DELRec: Distilling Sequential Pattern to Enhance LLMs-based Sequential Recommendation

Haoyi Zhang[1,†], Guohao Sun[1,†], Jinhu Lu[1], Guanfeng Liu[2], Xiu Susie Fang[1,‡]
[1]Donghua University, Shanghai, 201620, China
[2]Department of Computing, Macquarie University, Sydney, NSW 2109, Australia
2232980@mail.dhu.edu.cn, ghsun@dhu.edu.cn, 1209126@mail.dhu.edu.cn,
guanfeng.liu@mq.edu.au, xiu.fang@dhu.edu.cn

***Abstract*—Sequential recommendation (SR) tasks aim to predict users' next interaction by learning their behavior sequence and capturing the connection between users' past interactions and their changing preferences. Conventional SR models often focus solely on capturing sequential patterns within the training data, neglecting the broader context and semantic information embedded in item titles from external sources. This limits their predictive power and adaptability. Large language models (LLMs) have recently shown promise in SR tasks due to their advanced understanding capabilities and strong generalization abilities. Researchers have attempted to enhance LLMs-based recommendation performance by incorporating information from conventional SR models. However, previous approaches have encountered problems such as 1) limited textual information leading to poor recommendation performance, 2) incomplete understanding and utilization of conventional SR model information by LLMs, and 3) excessive complexity and low interpretability of LLMs-based methods.**

To improve the performance of LLMs-based SR, we propose a novel framework, Distilling Sequential Pattern to Enhance LLMs-based Sequential Recommendation (DELRec), which aims to extract knowledge from conventional SR models and enable LLMs to easily comprehend and utilize the extracted knowledge for more effective SRs. DELRec consists of two main stages: 1) *Distill Pattern from Conventional SR Models*, focusing on extracting behavioral patterns exhibited by conventional SR models using soft prompts through two well-designed strategies; 2) *LLMs-based Sequential Recommendation*, aiming to fine-tune LLMs to effectively use the distilled auxiliary information to perform SR tasks. Extensive experimental results conducted on four real datasets validate the effectiveness of the DELRec framework. The code is available at https://github.com/haoge6660101/DELRec_hao.

***Index Terms*—large language model, sequential recommendation, pattern distillation**

I. INTRODUCTION

Sequential recommendation [1–3] predicts users' next interaction by learning their behavior sequence. SR aims to improve the accuracy of recommendations by understanding and modeling the relationship between users' interaction history and their evolving preferences. Conventional SR models [4] only capture sequential patterns within training data, often overlooking the broader context and semantic information embedded in item titles that can be obtained from external sources. These limitations restrict their predictive ability and adaptability to constantly changing scenarios.

Large language models have recently shown promise in SR tasks due to their advanced comprehension abilities and powerful generalization capabilities [5, 6]. As LLMs are trained on vast datasets containing abundant information, including inherent item features and details, they can infer user preferences and predict future actions [7] by leveraging LLMs' understanding of item attributes and inferring based on world knowledge. However, using LLMs directly as sequential recommenders [8] can pose certain problems. For instance, due to a lack of domain-specific expertise [9] in the recommendation or an incomplete understanding of the recommendation patterns in SR tasks, LLMs often perform poorly when directly used as recommenders.

Thus, researchers have previously proposed integrating auxiliary information from conventional SR models into LLMs. These approaches aim to provide information from conventional SR models to LLMs to enhance the accuracy of SR tasks conducted by LLMs. The LLMs-based SR integrating conventional SR models can be roughly categorized into three paradigms. These three paradigms are illustrated in Figure 1, and the detailed introduction and issues caused by them are summarized below:

Providing LLMs with textual information extracted from conventional SR models. This paradigm typically involves incorporating the textual information extracted from conventional SR models into the prompt [10]. However, this paradigm often suffers from poor recommendation performance due to the limited information provided by the prompt. One fundamental reason is that textual information is often insufficient for accurately and comprehensively describing conventional SR models' specific recommendation information and behavior patterns.

Providing LLMs with embeddings from conventional SR models. This paradigm typically merges embeddings from conventional SR models with a prompt before inputting them into LLMs to generate item recommendations. This paradigm uses embeddings encoded by conventional SR models as auxiliary information for the recommendation process provided to LLMs and often involves a projector to align the dimensions of conventional SR model's embeddings with the language space

[†]Equal Contribution
[‡]Corresponding Author

of LLMs. However, due to poor projector design or changes in embedding dimensions [11] that result in information loss, LLMs may not fully comprehend the meanings conveyed by these embeddings.

Combining embeddings from LLMs and conventional SR models. Instead of using LLMs as recommenders, this paradigm utilizes LLMs' encoding and representation capabilities. It typically involves utilizing LLMs to encode a given text or sequence and simultaneously employing conventional models to obtain item or user encodings. Subsequently, these two types of embeddings are combined and processed in various ways to generate recommendation scores for items. Although this paradigm enables the integration of information from both conventional SR models and LLMs, it also introduces challenges in comprehending and interpreting recommendations. This may potentially undermine some key advantages of using LLMs for recommendations, such as their simplicity and interpretability.

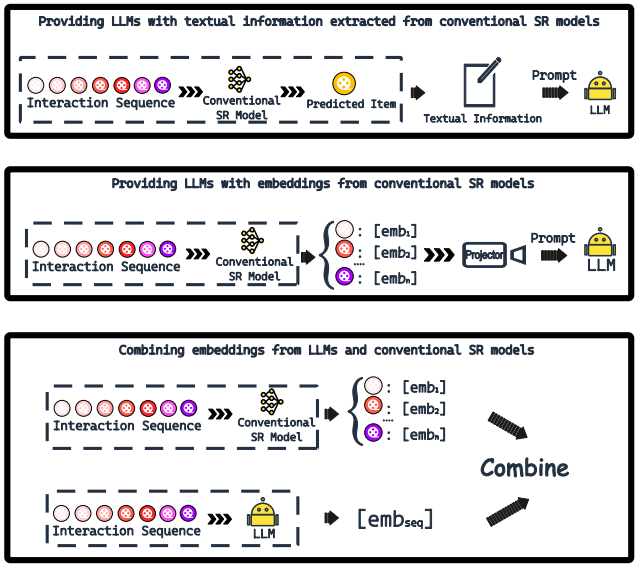

Fig. 1. Demonstration of three paradigms of the integration of conventional SR models with LLMs-based SR.

To tackle the aforementioned problems, we propose Distilling Sequential Pattern to Enhance LLMs-based Sequential Recommendation (DELRec), which aims to distill the behavioral patterns of conventional SR models and empower LLMs to easily comprehend and leverage this supplementary information for more effective SRs. The paradigm of DELRec is roughly shown in Figure 2, and it contains two stages:

Distill Pattern from Conventional SR Models. Rather than inputting encoded information from conventional SR models or LLMs as previous methods did, the first stage of DELRec (*Distill Pattern from Conventional SR Models*) is inspired by knowledge distillation [12] techniques used in LLMs. The objective is to distill conventional SR models' recommendation patterns and information that are understandable to LLMs. This stage involves using LLMs to extract useful knowledge of conventional SR models into soft prompts [13]. Specifically, this stage consists of two key components: *Temporal Analysis* (TA) and *Recommendation Pattern Simulating* (RPS). Through *Temporal Analysis*, LLMs are able to simulate conventional SR models for temporal analysis, thereby gaining temporal awareness. *Recommendation Pattern Simulating* allows LLMs to learn the recommendation results of conventional SR models, enabling them to simulate the recommendation behavior patterns of conventional SR models. During this stage, LLMs conduct tasks of these two components concurrently through multi-task learning and adjust the parameters of the soft prompts. Consequently, LLMs are empowered to effectively comprehend and simulate the recommendation process employed by conventional SR models. This is a process of transforming the knowledge of conventional SR models into a form that LLMs can fully utilize.

LLMs-based Sequential Recommendation. After getting the distilled SR knowledge in the first stage for SR tasks, we propose *LLMs-based Sequential Recommendation* for effectively instructing LLMs. Instead of using a projector for embedding mapping, we insert the learned soft prompts into the prompt to avoid information inefficiency or loss caused by using a projector to align embedding dimensions and then fine-tune the LLMs to adapt to the learning tasks that utilize auxiliary information.

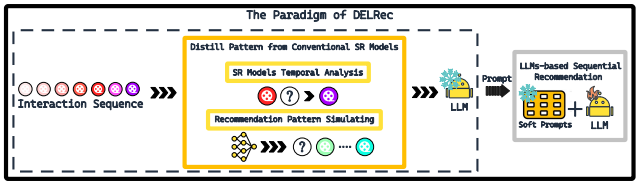

Fig. 2. Demonstration of the paradigm of DELRec.

The main contributions of our work are summarized as follows:

- Proposing a novel framework, DELRec, to enhance the LLMs-based SR performance with conventional SR models.
- Proposing two novel components in the first stage of DELRec to distill the information and behavior patterns of conventional SR models with soft prompts.
- Designing an ingenious method in the second stage of DELRec to fine-tune LLMs, enabling them to utilize the auxiliary information and achieve more accurate SR tasks.
- Conducting extensive experiments to demonstrate the effectiveness of DELRec.

II. RELATED WORK

This section presents a literature review on Conventional Sequential Recommendation and LLMs-based Sequential Recommendation.

A. Conventional Sequential Recommendation

In the field of SR, recognizing the sequential pattern of user interactions is crucial for predicting their next preference.

