4-2024

# Sequential recommendation: From representation learning to reasoning

Lei WANG
*Singapore Management University*, lei.wang.2019@phdcs.smu.edu.sg

# SEQUENTIAL RECOMMENDATION: FROM REPRESENTATION LEARNING TO REASONING

LEI WANG

SINGAPORE MANAGEMENT UNIVERSITY

2024

# Sequential Recommendation: From Representation Learning to Reasoning

by

**Lei Wang**

Submitted to School of Computing and Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

**Dissertation Committee:**

Ee-Peng LIM (Supervisor / Chair)
Professor of School of Computing and Information Systems
Singapore Management University

Hady Wirawan LAUW
Associate Professor of School of Computing and Information Systems
Singapore Management University

Yong WANG
Assistant Professor of School of Computing and Information Systems
Singapore Management University

Kwan Hui LIM
Associate Professor of School of Information Systems Technology and Design Pillar
Singapore University of Technology and Design

Singapore Management University
2024

I hereby declare that this PhD dissertation is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree in any university previously.

Lei Wang

---

Lei Wang

30 April 2024

# Sequential Recommendation: From Representation Learning to Reasoning

Lei Wang

## Abstract

The recommender system is a crucial component of today's online services. It helps users navigate through an overwhelmingly large number of items and discovering those that interest them. Unlike general recommender systems, which recommend items based on the user's overall preferences, sequential recommender systems consider the order of user-item interactions. Sequential recommendations aim to predict the next item a user will interact with, given a sequence of previously interacted items, while considering the short-term and long-term dependencies among items.

In this thesis, we focus on sequential recommendation methods: from representation learning to large language model (LLM)-based reasoning. On the one hand, representation learning-based sequential recommendation methods usually feed ID embeddings of interacted items into models, such as deep neural networks, to generate user representation vectors. They then rank candidate items to create a recommendation list based on the similarity between user representation vectors and candidate item vectors. On the other hand, the LLM-based reasoning approach mainly depends on the LLM's strong reasoning ability and rich world knowledge. LLM-based reasoners require carefully designed prompts and/or demonstration examples considering the task complexity and prompt length constraint.

This thesis consists of three parts. In the first part, we aim to improve representation learning for sequential recommendation and present our efforts in building an explanation-guided contrastive learning sequential recommendation model. In particular, we first present the data sparsity issue in the sequential recommendation and the false positive problem in contrastive learning. Next, we demonstrate how to

utilize explanation methods for explanation-guided augmentation to enhance positive and negative views for contrastive learning-based sequential recommendation, thereby improving the learned representations.

Most sequential recommendation methods primarily focus on improving the quality of user representation. However, representation learning-based methods still suffer from several issues: 1) data sparsity; 2) difficulty adapting to unseen tasks; and 3) lack of world knowledge; 4) lack of human-style reasoning for generating explanations. To address these issues, the second part of this thesis investigates how we can build sequential recommendation models based on large language models. In particular, we introduce two new research directions for LLM-based sequential recommendation: 1) zero-shot LLM-based reasoning of recommended items and 2) few-shot LLM-based reasoning of recommended items. For zero-shot LLM-based reasoning of recommended items, we use an external module for generating candidate items to reduce the recommendation space and a 3-step prompting method for capturing user preferences and making ranked recommendations. For few-shot LLM-based reasoning of recommended items, we study what makes in-context learning work for sequential recommendation and propose incorporating multiple demonstrations into one aggregated demonstration to avoid the long input problem and improve recommendation accuracy. Both directions offers new and exciting research possibilities for using LLMs in recommender systems.

LLMs are generally capable of human-style reasoning which could be used to generate explanations for a large set of tasks. Therefore, the final part of the thesis addresses the explanation generation task and the evaluation of explanation for sequential recommendation results using LLMs. Specifically, we introduce a framework for LLM-based explanation to support automatic evaluation of an LLM's ability to generate plausible post-hoc explanations from the content filtering and collaborative filtering perspectives. Using our created benchmark data, the experiment results show that ChatGPT with appropriate prompting can be a promising explainer for recommendation tasks.

# Acknowledgements

My past five years at SMU, including four and a half years pursuing my PhD and half a year as a research engineer in Living Analytics Research Centre (LARC), have been an unforgettable and invaluable experience. This was my first time living and studying in a country outside China for such a long time. I however enjoyed every day of my stay in Singapore conducting research while overcoming challenges. I feel very fortunate to have the opportunity to pursue this PhD journey and it would not have been possible without the help and support of many individuals, and I deeply appreciate them.

