation performance compared to all baselines. This performance improvement can be attributed to the superiority of RecInterpreter, which explicitly inspires LLM to interpret hidden behavioral pattern representation with textual description with our designed sequence-recovery prompting. This explicit prompt can empower LLM to comprehensively understand the behavioral patterns encoded by sequential recommenders, which thereafter boosts recommendation performance when tuning LLM with LoRA.

- When combined with different sequential recommenders, RecInterpreter displays different recommendation performances. Specifically, RecInterpreter tends to achieve better recommendation performance when combined with Caser and

Table 2: Recommendation performance comparison between RecInterpreter and baselines on three datasets. Bold indicates the best performance and the underline indicates the best among baselines. RecInterpreter_g, RecInterpreter_c, and RecInterpreter_s denote RecInterpreter with GRU4Rec, Caser, and SASRec as sequential recommenders respectively. The experiments are conducted 3 times and the average is reported.

	Movie	Steam	Book
	HR	HR	HR
GRU4Rec	0.3750	0.4535	0.3867
Caser	0.3861	0.4885	0.4174
SASRec	0.3444	0.5282	0.3581
ChatRec	0.2000	0.2039	0.3306
MoRec	0.2822	0.3956	0.2877
TallRec	0.3895	0.6787	0.4983
LLaRA	<u>0.4737</u>	<u>0.7139</u>	<u>0.5282</u>
RecInterpreter_g	0.6737	0.7407	0.5681
RecInterpreter_c	**0.7157**	**0.7625**	0.5415
RecInterpreter_s	0.7073	0.7482	**0.5782**

SASRec than GRU4Rec. The reason could be similar to the performance of sequence recovery, *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 LLM-based recommenders, we find that in-context learning approaches typically perform not well enough, since the pre-training of LLM involves few recommendation-related tasks or corpus (Bao et al., 2023b; Liao et al., 2023). Therefore, it is challenging for LLM to perform recommendation tasks without fine-tuning. The performance of MoRec is somehow unsatisfactory, likely because we only use item titles to construct representations, for fair comparison with other approaches. Additional modality information like user profiles and item images may be necessary for MoRec to achieve its full potential(Yuan et al., 2023). Furthermore, LLaRA's superior performance over TallRec 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 tuning-based approaches (TallRec, LLaRA, and RecInterpreter), we discover that RecInterpreter outperforms the other two methods. To be specific to their differences, we could discern that TallRec utilizes the textual information (*i.e.,* title sequence) to describe user historical behavior in its prompt design. LLaRA integrates both the textual information and mapped hidden representations obtained from sequential recommenders to describe user historical behavior in the prompt. However, LLaRA implicitly incorporates the hidden representations with LLM by only optimizing the recommendation objective, such as Equation 4. In contrast, RecInterpreter explicitly allows LLM to understand the hidden representations by the proposed sequence-recovery prompting, which enables LLM to discard the textual information, and only leverage the projected hidden representations to describe user behavior history. The performance improvement verifies the rationale and superiority of RecInterpreter.

5 Conclusion and Limitations

We propose RecInterpreter, inspiring LLMs to understand conventional sequential recommenders. RecInterpreter draws inspiration from recent advances in multi-modal language models, that hidden representation of modality-specific encoders could be perceived by LLMs through simple projection. Therefore, RecInterpreter designs the sequence-recovery prompting, allowing LLM to understand the hidden representation of sequential recommenders. Besides, RecInterpreter provides a novel scheme to tune LLM for recommendation based on the sequence-recovery prompting.

Meanwhile, RecInterpreter also has limitations: 1) the projection is simply set as an MLP layer; and 2) the size of the datasets is not large enough. We believe these can be resolved in further research with more advanced adapters such as Q-former (Li et al., 2023a), and larger datasets. Moreover, as an initial attempt to inspire LLMs to explicitly understand recommendation modality, RecInterpreter provides many research opportunities. For example, online service providers could apply RecInterpreter to their recommenders, and explore the application of LLMs from this perspective. Besides, researchers can design other frameworks other than the sequence-recovery, to better guide LLMs to understand recommendation modality.