For early research of conventional SR models, Markov chain models are often used for SR tasks. It mainly relies on probability theory and statistics to model state transitions

in sequential data. FPMC [14] combines the ideas of Markov chains and matrix factorization. Fossil [15] integrates an item similarity model with a high-order Markov chain, leveraging both the item matrix and user historical behavior.

With the continuous development of deep learning, more and more conventional SR models also employ various deep learning techniques such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), or Transformers to identify sequential patterns in user interactions.

GRU4Rec [16] uses Gated Recurrent Units (GRUs), a type of RNN, to analyze users' interactions. It predicts users' next actions by effectively capturing their long-term dependencies during a session. Caser [17] uses CNNs to process multi-dimensional interaction data, where interactions between users and items are encoded as a two-dimensional matrix, with each element representing a specific interaction. SASRec [18] utilizes an attention mechanism to process interaction data, where the behavior of users is encoded as vectors and serves as inputs for the attention model. TIGER [19] builds semantically meaningful codeword tuples as the semantic ID for each item, and trains a transformer-based model to predict the target item.

Although these methods are good at capturing sequential patterns, they fail to effectively utilize the semantic information contained within item titles.

B. LLMs-based Sequential Recommendation

With the ongoing development of LLMs, researchers are exploring methods to integrate conventional SR models with LLMs [20–22] in order to enhance the performance of SR tasks. We illustrate the existing methods according to the three paradigms summarized in the introduction section.

Providing LLMs with textual information extracted from conventional SR models. PISA [23] takes numerical sequences as input and generates numerical outputs. It converts input and output into prompts and predicts sentence by sentence using a LLM. RecRanker [24] uses importance-aware and clustering-based sampling to collect user data for training. It enhances performance by integrating recommendation results from conventional models into textual prompts. Llama4Rec [25] proposes customized data augmentation and prompt augmentation strategies to enhance both conventional models and LLMs. It uses an adaptive aggregation module to combine the textual results of the two models into prompts, refining the final recommendation outcome. LLMSEQPROMPT [26] improves SR accuracy by injecting domain knowledge into the prompts of LLMs. It uses the session containing the item list as a prompt and the last item name as completion, and then fine-tunes the LLMs. LLM-TRSR [27] segments users' interactions and generates recurrent summaries using LLMs, then it builds prompts that contain users' preference summaries, recent interactions, and candidate information, then finally uses prompts to fine-tune LLMs. Despite using textual information from conventional SR models to prompt LLMs, the limited information provided by textual information leads to poor performance of LLMs-based SR.

Providing LLMs with embeddings from conventional SR models. LLaRA [28] leverages multimodal mapping by inserting prompts with item embeddings encoded by conventional SR models, and then fine-tunes LLMs with item interaction relationships. LLMGR [29] uses a tokenizer to extract text embeddings and conventional models to extract graph embeddings. It also develops auxiliary and main instructions to adjust task prompts and effectively combine text and graph embeddings, thereby enhancing the effectiveness of SR. RLMRec [30] combines representation learning with LLMs to capture intricate semantic embeddings of user behaviors and preferences. RLMRec uses LLMs for user/item profiling to incorporate auxiliary textual signals, and it aligns these semantic embeddings with collaborative relational textual signals through cross-view alignment. LLM2BERT4Rec [26] initializes item embeddings in BERT4Rec [31] with LLMs embeddings, and reduces the dimensionality of LLMs embeddings using PCA [32] as projector before initialization. Then, it trains BERT4Rec with these embeddings using the BERT's [33] masked language model training protocol. However, these methods typically require a projector to align embedding dimensions and semantic space. Poorly designed projectors can lead to information loss or ineffective utilization by LLMs.

Combining embeddings from LLMs and conventional SR models. LLM-based Generation of Item-Description [34] utilizes LLMs to generate the description and then employs BERT to encode the movie's information along with its original movie ID. The combined embeddings obtained from these two steps are then input into a conventional recommendation model in order to obtain the recommended item. LlamaRec [35] employs a conventional model to generate embeddings for item recall, passes these embeddings to LLMs, and ultimately utilizes a verbalizer to transform the output logits into a candidate probability distribution. RA-Rec [36] first converts the users' browsing history sequences into natural language, then uses prompt-tuning to guide LLMs to encode them to the user embeddings, and finally combines them with the item embedding extracted from SR models to obtain scores for the predicted items. LLMSEQSIM [26] uses LLMs to obtain item embeddings, and it calculates session embeddings by combining embeddings of preference items within the session. Then, it computes similarity to recommend the most similar item to the user. Relation-aware SR with Latent Relation Discovery (LRD) [37] utilizes the LRD framework based on LLMs to predict the latent relationship between items, and it reconstructs item embeddings based on these relationships. Then, these latent relationships are integrated into the recommender models along with the embeddings extracted by conventional SR models. However, these methods may undermine key advantages of using LLMs for recommendations, such as simplicity and interpretability.

The proposed DELRec utilizes soft prompts to extract information and recommendation patterns from conventional SR models by carefully designing components. These extracted soft prompts are then inserted into the prompt, allowing for fine-tuning of LLMs to learn this auxiliary information.

DELRec effectively addresses previous problems and improves the performance of LLMs-based SR.

III. PRELIMINARY

A. Task Formulation

We consider a recommender system with a set of users U, where a user $u \in U$ has an interaction sequence that consists of a sequence of n items $(I_1, I_2, ..., I_n)$ in chronological order (n can be different for different users). The SR task is defined as follows: given the user interaction sequence $I_{1:n-1} = (I_1, I_2, ..., I_{n-1})$, a sequential recommender aims to predict the target item I_n from a set of candidate items C, where candidate set that consists of m items $(I_1, I_2, ..., I_m)$ is typically selected from the entire item set I, where $m \ll |I|$.

Unlike conventional SR models, we leverage LLMs to solve the recommendation task in an instruction-following paradigm. Specifically, for each user u, we construct a history prompt including the users' interactions $I_{1:n-1} = (I_1, I_2, ..., I_{n-1})$, a candidate item prompt including the candidate items C, and soft prompts including the recommendation patterns and information of conventional SR models. The aforementioned prompts are concatenated along with an instruction that explicitly describes the recommendation task, forming the final prompt P for LLMs. Finally, LLMs employ the prompt P to predict the target item I_n.

B. Soft Prompt

Hard prompts, also known as discrete prompts, are composed of specific vocabulary; unlike hard prompts, soft prompts remove the constraint that prompt embeddings must correspond to natural language words [38]. Soft prompts can be adjusted according to downstream task training data, so they can provide LLMs with "*only LLMs understand*" knowledge that is difficult or impossible for humans to describe in natural language. Although hard prompts usually correspond to natural language and are easily understood by humans, the purpose of prompt construction is to enable LLMs to perform tasks effectively, not for human consumption; additionally, during LLMs inference, both hard and soft prompts are ultimately passed to LLMs in the form of word embeddings, so it is not necessary to limit prompts to human-interpretable natural language [39]. Hence, unlike the general prompt, we will insert a portion of soft prompts into the construction of our prompts. Formally, we denote soft prompts as sp_j and hard prompts as hp_i, where j is the index of soft prompts in the prompt, and i is the index of hard prompts in the prompt. Our prompt P is constructed by both hard and soft prompts:

$$P = \{hp_1, hp_2, ..., sp_1, sp_2, ..., sp_k, ..., hp_{l-1}, hp_l\}, \quad (1)$$

where k represents the number of soft prompts in the prompt P and l represents the number of hard prompts in the prompt P. Both hard and soft prompts in our prompt P will also become word embeddings. However, unlike hard prompts corresponding to a fixed position in the language space, soft prompts will be processed into randomly initialized embeddings. As LLMs learn the target task, the position of the soft prompts in the language space will change:

$$E_1 = \sum_{j=1}^{k} f_{iniz}(sp_j) \quad (2)$$

where E_1 represents the corresponding embeddings directly initialized by the soft prompts, and f_{iniz} indicates the process of randomly initializing to the same dimension as the word embeddings in the language space of LLMs.

C. Parameter Efficient Fine-Tuning

The comprehensive fine-tuning of all parameters within LLMs demands considerable time and computational resources. To mitigate this issue, the approach of Parameter-Efficient Fine-Tuning (PEFT) concentrates on adjusting a minimal subset of parameters, thereby reducing computational demands while maintaining notable performance levels. An example of a PEFT method is AdaLoRA (Adaptive LoRA), which optimizes the number of trainable parameters for weight matrices and layers, unlike LoRA, which evenly distributes parameters across all modules. It allocates more parameters to important weight matrices and layers, while less important ones receive fewer parameters. The optimization goal for AdaLoRA is formulated as follows:

$$\max_{\Theta} \sum_{(x,y)\in D} \sum_{t=1}^{|y|} \log(P_{\Phi_0+\Delta\Phi_0(\Theta)}(y_t|x, y_{<t})), \quad (3)$$

where AdaLoRA introduces the parameters Θ, which are smaller than the original LLM parameters Φ_0.

IV. METHODOLOGY

We propose DELRec framework to improve the performance of LLMs-based SR. The overview is depicted in Figure 3.

In the beginning, DELRec constructs three distinct prompts for different tasks. In the first stage, DELRec updates soft prompts' parameters through two components, *Temporal Analysis* and *Recommendation Pattern Simulating*, to distill information and recommendation patterns from conventional SR models. In the second stage, DELRec integrates the distilled recommendation pattern into LLMs and fine-tunes them to improve performance in conducting more accurate SR tasks.