First and foremost, my deepest thanks go to my advisor, Prof Ee-Peng Lim. Prof Lim gave me the opportunity to work at LARC before my PhD study. Having known my weakness in English conversation, he encouraged me to always communicate with him in English so as to improve my English listening and speaking skills. He has been very helpful in correcting my English pronunciation and suggesting ways to express myself better. He has taught me how to conduct solid research, how to identify research questions, how to propose new ideas, how to write better, and how to give good presentation. I truly appreciate his efforts as my thesis advisor. His passion for research has truly inspired me. I would like to express my gratitude to him for the guidance, patience, and encouragement which have made my Ph.D. journey both enjoyable and productive. My dissertation would not have been possible without him.

I would like to thank my dissertation committee members: Professors Hady W. Lauw, Yong Wang, Jisun AN, and Kwan Hui Lim. They have provided very valuable suggestions and insightful comments during my Qualifying Exam, Oral Proposal Defense, and Final Dissertation Defense. I would also like to thank BOO Chui Ngoh, Caroline TAN, Desmond YAP, Prof. Baihua Zheng, ONG Chew Hong, and Andrew TAY Ren Ann for their help and support in administrative and technical matters.

During my PhD study, I was fortunate to have the opportunity to work with many outstanding researchers. I would like to express my gratitude to all my collaborators with whom I have co-authored many papers on large language models: Zhiqiang Hu, Prof. Yunshi Lan, Prof. Xing Xu, Dr. Wei Qin, Yihuai Lan, Wanyu Xu, Songheng Zhang, Jiabang He, Yi Hu, Prof. Hao Wang, Shenshen Li, YuanYuan Liang, Qingyuan Tian, Yang Wang, and Dr. Yun Wang. I am grateful to my mentors and collaborators on math word problem solving: Prof. Dongxiang Zhang, Dr. Yan Wang, Jipeng Zhang, Dr. Yi Bin, Dr. Minghuan Tan, Jierui Li, Zhenwen Liang, Dr. Deng Cai, Prof. Jing Jiang, Qiyuan Zhang, and Dr. Long Guo. On the work on recommender systems, I thank Weijieying Ren, Dr. Tianxiang Zhao, and Dr. Zhiwei Liu. I really appreciate the guidance and advice from Prof. Roy Ka-Wei Lee, Prof. Bingtian Dai, Prof. Deqiang Ouyang, Prof. Jianfei Yu, Dr. Shuohang Wang, Dr. Chenghao Liu, Dr. Pei-Chi Lo, Dr. Sicheng Yu, Dr. Yaoyiran Li, Dr. Zhihao Wen, Dr. Gary Ang, Dr. Yixin Xu, and Prof. Shengfeng He. I am thankful to my close friends in Singapore who helped to relieve me from the stress during PhD: Yibin Lai, Rui Cao, Haoxin Tu, Han Tang, Yufeng Dai, Shawn, and Xiaosen Zheng. I also thank my juniors and friends from UESTC who accompanied me during my master's study: Linfei Lei, Xiang Zhou, Ying Liu, Mengting Ding, Yi Liu, Yuyang Nie, Shan Gou, Ni Wang, Yu Liu, and Zheng Zhu. Lastly, I am forever grateful to my five BFFs for their consistent support and companionship over the years.

I thank my parents and grandmother for their endless love and support. They have always stood by me during the 13 years of my university life. The one or two video chat each week has been the only way we could maintain our very close relationship over these years. I hope that my PhD graduation will make my parents proud, and they do not need to worry about my study and career in the future.

Lastly, I would like to thank my girlfriend Shan for her endless love and support. Prior to my PhD study, Shan and I spent much happy time doing things together, including doing cooking, traveling, and studying. Shan is really good at cooking, and I am her most ardent food fan. Unfortunately, during the long PhD study and Covid-19 pandemic, we had not been able to meet up to do things together. I am immensely grateful for her patience and sacrifices, including leaving her job to study at SMU, and learning English to adapt to Singapore’s environment, so that we could live and study together. I appreciate everything she has done for me.

# Table of Contents

# List of Figures

# List of Tables

# Authorship Attribution Statement

This thesis includes material from 3 papers in which I am listed as an author.

Chapter 3 is published with material from:

- **Lei Wang**, Ee-Peng Lim, Zhiwei Liu, Tianxiang Zhao. Explanation guided contrastive learning for sequential recommendation. Proceedings of the 31st ACM International Conference on Information & Knowledge Management (**CIKM** 2022).

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- Prof. Ee-Peng Lim improved the idea, suggested the design of the experiments, and revised the manuscript.
- Dr. Zhiwei Liu and Dr. Tianxiang Zhao provided suggestions to improve the idea.

Chapter 4 is from:

- **Lei Wang** and Ee-Peng Lim. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. Preprint 2023. **Cited by 66 papers**.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- Prof. Ee-Peng Lim improved the idea, suggested the design of the experiments, and revised the manuscript.