Ethical Impact

The proposed RecInterpreter can interpret conventional recommenders and significantly improve the performance of recommendation. Therefore, it can be applied to real-world platforms to provide more satisfying recommendation results. One concern of interpreting existing recommenders is the potential for privacy disclosure. Therefore, we kindly advise researchers to be cautious about the usage of RecInterpreter in real-world applications.

References

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*, 4:80–90.

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*.

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnan He, and Qi Tian. 2023a. A bi-step grounding paradigm for large language models in recommendation systems. *CoRR*, abs/2308.08434.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023b. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, pages 1007–1014. ACM.

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*, pages 46–54.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. 2021. High-performance large-scale image recognition without normalization. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 1059–1071.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *CoRR*, abs/2305.14314.

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.

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*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488.

Ali Edalati, Marzieh S. Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *CoRR*, abs/2212.10650.

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*.

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.

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*, pages 299–315. ACM.

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.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

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*.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

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.

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*.

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.

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):75:1–75:27.

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

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *NeurIPS*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023a. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR.

Kunchang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023b. Videochat: Chat-centric video understanding. *CoRR*, abs/2305.06355.

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

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, and Xiang Wang. 2023. Llara: Aligning large language models with sequential recommenders. *CoRR*, abs/2312.02445.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2023a. How can recommender systems benefit from large language models: A survey. *arXiv preprint arXiv:2306.05817*.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2023b. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. *arXiv preprint arXiv:2308.11131*.

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.

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.

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.

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.

OpenAI. 2023a. GPT-4 technical report. *CoRR*, abs/2303.08774.

OpenAI. 2023b. Gpt-4v(ision) system card.

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

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. *CoRR*, abs/2304.03277.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10674–10685. IEEE.

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

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. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

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. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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*, pages 5998–6008.

Likang Wu, Zhaopeng Qiu, Zhi Zheng, Hengshu Zhu, and Enhong Chen. 2023a. Exploring large language model for graph data understanding in online job recommendations. *arXiv preprint arXiv:2307.05722*.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023b. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860*.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2023c. Next-gpt: Any-to-any multimodal LLM. *CoRR*, abs/2309.05519.

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*, pages 1259–1273.

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*, pages 331–340. ACM.

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*, pages 2639–2649. ACM.

An Zhang, Fangfu Liu, Wenchang Ma, Zhibo Cai, Xiang Wang, and Tat-Seng Chua. 2023a. Boosting differentiable causal discovery via adaptive sample reweighting. *CoRR*, abs/2303.03187.

Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023b. Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation. In *RecSys*, pages 993–999. ACM.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023c. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001*.

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*, pages 11–20.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2023d. Collm: Integrating collaborative embeddings into large language models for recommendation. *CoRR*, abs/2310.19488.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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.

A Appendix

A.1 Datasets

- **Movie**[1] is a well-known dataset for movie recommendation, containing users' rating history. We preserve titles as the textual descriptions of movies.

- **Steam** (Kang and McAuley, 2018) dataset contains user reviews of video games on the Steam Store. The titles of video games are also available as textual descriptions.

- **Book** is collected from a social book cataloging website[2], where users can search, rate, and provide reviews of various books.

Since tuning the projection layer requires backpropagation from LLaMA, the training phase is more time-consuming than conventional recommenders, and the size of datasets should not be too large. Therefore, we select the MovieLens100K dataset in our experiment. For the Steam dataset, we first remove users who have less than 20 reviews, which keeps the same as the processing of MovieLens. Then we sample 1/3 of users and 1/3 of games and preserve their interactions to acquire a moderate size of dataset. For the Book dataset, we remove users and books with less than 20 interactions.

For all datasets, we first sort all sequences in chronological order and then split the data into training, validation, and testing data at the ratio of 8:1:1. This splitting strategy ensures that later

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

interactions would not appear in the training set, avoiding any potential of information leakage (Ji et al., 2023).