A. Prompt Construction

We construct two different prompts for the two components of the first stage, and construct one prompt for the second stage. We create three different prompts to help LLMs comprehend and execute the diverse tasks in DELRec. In the general template, all three prompts consist of instruction, processed interaction sequence, candidate set, soft prompts, and prediction. Details of these components are as follows:

Instruction. Each of these three prompts contains unique instructions that specify LLMs' tasks.

Processed Interaction Sequence. The interaction sequence

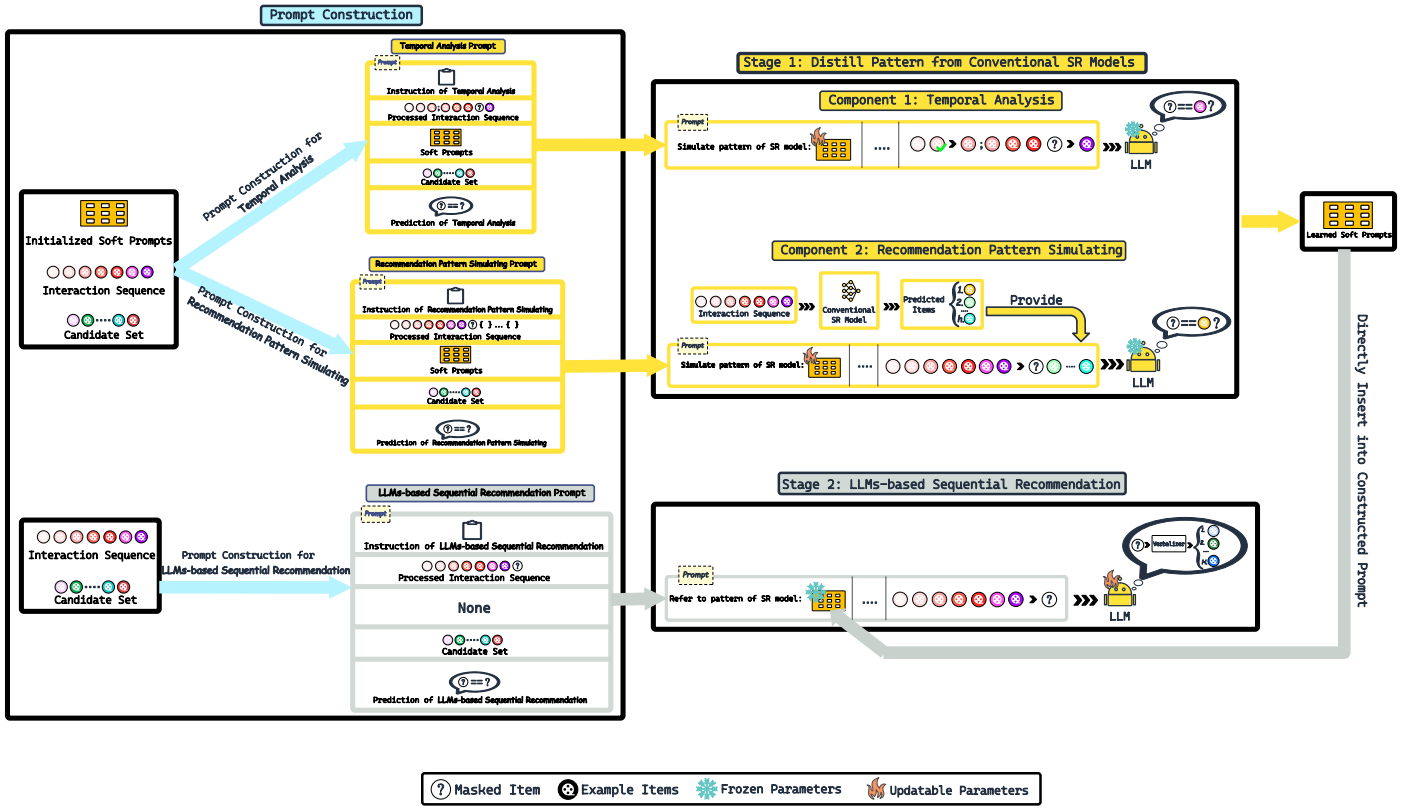

Fig. 3. Illustrating the proposed DELRec. There are three parts in DELRec: *Prompt Construction* aims to construct prompts for the next two stages to guide LLMs to perform different tasks. *Stage 1: Distill Pattern from Conventional SR Models* involves using soft prompts to distill SR patterns from conventional SR models. This stage includes two components: *Temporal Analysis* and *Recommendation Pattern Simulating*. *Temporal Analysis* focuses on performing time analysis on conventional SR models and providing similar time knowledge to LLMs; *Recommendation Pattern Simulating* aims to simulate the recommendation results of conventional SR models, enabling soft prompts to distill similar recommendation knowledge. *Stage 2: LLMs-based Sequential Recommendation* aims to insert distilled soft prompts into constructed prompt and fine-tune LLMs, enabling them to predict the ground truth.

changes based on the three distinct tasks, offering diverse contexts and information sources to the LLMs, which constitutes the most crucial part of the prompts.

Candidate Set. The candidate set consists of one item to be predicted and a series of other random items. This guides LLMs in selecting the target item from the set to ensure that non-existent items are not included in their output.

Soft Prompts. The Soft Prompts part evolves through different stages. For the first stage, initialized soft prompts are inserted into this part to distill information and recommendation patterns from conventional SR models. For the second stage, this part is initially set to none, but after the first stage, learned soft prompts are inserted to provide distilled information, enabling LLMs to make more accurate SRs.

Prediction. The content of the prediction includes masked parts that require the LLMs to predict. It aims to output what the masked item should be according to the instructions.

Specifically, for the task of the first component (*Temporal Analysis*) in the first stage, the prompt we construct aims to let the LLMs analyze the temporal dynamics of the conventional SR models. This component involves updating the parameters of soft prompts, thereby allowing LLMs to simulate the recommendation patterns of conventional SR models at a process level. The detailed content will be introduced in Section IV-B. For the task of the second component (*Recommendation Pattern Simulating*) in the first stage, our constructed prompt is intended to facilitate LLMs in learning and simulating recommendation results from conventional SR models. It also updates soft prompt parameters, enabling LLMs to simulate recommendation patterns at a result level. Further details will be provided in Section IV-B.

Regarding tasks in the second stage, our constructed prompt aims to guide LLMs towards considering previously learned soft prompts more as a reference. It is worth noting that, when constructing the prompt for the second stage, we first leave the places for soft prompts empty and insert the learned soft prompts after the first stage. Additionally, we fine-tune LLMs to let them utilize auxiliary information extracted from conventional SR models for predicting ground truth, ultimately enhancing their ability to perform more accurate SR. The detail will be introduced in Section IV-C.

It is crucial to highlight that when populating the prompts with information, unlike conventional SR tasks where IDs denote items, we represent all items in the prompts (candidate set, processed interaction sequence, prediction) using textual titles. This approach equips LLMs with richer intrinsic details about the items, enhancing their ability to execute the designated tasks effectively. Furthermore, we will incorporate specific names of the conventional SR models to harness the pre-existing knowledge of LLMs and facilitate a deeper understanding of these conventional SR models' recommendation

patterns.

B. Distill Pattern from Conventional SR Models

The first stage of DELRec, *Distill Pattern from Conventional SR Models*, uses soft prompts inserted into the constructed prompts to accurately capture the information and recommendation behavior patterns of conventional SR models for LLMs. We propose this stage to provide LLMs with information from conventional SR models to enhance the performance of LLMs as recommenders. Specifically, this stage consists of two components, namely *Temporal Analysis* and *Recommendation Pattern Simulating*.

Temporal Analysis. Since SR tasks aim to recommend items that are temporally closer based on user interaction sequences, which exhibits strong temporal dynamics, it is crucial to perform a temporal analysis of conventional SR models and provide similar temporal knowledge to LLMs to better simulate conventional SR models' recommendation patterns.

Most conventional SR models (*e.g.*, GRU4Rec, SASRec) achieve temporal awareness by aggregating the features of items in the user interaction sequence to the most recent item in the sequence. We aim to enable LLMs to similarly recognize and learn the importance of "the most recent item," thereby acquiring relevant temporal knowledge. Therefore, our proposed strategy is to provide the interaction sequence and target item, and let the LLMs predict the most recent item in the sequence——a behavior we refer to as PMRI (Predicting Most Recent Item).

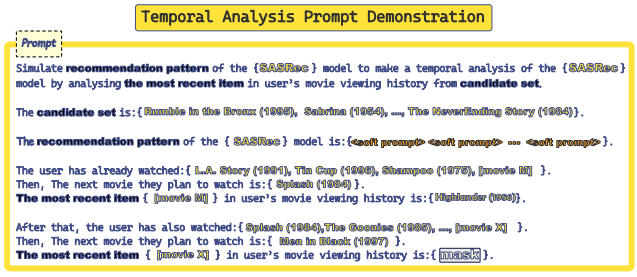

Fig. 4. Demonstration of the prompt for *Temporal Analysis*.

Specifically, our strategy will allow LLMs to perform PMRI on the sequences, and we will also provide in-context learning (ICL) [40] in an ingenious way to not only help LLMs enhance their learning efficiency and quality, but also increase LLMs' temporal awareness, our strategy is as follows:

Given the user interaction sequence $I_{1:n-1} = (I_1, I_2, ..., I_{\alpha-1}, I_\alpha, ..., I_{n-2}, I_{n-1})$, we then inform LLMs that the α-th item will be the next interaction item for the sequence of the first k items $I_{1:\alpha-1}$, which takes the previous part of the sequence as ICL provided to the LLMs. Similarly, we take the last item I_{n-1} as the next item for a sequence $I_{\alpha:n-2}$ and mask the second-to-last item I_{n-2}, allowing LLMs to predict the masked item I_{n-2} and assign it as the label y^0 for this task. During the prediction process, we will use a simple verbalizer to effectively convert the output of the LLM head (*i.e.*, the output scores of all tokens) into ranking scores for all items. In the learning process, the parameters of the LLMs Φ_0 are frozen, and only the parameters of the soft prompts Φ are updated. Afterward, the soft prompts in the prompt will contain knowledge of the conventional SR models' aggregation of item features and knowledge similar to the temporal information of conventional SR models. The prompt of this component is shown in Figure 4. Formally, the learnable parameters of soft prompts Φ are optimized by minimizing the loss function of *Temporal Analysis* (TA):

$$L_{TA} = \sum_{(x^0, y^0) \in D_0} -\log(P_{\Phi_0+\Phi}(y^0|x^0)), \tag{4}$$

where $D_0 = \{(x_i^0, y_i^0)\}_{i=1,...,N}$ contains the prompt and masked item in the aforementioned.

Recommendation Pattern Simulating. Besides *Temporal Analysis*, it is also essential for LLMs to simulate conventional SR models in making similar recommendations, which enables the distillation from the recommendation knowledge of conventional SR models into soft prompts.

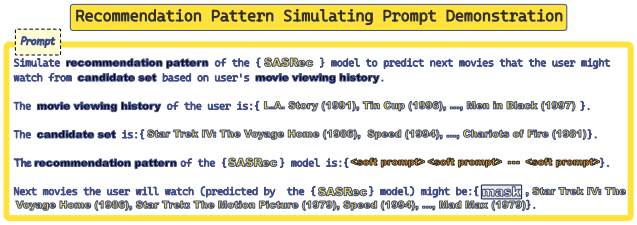

Fig. 5. Demonstration of the prompt for *Recommendation Pattern Simulating*.

Specifically, we will let LLMs simulate the recommendation patterns of conventional SR models as closely as possible and let LLMs predict the recommendation results of conventional SR models (rather than the ground truth) based on the user interaction sequence. This process can be described as: given the user interaction sequence $I_{1:n-1} = (I_1, I_2, ..., I_{n-1})$ and providing the top h recommended items $sr_{1:h} = (sr_1, sr_2, ..., sr_h)$ based on the conventional SR model's predicted probabilities for interaction sequence, we take the highest probability item sr_1 as the label y^1 for the *Recommendation Pattern Simulating* task. Then, during the prediction of y^1, LLMs update the parameters of soft prompts, allowing LLMs to fit the results of the conventional SR model well. The prompt of the task is shown in Figure 5. Specifically, the loss function of *Recommendation Pattern Simulating* (RPS) can be formulated as:

$$L_{RPS} = \sum_{(x^1, y^1) \in D_1} -\log(P_{\Phi_0+\Phi}(y^1|x^1)), \tag{5}$$

where $D_1 = \{(x_i^1, y_i^1)\}_{i=1,...,N}$ consists of the prompt and conventional SR models predicted items in the RPS step.

After obtaining the loss functions for *Temporal Analysis* and *Recommendation Pattern Simulating*, we will proceed to update the parameters of soft prompts in a multi-task learning [41] manner, allowing LLMs to learn from two target tasks simultaneously, thereby achieving the distillation of recommendation behavior patterns for conventional SR models. The learning objective can be defined as:

$$\min_{\Phi} \{\lambda L_{TA} + (1-\lambda) L_{RPS}\}, \tag{6}$$

where λ and $1-\lambda$ represent the weights of learning objectives of the two components, which are dynamically adjusted during training.

C. LLMs-based Sequential Recommendation

In the first stage (*Distill Pattern from Conventional SR Models*), we successfully distilled the recommendation patterns from the conventional SR models. To enable LLMs to fully and effectively utilize the distilled auxiliary knowledge to achieve more accurate SR tasks, we proposed *LLMs-based Sequential Recommendation*, the second stage of DELRec. In this stage, we insert soft prompts learned in the first stage into the prompt constructed in section IV-A and freeze these parameters. Then, we fine-tune LLMs using ground truth.

Insert Soft Prompts. In previous research, to enable LLMs to utilize auxiliary information from conventional SR models (such as item embeddings), people often used projectors ($e.g.$, MLP, Tiny Transformers) to map the embeddings into the language space of LLMs. However, this approach often suffers from poorly designed projectors, which may fail to fully convey the information embedded in original embeddings to LLMs or limit their generalization capabilities. Therefore, distilled soft prompts will be inserted into the prompt we have constructed for the second stage, thereby overcoming previously mentioned issues. The prompt is shown in Figure 6.

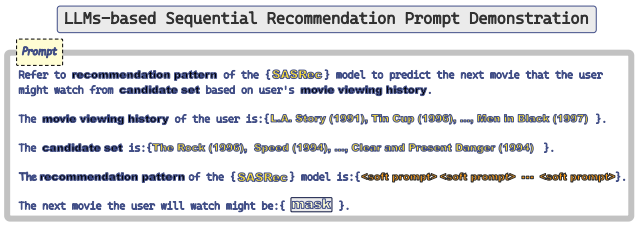

Fig. 6. Demonstration of the prompt for *LLMs-based Sequential Recommendation*.

Specifically, we incorporate the learned soft prompts $sp_{1:k} = (sp_1, sp_2, ..., sp_k)$ into the prompt P. In other words, we use the distilled recommendation behavior patterns from the conventional SR models as context for LLMs to predict the target I_n:

$$I_n = LLM(P(sp_{1:k})) \tag{7}$$

where $P(\cdot)$ represents the process of inserting learned soft prompts $sp_{1:k}$ into constructed prompt P, and $LLM(\cdot)$ indicates that LLMs utilize the prompt to perform SR tasks.

LLMs Fine-tuning. However, the distilled soft prompts may not be ideal for the language of fixed hard prompts, and these distilled soft prompts may contain noise. Therefore, we need to fine-tune LLMs' parameters to bridge the semantic gap between the two (distilled soft prompts and hard prompts) and guide LLMs to use the distilled soft prompts more as a reference, thus fully utilizing these distilled auxiliary information. Formally, given a user interaction sequence $I_{1:n-1} = (I_1, I_2, ..., I_{n-1})$, where the next item I_n the user will interact with is the label $\overline{y}$. In the learning process, when using soft prompts as auxiliary information to guide LLMs in predicting label $\overline{y}$, we freeze the parameters of soft prompts Φ and fine-tune the LLMs using PEFT (AdaLora). Formally, the learning objectives of the LLMs-based Sequential Recommendation (LSR) can be described as follows:

$$\min_{\Theta}\{L_{LSR} = \sum_{(\overline{x},\overline{y})\in\overline{D}} -\log(P_{\Phi_0+\Delta\Phi_0(\Theta)+\Phi}(\overline{y}|\overline{x}))\}, \tag{8}$$

where $\overline{D} = \{(\overline{x_i}, \overline{y_i})\}_{i=1,...,N}$ contains the prompt $\overline{x}$ and the anticipated ground truth $\overline{y}$.

V. EXPERIMENTS

In this section, we assess the performance of our proposed framework, DELRec, on four real-world datasets. We compare it against various baselines, including conventional SR models and LLMs-based models. Our experiments are designed to answer the following research questions:

RQ1: Whether the proposed framework outperforms baseline methods, including the conventional SR models and other LLMs-based models, for SR?

RQ2: Are our proposed DELRec able to learn meaningful recommendation behavior patterns or information?

RQ3: How can key components affect our proposed method? Specifically, how is the efficacy of proposed *Temporal Analysis* and *Recommendation Pattern Simulating*?

RQ4: How do hyperparameters influence DELRec? How do the size and sparsity of datasets affect the performance of DELRec?

RQ5: What are DELRec's memory footprint, inference time and real-time response capability? Does DELRec face challenges with cold start issue?

A. Setup

1) Datasets: We evaluate the proposed DELRec and baseline methods on four real-world datasets, namely MovieLens-100K, Beauty, Steam, and Home & Kitchen, which are commonly used in the SR field. We show the detailed statistics of the datasets in Table I:

MovieLens-100K[1]is a commonly used movie recommendation dataset that includes ratings given by users to movies and the titles of those movies.

Steam[2]not only contains user reviews of video games on the Steam Store, but also covers a variety of game titles.

Beauty[3]contains user feedback on beauty items from Amazon.

Home & Kitchen[3] is a dataset of user feedback containing various items needed for home and kitchen from Amazon website. This dataset contains millions of users and items to test the scalability of DELRec in real-world loads.

For all datasets, we follow [18] in treating users' implicit feedback as interactions between users and items, and determine the sequence order of inputs based on timestamps. Since

[1]https://grouplens.org/datasets/movielens/100k/
[2]https://huggingface.co/datasets/joyliao7777/LLaRA/tree/main/steam
[3]https://github.com/RUCAIBox/RecSysDatasets

users with few interactions are very sparse, it is difficult to extract meaningful patterns or preferences from them. The behavior of these users may not be representative and can easily introduce noise. Therefore, we filter out users and items with fewer than 5 interactions. Meanwhile, we arrange them chronologically as [28] do, and divide the data into training, validation, and test sets in an 8:1:1 ratio. This division method ensures that interactions used for training do not appear in subsequent data, thereby avoiding any potential information leakage.

TABLE I
STATISTICS OF DATASETS.