A.2 Baselines

- **GRU4Rec** (Hidasi et al., 2016) an RNN-based sequential recommender, which leverages GRU to encode users' interaction sequences.

- **Caser** (Tang and Wang, 2018) is a CNN-based sequential recommender. We apply one vertical filter and 16 horizontal filters with heights {2, 3, 4}.

- **SASRec** (Kang and McAuley, 2018) a self-attention based sequential recommender that captures long-range dependencies in user-item interactions.

- **ChatRec** is adapted from (Gao et al., 2023). 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.

- **MoRec** (Yuan et al., 2023) 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.

- **TallRec** (Bao et al., 2023b) constructs recommendation corpus by transferring interaction sequences into textual prompts, and tunes LLMs for recommendation with the corpus of specific domains.

- **LLaRA** (Liao et al., 2023) 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.

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.

- **LLM-based recommenders** can also be divided into several subcategories. We select **ChatRec** as the representative in-context learning based method, **TallRec** as the representative prompt tuning based method, and **LLaRA** as the representative modality alignment based method.

A.3 Implementation details

We implement all approaches with Python 3.10, PyTorch 2.0.0, and transformers 4.32.0 in a single Nvidia A40. 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 (Hidasi et al., 2016), Caser (Tang and Wang, 2018), and SASRec (Kang and McAuley, 2018) on the training datasets. We use Adam optimizer, the learning rate is tuned as 0.001 and the batch size is set as 256. We adopt L2 regularization for all models and the coefficient is searched in [1e-3, 1e-4, 1e-5, 1e-6, 1e-7]. The embedding size is searched in [16, 64, 256, 1024]. The sequential recommenders would be frozen after training.

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 (Tang and Wang, 2018), 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 $L \times D$ (Hidasi et al., 2016; Kang and McAuley, 2018). 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 two-layer MLP with GELU activation as the projector. To train the projector, we select AdamW as the optimizer, and adopt a warmup learning rate schedule: the learning is set as 0.0001 at the 1st epoch, increases linearly to 0.0005 at the 5th epoch, and remains unchanged. We search the L2 regularization coeffi-

cient in the range of [1e-4, 1e-5, 1e-6]. We select LLaMA2-7B (Touvron et al., 2023b) as the LLM in our experiment. For the sequence recovery framework, we set the maximum generated token length as 50, since each hidden representation contains several items. Besides, constraining the generation length also prevents LLM from generating too many tokens that may include nonsense output. It takes about 2 hours, 6 hours, and 5 hours to train the model for a single epoch in the Movie, Steam, and Book datasets respectively. And training for 20 epochs is generally enough for convergence.

In the LoRA tuning phase for recommendation, we adopt AdamW as the optimizer, search the learning rate in [1e-4, 1e-5, 1e-6], search the L2 coefficient in [1e-6, 1e-7, 1e-8], and set the rank of LoRA as 8.

The experiments are conducted 3 times and the average is reported.

A.4 Introduction of LoRA

Fine-tuning all parameters of pre-trained LLMs on task-specific data is the primary approach for adapting LLMs to new tasks. However, this full fine-tuning is extremely time and resource intensive. 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 (Hu et al., 2022; Liu et al., 2021; Dettmers et al., 2023; Edalati et al., 2022). Among such methods, Low-Rank Adaptation (LoRA) has gained significant attention (Hu et al., 2022). LoRA injects trainable low-rank decomposition matrices into each layer of the LLM's Transformer architecture. 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 LoRA objective is:

$$\max_{\Theta} \sum_{i=1}^{N} \log P_{\Phi_0+\Delta\Phi(\Theta)} \left(\mathbf{Y}^i | \mathbf{X}, \mathbf{Y}^{[1:i-1]} \right), \quad (5)$$

where N is the number of tokens in $\mathbf{Y}$, $\mathbf{Y}^i$ is the ith token, and Θ denotes the small set of LoRA parameters introduced in the decomposition matrices $\Delta\Phi(\Theta)$. In contrast, Φ_0 represents the large frozen base LLM parameters. Thus, only Θ is updated via backpropagation during LoRA training.