Dataset	sequence	item	interaction	sparsity
MovieLens-100K	943	1682	100,000	93.70%
Steam	11,938	3,581	274,726	99.36%
Beauty	324,038	32,586	371,345	99.99%
Home & Kitchen	9,767,606	1,286,050	21,928,568	99.99%

2) Baselines: To demonstrate the effectiveness of our DELRec framework, we use two types of baseline models, namely conventional SR models and LLMs-based models. For conventional SR models, we will select the deep learning models that are more widely used and effective in the field of SR. We adopt three models that are often regarded as baselines:

GRU4Rec: GRU4Rec is based on RNN, which focuses on capturing temporal dependencies in user behavior sequences. It processes sequence data by GRU, which can effectively predict the users' next interaction.

Caser: Caser based on CNN predicts users' next interaction by capturing local and global patterns in behavior sequence.

SASRec: SASRec is an SR model based on a transformer, which aims to predict users' interaction by capturing patterns in the sequence of user behaviors. It uses the self-attention mechanism to model users' long-term and short-term preferences.

For the LLMs-based models, we will select three different open-source LLMs and the best-performing LLMs-based SR models (for a fair comparison, we will replace their LLM backbone with **Flan-T5-XL** [42]) from three different paradigms described in Section I:

Bert-Large: Bert-Large is a milestone LLM capable of performing Masked language modeling (MLM) tasks.

Flan-T5-Large: Flan-T5-Large (700M) are well-known open-source LLMs with an encoder-decoder structure.

Flan-T5-XL: Structurally similar to Flan-T5-Large, but due to the larger model size of Flan-T5-XL (3B), Flan-T5-XL performs better on complex tasks.

RecRanker: RecRanker cleverly samples items and users and inputs the results of conventional recommendation models into the prompt. It belongs to the first paradigm.

LLMSEQPROMPT: LLMSEQPROMPT enhances the accuracy of SR by integrating domain knowledge into prompts, fine-tuning LLMs, and then it generates recommendations based on prompts. It belongs to the first paradigm.

LLM-TRSR: LLM-TRSR analyzes users' historical interactions, generates recurrent summaries, and builds prompts that include users' preferences and candidates, then it uses prompts to fine-tune LLMs. It belongs to the first paradigm.

LLaRA: LLaRA inserts the embedding of items encoded by the conventional SR model into the prompt. It belongs to the second paradigm.

LLM2BERT4Rec: LLM2BERT4Rec uses LLMs embeddings to initialize the item embeddings in BERT4Rec, and then changes the dimension of the embeddings by using PCA as a projector before initialization. Then, it uses these embeddings to fine-tune BERT4Rec. It belongs to the second paradigm.

LlamaRec: LlamaRec recalls items with conventional model embeddings and uses a verbalizer to convert the output logits of LLMs into candidate probabilities. It belongs to the third paradigm.

LLMSEQSIM: LLMSEQSIM obtains item embeddings by using LLMs, then calculates session embeddings with item embeddings. It finally calculates similarity and recommends the most similar item to the user. It belongs to the third paradigm.

KDA$_{\mathbf{LRD}}$: KDA$_{LRD}$ uses KDA [43] as backbone, where KDA employs a Fourier-based temporal evolution module to track the dynamic changes in item relations over time. Then, it enhances KDA with LRD which was introduced in section II-B. It belongs to the third paradigm.

We divide the proposed DELRec into three types, with Caser, GRU4Rec, and SASRec serving as the conventional SR model backbones in this framework. And we choose **Flan-T5-XL** as our LLMs backbone, because the main task in the framework we propose is Masked Language Modeling (MLM). We will also use **Flan-T5-Large** to perform ablation experiments. It's worth noting that the LLM backbones of our proposed framework can also use open-source Decoder-Only structured LLMs, such as Llama2 [44], and are not constrained by the types of LLMs.

3) Implementation Details: We will introduce our implementation details in two parts: conventional SR models part and LLMs-based SR model part. First, for the three conventional SR models used as backbones, the implementation details will be respectively modified:

SASRec: For the training of **SASRec**, we use the Adam optimizer and we set the embedding size as 100, and we also use two self-attention blocks with a learning rate of 1e-3, a dropout rate of 0.5 and a batch size of 128.

Caser: For the training of **Caser**, the set is similar to **SASRec**. We use the Adam optimizer and set the number of horizontal filters as 16. And we also set the embedding size as 100 with a learning rate of 1e-3, a dropout rate of 0.4 and a batch size of 128.

GRU4Rec: For for the training of **GRU4Rec**, we use the Adagrad optimizer. And we set the embedding size as 64 with a learning rate of 0.01, a dropout rate of 0.3 and a batch size of 50.

Secondly, for the parts of DELRec that use LLMs, the implementation details will be divided into two parts: the first

TABLE II

OVERALL PERFORMANCE OF DELREC ON FOUR DATASETS. THE BEST-PERFORMING METHOD IN EACH COLUMN IS BOLDFACED, AND THE SECOND-BEST METHOD (IF NOT DELREC) IN EACH COLUMN IS UNDERLINED. THE SUPERSCRIPT $*$ AND $**$ INDICATE $p \leq 0.01$ AND $p \leq 0.05$ FOR THE PAIRED T-TEST OF DELREC VS. CONVENTIONAL SR MODEL BACKBONE.

		MovieLens-100K					Steam				
		HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
Conventional	Caser	0.3250	0.6640	0.5034	0.7713	0.5564	0.3767	0.6680	0.5117	0.7856	0.5618
	GRU4Rec	0.3162	0.6495	0.4824	0.7568	0.5430	0.3786	0.6835	0.5241	0.7915	0.5672
	SASRec	0.3341	0.6704	0.5183	0.8062	0.5727	0.3852	0.6977	**0.5305**	0.8337	0.5943
LLMs-based	Bert-Large	0.0053	0.0204	0.0126	0.0404	0.0236	0.0081	0.0424	0.0349	0.1367	0.0587
	Flan-T5-Large	0.0375	0.0703	0.0509	0.1529	0.1076	0.0540	0.0893	0.0586	0.1938	0.1237
	Flan-T5-XL	0.0938	0.2441	0.1711	0.3437	0.2121	0.1033	0.2857	0.1826	0.4165	0.2474
	LlamaRec	0.2870	0.5873	0.4253	0.7183	0.5387	0.3511	0.6278	0.4522	0.7691	0.5489
	RecRanker	0.3246	0.6492	0.4694	0.7570	0.5464	0.3724	0.6637	0.4743	0.7823	0.5585
	LLaRA	0.3523	0.6753	0.5016	0.8089	0.5632	0.3855	0.6811	0.5081	0.8461	0.5943
	LLMSEQPROMPT	0.2702	0.5681	0.4152	0.6921	0.5332	0.3597	0.6439	0.4546	0.7740	0.5538
	LLM2BERT4Rec	0.3365	0.6507	0.4743	0.7615	0.5518	0.3825	0.6742	0.4886	0.8163	0.5706
	LLMSEQSIM	0.3397	0.6532	0.4728	0.7690	0.5521	0.3848	0.6895	0.4937	0.8171	0.5711
	LLM-TRSR	0.3169	0.6356	0.4637	0.7533	0.5476	0.3691	0.6582	0.4718	0.7788	0.5546
	KDA_{LRD}	0.3618	0.6734	0.5089	0.8185	0.5824	0.4028	0.6904	0.5138	0.8522	0.6077
Ours	DELRec (Caser)	0.3664*	0.6804**	0.5108**	0.8215*	0.5843*	0.4157**	0.6946**	0.5116	0.8604*	0.6108*
	DELRec (GRU4Rec)	0.3635*	0.6722*	0.5062*	0.8137*	0.5807*	0.4296*	0.7099**	0.5192	0.8679*	0.6154*
	DELRec (SASRec)	**0.3701***	**0.6919***	**0.5237***	**0.8386***	**0.5950***	**0.4372***	**0.7285***	0.5286	**0.8863***	**0.6203***

		Beauty					Home & Kitchen				
		HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
Conventional	Caser	0.2241	0.4387	0.4073	0.4664	0.4257	0.1057	0.1597	0.1198	0.2184	0.1536
	GRU4Rec	0.2369	0.4544	0.4109	0.4755	0.4283	0.1091	0.1636	0.1259	0.2239	0.1581
	SASRec	0.2573	0.4629	0.4212	0.5044	0.4368	0.1146	0.1779	0.1365	0.2427	0.1685
LLMs-based	Bert-Large	0.0016	0.0154	0.0082	0.0327	0.011	0.0004	0.0013	0.0008	0.0057	0.0031
	Flan-T5-Large	0.0195	0.0346	0.0265	0.1055	0.087	0.0086	0.0106	0.0066	0.0342	0.0135
	Flan-T5-XL	0.0652	0.2071	0.1402	0.2709	0.1931	0.0257	0.0486	0.0301	0.0728	0.0427
	LlamaRec	0.2361	0.4318	0.3879	0.4618	0.4065	0.1082	0.1618	0.1244	0.2215	0.1574
	RecRanker	0.2470	0.4443	0.4051	0.4686	0.4274	0.1228	0.1811	0.1386	0.2570	0.1713
	LLaRA	0.2752	0.4579	0.4173	0.4886	0.4310	0.1545	0.2046	0.1487	0.2892	0.1902
	LLMSEQPROMPT	0.2390	0.4364	0.3981	0.4627	0.4183	0.1139	0.1752	0.1306	0.2366	0.1604
	LLM2BERT4Rec	0.2575	0.4662	0.4223	0.5132	0.4392	0.1393	0.1870	0.1412	0.2743	0.1819
	LLMSEQSIM	0.2739	0.4724	0.4267	0.5268	0.4436	0.1481	0.1964	0.1452	0.2825	0.1876
	LLM-TRSR	0.2482	0.4593	0.4195	0.4953	0.4342	0.1265	0.1825	0.1383	0.2661	0.1758
	KDA_{LRD}	0.2856	0.4886	0.4261	0.5366	0.4485	0.1684	0.2207	0.1523	0.2957	0.1945
Ours	DELRec (Caser)	0.2819*	0.4850*	0.4266*	0.5345*	0.4479*	0.1707*	0.2256*	0.1564*	0.3025*	0.1987*
	DELRec (GRU4Rec)	0.3083*	0.4929*	0.4283*	0.5519*	0.4576*	0.1729*	0.2293*	0.1572*	0.3079*	0.2013*
	DELRec (SASRec)	**0.3177***	**0.5013***	**0.4445***	**0.5730***	**0.4683***	**0.1836***	**0.2458***	**0.1639***	**0.3155***	**0.2064***

stage of DELRec and the second stage of DELRec:

The First Stage: For the first stage of DELRec, we set the length of user interaction sequences n as 10, which means setting the most recent 10 interactions as the user interaction sequence in order and padding sequences less than 10. we designate the number of user candidate items m to be 15. In other words, these 15 candidate items are composed of one correct value for prediction and fourteen other randomly selected items. Regarding the number of examples α in the ICL of *Temporal Analysis*, we have chosen α as 4 for MovieLens-100K and Beauty based on [45], and we have selected α as 6 for Steam and Home & Kitchen. For the first stage of DELRec, we use the Lion [46] optimizer, with a learning rate of 5e-3 and weight decay of 1e-5 and run on 10 Nvidia 3090 GPUs.

The Seconde Stage: For the second stage of DELRec, we use the same values of n and m as in the first stage, and also use AdaLoRA and Lion optimizer, with a learning rate of 1e-4 and weight decay of 1e-6 and run on 10 Nvidia 3090 GPUs.

4) Evaluation Metrics: For ranking evaluation, we use top-k Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG) as measurement metrics, specifically adopting HR@1, HR@5, HR@10, NDCG@5 and NDCG@10.

*B. Performance Comparison (**RQ1**)*

Table II presents the performance of our method DELRec and various baselines under five evaluation metrics. Comparing DELRec with the aforementioned baseline models, we can derive the following observations:

Conventional SR Models: Combining the results from the table and t-test, we found that DELRec achieved significant performance improvements over all conventional SR model backbones on four datasets. It achieved almost the highest HR@k and NDCG@k scores among conventional SR models that only rely on user interactions. The key reason for this superior improvement is that DELRec effectively combines the information from conventional SR models with the powerful inference ability and rich world knowledge of LLMs, thereby providing more accurate recommendations.

Open-source LLMs: When comparing with some original open-source LLMs (*e.g.*, **BERT**, **Flan-T5**), it is evident that these baseline models not only underperform DELRec in recommendation tasks but also exhibit lower metrics compared to conventional SR models and other LLMs-based recommendation methods. This discrepancy is because while these LLMs possess strong generalization capabilities, they lack domain-specific knowledge and understanding of recommendation patterns, which hinders their performance in recommendation tasks. Therefore, it is crucial to provide appropriate auxiliary information to adapt LLMs to specific recommendation tasks.

LLMs-based SR Models: When considering the other LLMs-based methods we have chosen, the reasons for DELRec's superior performance can be analyzed from several perspectives. Firstly, some methods directly provide recommendation results extracted from conventional recommendation models to prompt LLMs (*e.g.*, **RecRanker**, **LLMSEQPROMPT**, **LLM-TRSR**). For these methods, two main issues arise: 1)

textual information often falls short in accurately and comprehensively describing the specific recommendation behavior patterns of conventional SR models; 2) without guiding information on users' past behavior, LLMs are limited to making decisions based solely on the provided results. Secondly, some methods provide user or item embeddings through dimension transformation to LLMs ($e.g.$, **LLaRA**, **LLM2BERT4Rec**). The main reason that their performance is inferior to DELRec is: although those original embeddings contain information from conventional SR models, the dimension transformation of embeddings through poor designed projector leads to information loss, and ultimately, these embeddings do not align well with the semantic space of LLMs. Thirdly, while some methods ($e.g.$, **LlamaRec**, **LLMSEQSIM**, **KDA$_{\mathbf{LRD}}$**) are able to filter out items information from conventional models for LLMs, they do not provide effective guidance information. There is still room for improvement in terms of providing guidance information for LLMs-based recommendations.

Significance Test: To ensure that DELRec's performance improvements are significant [47] and not due to chance, we conducted significance test using t-test. We evaluated DELRec's performance compared to conventional SR model backbones based on results from multiple experiments. Results indicated that DELRec showed significant performance improvements over all conventional SR model backbones. These results are presented in Table II, where the superscript $*$ and $**$ indicate significance levels of $p \leq 0.01$ and $p \leq 0.05$, respectively.

C. Ablation Studies I (*RQ2*)

To address **RQ2**, we experimented on the soft prompts distilled in the first stage of DELRec and used **SASRec** as the backbone model. As these soft prompts do not correspond to natural language and are not easily interpretable by humans, we performed three transformations on a portion of the soft prompts to verify if they truly capture meaningful recommendation behavior patterns or information. These transformations include:

***w/o* SP (Soft Prompts)**: We removed the soft prompts section and the part of instruction that directs LLMs to refer to auxiliary information from the conventional SR models.

***w* MCP (Manual Construction Prompts)**: This transformation is similar to the general prompt where hard prompts are used to construct auxiliary information. For constructing auxiliary information, we attempted to describe the recommendation process of SASRec model in natural language and replaced the original soft prompts with it.

***w* USP (Untrained Soft Prompts)**: Soft prompts that have not undergone training in the first stage were initialized randomly and inserted into our prompt.

Finally, the three transformed methods are compared with the complete DELRec after fine-tuning in the second stage. Table III shows the measurement metrics of DELRec under four different conditions. Based on our observations, we make the following inferences:

- Soft prompts that have undergone our designed *Distill Pattern from Conventional SR Models* approach surpass all three abovementioned methods. This indicates that our distillation method is able to effectively extract valuable recommendation patterns and information from conventional SR models for LLMs.
- In methods that solely utilize pure hard prompts without soft prompts or manual construction, the Manual Construction method enhances LLMs by describing the recommendation patterns of the conventional SR model in nature language. But due to inaccuracies or insufficient information in these descriptions, the metrics of this method only show slight improvements compared to the No Soft Prompts method.
- Among the few baselines we selected, the Random Soft Prompts method performs poorly in metrics. This can be attributed to random soft prompts being scattered throughout the semantic space with no meaningful context, resulting in strong noise and providing little assistance or potentially misleading LLMs.

D. Ablation Studies II (*RQ3*)

We will verify the impact of various components in DELRec on the framework through the following ablation experiments. The results are shown in Table IV. We introduce the variants and analyze their effect respectively:

***w/o* DPSM (*Distill Pattern from Conventional SR Models*)**: By removing the process of distilling recommendation behavior patterns from conventional SR models in the first stage of DELRec, we observed a decline in performance. This is because LLMs lack auxiliary information from conventional SR models, which hinders their ability to effectively guide the recommendation process.

***w/o* LSR (*LLMs-based Sequential Recommendation*)**: After distillation in the first stage, we remove the fine-tuning process of LLMs in the second stage. This led to a decline in metrics and caused LLMs to exhibit a greater tendency toward favoring items predicted by conventional SR models. It can be attributed to using information extracted directly from conventional SR models introduces noise that may interfere with LLMs-based recommendations.

***w/o* TA (*Temporal Analysis*)**: Removing *Temporal Analysis* during the first stage caused insufficient guidance for LLMs to simulate feature aggregation processes similar to conventional SR models. The reason is that distilled soft prompts lack temporal characteristics.

***w/o* RPS (*Recommendation Pattern Simulating*)**: Eliminating *Recommendation Pattern Simulating* during the first stage shows a decline in metrics, and it becomes challenging for LLMs to effectively simulate overall recommendation behavior patterns exhibited by conventional SR models. It can be attributed to disrupting integration between prediction results of those of conventional SR models and LLMs.

***w* UDPSM (Updating both soft prompts and LLMs parameters in DPSM)**: We observed a decline in performance when updating both soft prompts and LLMs' parameters

TABLE III
ABLATION STUDIES I FOR LEARNED SOFT PROMPTS ON FOUR DATASETS. THE BEST-PERFORMING METHOD IN EACH COLUMN IS BOLDFACED.

	MovieLens-100K					Steam				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
w/o SP	0.3020	0.6211	0.4450	0.8057	0.5646	0.3426	0.6316	0.4785	0.8266	0.5731
w MCP	0.3106	0.6357	0.4603	0.8141	0.5712	0.3608	0.6620	0.4642	0.8531	0.5947
w USP	0.2752	0.5724	0.4296	0.7913	0.5259	0.2977	0.5964	0.4316	0.8136	0.5487
Default	**0.3701**	**0.6919**	**0.5237**	**0.8386**	**0.5950**	**0.4372**	**0.7285**	**0.5286**	**0.8863**	**0.6203**

	Beauty					Home & Kitchen				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
w/o SP	0.2765	0.4762	0.4144	0.5365	0.4521	0.1539	0.2074	0.1356	0.2832	0.2483
w MCP	0.2698	0.4738	0.4088	0.5209	0.4463	0.1602	0.2215	0.1498	0.2968	0.2557
w USP	0.2384	0.4207	0.3764	0.4916	0.4210	0.1341	0.1936	0.1163	0.2541	0.2215
Default	**0.3177**	**0.5013**	**0.4445**	**0.5730**	**0.4683**	**0.1836**	**0.2458**	**0.1639**	**0.3155**	**0.2604**

TABLE IV
ABLATION STUDIES II ON FOUR DATASETS. THE BEST-PERFORMING METHOD IN EACH COLUMN IS BOLDFACED.

	MovieLens-100K					Steam				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
w/o DPSM	0.3020	0.6211	0.4450	0.8057	0.5646	0.3426	0.6316	0.4785	0.8266	0.5731
w/o LSR	0.2814	0.5843	0.4223	0.7845	0.5314	0.3235	0.6120	0.4526	0.8105	0.5744
w/o TA	0.3425	0.6617	0.4867	0.8231	0.5797	0.3710	0.6676	0.4945	0.8450	0.5909
w/o PRS	0.3379	0.6585	0.4805	0.8187	0.5715	0.3555	0.6435	0.4798	0.8313	0.5886
w UDPSM	0.3513	0.6726	0.5013	0.8324	0.5906	0.3974	0.6969	0.5071	0.8687	0.6142
w ULSR	0.3486	0.6678	0.4930	0.8276	0.5823	0.3836	0.6829	0.4983	0.8542	0.6074
w Flan-T5-Large	0.2592	0.5446	0.4074	0.7639	0.5008	0.3018	0.5772	0.4279	0.7924	0.5687
Default	**0.3701**	**0.6919**	**0.5237**	**0.8386**	**0.5950**	**0.4372**	**0.7285**	**0.5286**	**0.8863**	**0.6203**

	Beauty					Home & Kitchen				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
w/o DPSM	0.2765	0.4762	0.4144	0.5365	0.4521	0.1539	0.2074	0.1356	0.2832	0.2483
w/o LSR	0.2666	0.4683	0.4117	0.5287	0.4475	0.1491	0.2013	0.1280	0.2777	0.2425
w/o TA	0.2815	0.4785	0.4192	0.5396	0.4563	0.1564	0.2156	0.1432	0.2915	0.2490
w/o PRS	0.2903	0.4861	0.4294	0.5431	0.4587	0.1682	0.2234	0.1486	0.2967	0.2512
w UDPSM	0.3067	0.4927	0.4373	0.5582	0.4644	0.1738	0.2335	0.1612	0.3082	0.2588
w ULSR	0.2945	0.4886	0.4321	0.5506	0.4619	0.1704	0.2272	0.1566	0.3024	0.2561
w Flan-T5-Large	0.2384	0.4362	0.3865	0.5013	0.4288	0.1235	0.1867	0.1139	0.2594	0.2373
Default	**0.3177**	**0.5013**	**0.4445**	**0.5730**	**0.4683**	**0.1836**	**0.2458**	**0.1639**	**0.3155**	**0.2604**

in DELRec's first stage, rather than just the soft prompts. This simultaneous updating prevents soft prompts from being updated independently, and this results in reducing information flow to the soft prompts and hindering effective SR pattern distillation.

***w* ULSR (Updating both soft prompts and LLMs parameters in LSR)**: We observed a decline in performance when updating both soft prompts' and LLMs' parameters in the second stage of DELRec, rather than only updating the LLMs. The reason is that updating soft prompt parameters in the second stage causes the loss of information and recommendation pattern distilled in the first stage, thereby resulting in those soft prompts unable to provide better reference for LLMs.

***w* Flan-T5-Large**: In addition to ablation experiments on DELRec components, we explored using **Flan-T5-Large** as a smaller-scale LLM backbone. Results indicated that both the size and capacity of LLMs impact DELRec's performance.

*E. Dataset Impact and Hyperparameter Analysis (**RQ4**)*

We experimented with the hyperparameters in DELRec, including the soft prompts size k and the top h recommended items from the conventional SR model ($e.g.$, **SASRec**). We used the HR@1 metric to display results, as it directly reflects the model's ability to recommend the most relevant item, crucial for user satisfaction in SR.

Soft Prompts Size: We examined the impact of soft prompts size k on DELRec's performance. As shown in Figure 7, DELRec's metrics initially improve with an increase in k. However, after reaching 80, the metrics level off. This suggests that while soft prompts enhance prompt information through LLMs' training, an excessive amount can introduce noise or lead to overfitting. Thus, beyond a certain size, additional soft prompts do not significantly improve overall performance.

Recommended Items Size: We investigated how the overall performance changes with varying sizes h of recommended items provided by conventional SR model during *Recommendation Pattern Simulating*. Figure 8 shows a relationship between h and performance. Providing conventional SR model-recommended items helps LLMs understand recommendation patterns. However, too large recommended items size may mislead LLMs and result in excessively long prompts, which could potentially undermine LLMs' attention mechanism.

We also conducted experiments to analyze the impact of dataset size and sparsity on DELRec's performance.

Dataset Size: Regarding the impact of dataset size on DELRec's performance, the results in Table II show that DELRec's performance does not necessarily increase with the dataset size. This indicates that dataset size does not significantly affect DELRec's performance, demonstrating its scalability.

Dataset Sparsity: To better analyze the impact of dataset sparsity on DELRec's performance, we added a dataset called **KuaiRec**[4](a user viewing record dataset from Kuaishou app, containing 7,176 users, 10,728 items, and approximately 12,530,806 interactions, with about 83.72% sparsity). We selected **KuaiRec** as the sparsest dataset among those we used to more clearly differentiate the sparsity levels of different datasets. Then we compared DELRec with $\mathbf{KDA}_{LRD}$ (a SOTA method for LLMs-based SR) and **SASRec** (a SOTA method

[4]https://github.com/chongminggao/KuaiRec

TABLE V

DATASET SPARSITY IMPACT ON THREE DATASETS. THE BEST-PERFORMING METHOD IN EACH COLUMN IS BOLDFACED, AND THE SECOND-BEST METHOD IN EACH COLUMN IS UNDERLINED. THE SPARSITY OF EACH DATASET IS INDICATED IN PARENTHESES NEXT TO THE DATASET'S NAME.

	Beauty (99.99%)					MovieLens-100K (93.70%)					KuaiRec (83.72%)				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
SASRec	0.2573	0.4629	0.4212	0.5044	0.4368	0.3341	0.6704	0.5183	0.8062	0.5727	0.5379	0.7934	0.5794	0.8737	0.6659
KDA$_{LRD}$	0.2856	0.4886	0.4261	0.5366	0.4485	0.3618	0.6734	0.5089	0.8185	0.5824	0.5446	0.8061	0.5826	0.8865	0.6715
DELRec	**0.3177**	**0.5013**	**0.4445**	**0.5730**	**0.4683**	**0.3701**	**0.6919**	**0.5237**	**0.8386**	**0.5950**	**0.5615**	**0.8243**	**0.5881**	**0.9001**	**0.6803**

for conventional SR) on three datasets: **MovieLens-100K**, **Beauty**, and **KuaiRec**. As shown in Table V, as dataset sparsity increases, excessive noise makes it difficult for DELRec to accurately capture recommendation patterns and predict unobserved interactions, resulting in performance decline. Even though DELRec's performance declines as sparsity increases, it consistently outperforms other baselines across all datasets.

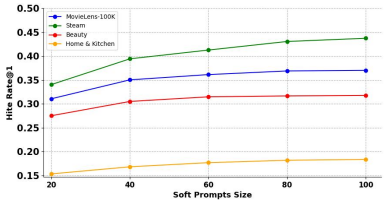

Fig. 7. Performance comparison w.r.t different soft prompts size k for training DELRec on the four datasets.

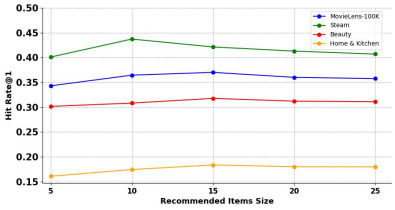

Fig. 8. Performance comparison w.r.t different recommended items size h for training DELRec on the four datasets.

F. Computational Efficiency Analysis (*RQ5*)

To answer the first question of **RQ5**, we tested the memory footprint and inference time of DELRec to verify its scalability under real-world loads:

Memory Footprint: The designed DELRec has around 3 billion parameters with its LLM backbone (**Flan-T5-XL**) and 0.2 million parameters for soft prompts, resulting in a model memory footprint of about 12 GB. When we use the largest dataset (**Home & Kitchen**), the peak memory footprint during training is approximately 234 GB; during inferring, the peak memory footprint is approximately 68 GB.

Inference Time & Real-Time Response Capability: Regarding the inference time of DELRec, we processed 1,000 requests using 10 Nvidia 3090 GPUs in batch. The total inference time was approximately 181.69s (the average of ten calculations), with an average processing time of 0.182s per request. In contrast, the LLM backbone (**Flan-T5-XL**) of DELRec, has an average inference time of 0.161s, which is only 21ms faster than DELRec. But DELRec significantly improves the accuracy of SR tasks and demonstrates good real-time response capability.

Cold Start: Regarding the second question of **RQ5**, it is noteworthy that DELRec does not exhibit significant shortcomings in cold start problem. This is because DELRec, an LLMs-based SR framework, has been pre-trained and accumulated rich world and recommendation domain knowledge through the first stage of distilled soft prompts. To verify this, we conducted a comparative experiment specifically for users with very few interactions (fewer than 3 interactions) on **Home & Kitchen** to closely mimic real-world loads. The results show that, with only a tiny amount of interactions, DELRec's metrics (*HR@1: 0.1082, HR@5: 0.1736, NDCG@5: 0.1344, HR@10: 0.2391, NDCG@10: 0.1605*) outperform **SASRec** (*HR@1: 0.0954, HR@5: 0.1418, NDCG@5: 0.1180, HR@10: 0.1937, NDCG@10: 0.1495*) and are almost on par with **KDA**$_{LRD}$ (*HR@1: 0.1077, HR@5: 0.1759, NDCG@5: 0.1358, HR@10: 0.2383, NDCG@10: 0.1587*).

G. Case Study

To investigate the effectiveness of integrating recommendation patterns from conventional SR models with LLMs' world knowledge, we conducted a comparative case study using **Flan-T5-XL**, **SASRec**, and DELRec. Here we choose a distinct example. For a user with a viewing history that includes "American Beauty (1999)", "Legends of the Fall (1994)", "Gladiator (2000)", "Out of Sight (1998)", "GoldenEye (1995)", "Mission: Impossible(1996)", "Malice (1993)", "Amistad (1997)", "Jurassic Park (1993)" and "Men in Black (1997)", we generated recommendations using the three models. As shown in Figure 9, **Flan-T5-XL** recommended "Men in Black II (2002)" based on the name of user's last watched movie. **SASRec**, considering user's recent viewing history, recommended the action/sci-fi film "Aliens (1986)", which aligns with the theme of "Men in Black". In contrast, DELRec combined conventional SR patterns with rich world knowledge and considered user's shift in preferences from drama/classic to action/sci-fi genres and recommended "Back to the Future (1985)", which was indeed user's next interaction.

Fig. 9. Case study comparison results of the effectiveness of three models in recommending movies: Flan-T5-XL, SASRec, and DELRec.

VI. CONCLUSION

This work introduces DELRec, a framework that enhances LLMs in SR tasks by extracting behavioral patterns from conventional SR models. Through proposed DPSM and LSR, DELRec reduces information loss and improves SR effectiveness. Extensive experiments on four real-world datasets validate our framework. DELRec offers a new approach for utilizing LLMs in complex SR tasks, capturing information and global context missed by conventional SR models.

References

[1] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: challenges, progress and prospects," *arXiv preprint arXiv:2001.04830*, 2019.

[2] X. Chen, Z. Li, W. Pan, and Z. Ming, "A survey on multi-behavior sequential recommendation," 2023. [Online]. Available: https://arxiv.org/abs/2308.15701

[3] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu *et al.*, "A survey on large language models for recommendation," *WWW*, vol. 27, no. 5, p. 60, 2024.

[4] A. Petrov and C. Macdonald, "Rss: effective and efficient training for sequential recommendation using recency sampling," *Recsys*, vol. 3, no. 1, pp. 1–32, 2024.

[5] J. Zhang, R. Xie, Y. Hou, W. X. Zhao, L. Lin, and J.-R. Wen, "Recommendation as instruction following: A large language model empowered recommendation approach," *arXiv preprint arXiv:2305.07001*, 2023.

[6] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)," in *RecSys*, 2022, pp. 299–315.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[8] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He, "Tallrec: An effective and efficient tuning framework to align large language model with recommendation," in *RecSys*, 2023, pp. 1007–1014.

[9] Y. Cao, N. Mehta, X. Yi, R. Keshavan, L. Heldt, L. Hong, E. H. Chi, and M. Sathiamoorthy, "Aligning large language models with recommendation knowledge," *arXiv preprint arXiv:2404.00245*, 2024.

[10] J. Zhai, X. Zheng, C.-D. Wang, H. Li, and Y. Tian, "Knowledge prompt-tuning for sequential recommendation," in *MM*, 2023, pp. 6451–6461.

[11] W. Luo, C. Song, L. Yi, and G. Cheng, "Kellmrec: Knowledge-enhanced large language models for recommendation," *arXiv preprint arXiv:2403.06642*, 2024.

[12] L. Li, Y. Zhang, and L. Chen, "Prompt distillation for efficient llm-based recommendation," in *CIKM*, 2023, pp. 1348–1357.

[13] J. Wu, T. Yu, R. Wang, Z. Song, R. Zhang, H. Zhao, C. Lu, S. Li, and R. Henao, "Infoprompt: Information-theoretic soft prompt tuning for natural language understanding," *NeurIPS*, vol. 36, 2024.

[14] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*. New York: Association for Computing Machinery, 2010.

[15] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*. IEEE, 2016, pp. 191–200.

[16] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.

[17] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*. ACM, 2018, pp. 565–573.

[18] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018, pp. 197–206.

[19] S. Rajput, N. Mehta, A. Singh, R. Hulikal Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Tran, J. Samost *et al.*, "Recommender systems with generative retrieval," *NeurIPS*, vol. 36, 2024.

[20] J. Li, M. Wang, J. Li, J. Fu, X. Shen, J. Shang, and J. McAuley, "Text is all you need: Learning language representations for sequential recommendation," in *SIGKDD*, 2023, pp. 1258–1267.

[21] Y. Hou, Z. He, J. McAuley, and W. X. Zhao, "Learning vector-quantized item representation for transferable sequential recommenders," in *WWW*, 2023, pp. 1162–1171.

[22] Y. Cui, F. Liu, P. Wang, B. Wang, H. Tang, Y. Wan, J. Wang, and J. Chen, "Distillation matters: Empowering sequential recommenders to match the performance of large language model," *arXiv preprint arXiv:2405.00338*, 2024.

[23] H. Xue and F. D. Salim, "Promptcast: A new prompt-based learning paradigm for time series forecasting," *TKDE*, 2023.

[24] S. Luo, B. He, H. Zhao, Y. Huang, A. Zhou, Z. Li, Y. Xiao, M. Zhan, and L. Song, "Recranker: Instruction tuning large language model as ranker for top-k recommendation," *arXiv preprint arXiv:2312.16018*, 2023.

[25] S. Luo, Y. Yao, B. He, Y. Huang, A. Zhou, X. Zhang, Y. Xiao, M. Zhan, and L. Song, "Integrating large language models into recommendation via mutual augmentation and adaptive aggregation," *arXiv preprint arXiv:2401.13870*, 2024.

[26] J. Harte, W. Zorgdrager, P. Louridas, A. Katsifodimos, D. Jannach, and M. Fragkoulis, "Leveraging large language models for sequential recommendation," in *Recsys*, 2023, pp. 1096–1102.

[27] Z. Zheng, W. Chao, Z. Qiu, H. Zhu, and H. Xiong, "Harnessing large language models for text-rich sequential recommendation," in *WWW*, 2024, pp. 3207–3216.

[28] J. Liao, S. Li, Z. Yang, J. Wu, Y. Yuan, X. Wang, and X. He, "Llara: Aligning large language models with sequential recommenders," *arXiv preprint arXiv:2312.02445*, 2023.

[29] N. Guo, H. Cheng, Q. Liang, L. Chen, and B. Han, "Integrating large language models with graphical session-based recommendation," *arXiv preprint arXiv:2402.16539*, 2024.

[30] X. Ren, W. Wei, L. Xia, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Representation learning with large language models for recommendation," in *WWW*, 2024, pp. 3464–3475.

[31] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *CIKM*, 2019, pp. 1441–1450.

[32] J. Shlens, "A tutorial on principal component analysis." [Online]. Available: https://arxiv.org/abs/1404.1100

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[34] A. Acharya, B. Singh, and N. Onoe, "Llm based generation of item-description for recommendation system," in *RecSys*, 2023, pp. 1204–1207.

[35] Z. Yue, S. Rabhi, G. de Souza Pereira Moreira, D. Wang, and E. Oldridge, "Llamarec: Two-stage recommendation using large language models for ranking," 2023. [Online]. Available: https://arxiv.org/abs/2311.02089

[36] X. Yu, L. Zhang, X. Zhao, Y. Wang, and Z. Ma, "Ra-rec: An efficient id representation alignment framework for llm-based recommendation," *arXiv preprint arXiv:2402.04527*, 2024.

[37] S. Yang, W. Ma, P. Sun, Q. Ai, Y. Liu, M. Cai, and M. Zhang, "Sequential recommendation with latent relations based on large language model," 2024. [Online]. Available: https://arxiv.org/abs/2403.18348

[38] L. Giray, "Prompt engineering with chatgpt: a guide for academic writers," *Annals of biomedical engineering*, vol. 51, no. 12, pp. 2629–2633, 2023.

[39] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[40] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, and J. Xu, "A survey on in-context learning," 2024. [Online]. Available: https://arxiv.org/abs/2301.00234

[41] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[42] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *JMLR*, vol. 25, no. 70, pp. 1–53, 2024.

[43] C. Wang, W. Ma, M. Zhang, C. Chen, Y. Liu, and S. Ma, "Toward dynamic user intention: Temporal evolutionary effects of item relations in sequential recommendation," *TOIS*, pp. 1–33, 2020.

[44] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[45] Z. Chu, Z. Wang, R. Zhang, Y. Ji, H. Wang, and T. Sun, "Improve temporal awareness of llms for sequential recommendation," *arXiv preprint arXiv:2405.02778*, 2024.

[46] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, H. Pham, X. Dong, T. Luong, C. Hsieh, Y. Lu, and Q. V. Le, "Symbolic discovery of optimization algorithms," in *NeurIPS*, 2023.

[47] J. S. Farris, M. Källersjö, A. G. Kluge, and C. Bult, "Constructing a significance test for incongruence," *Systematic biology*, vol. 44, no. 4, pp. 570–572.