Chapter 5 is published with material from:

- **Lei Wang** and Ee-Peng Lim. The Whole is Better than the Sum: Using Aggregated Demonstrations in In-Context Learning for Sequential Recommendation. 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics (**NAACL** 2024 Findings).

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).

- Prof. Ee-Peng Lim improved the idea, suggested the design of the experiments, and revised the manuscript.

# Chapter 1

# Introduction

## 1.1 Background

The recommender system is a crucial component of today's online services [27, 43]. It helps users navigate through a potentially very large set of items (e.g., products, songs, movies, articles, places of interest, and others) and discover the interesting ones for subsequent interactions (e.g., purchase, consumption, visit, and others). In non-sequential recommendation, items are recommended based on a user's overall preferences, without considering the sequential order of user-item interactions [113]. Specifically, the past item interactions of the user are modelled as a set. Nevertheless in many real world applications, users interact with items in sequential order. To learn the patterns within a user-item interaction sequence for predicting the next item a user wants to interact with, different sequential recommendation models has been developed [80, 30, 111].

Consider the user-item interaction sequence example in Figure 1.1(a). In this interaction sequence, the user has purchased five items in the past from a purple shirt and to a carton of milk. The sequential recommendation task is to predict the next item (denoted by a question mark) given the above input sequence. Both the short-term and long-term dependencies among items are to be learned from the many observed user-item sequences in order to perform sequential recommendation

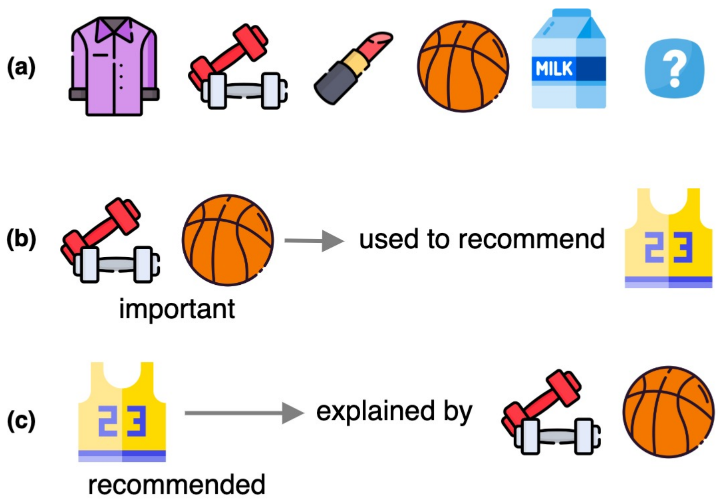


Figure 1.1: A sample user-item interaction sequence.

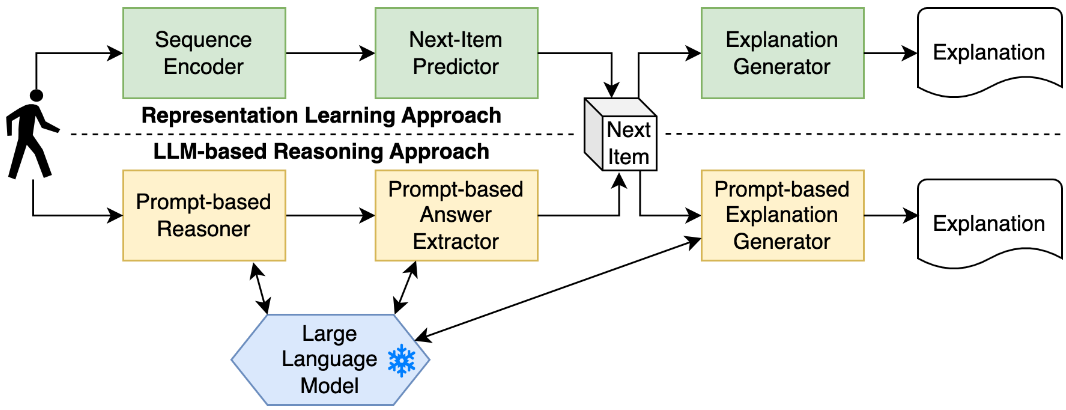


Figure 1.2: Representation Learning Approach versus Large Language Model-based Reasoning Approach.

effectively.

## 1.2 Motivation

As shown in Figure 1.2, we broadly categorize the existing sequential recommendation methods under the *representation learning* and *large language model (LLM)-based reasoning* approaches. Before the era of LLMs, sequential recommendation models primarily relied on representation learning. This includes both machine learning and deep learning-based methods. In representation learning, a user's item sequence is input into a sequence encoder, which generates a corresponding representation vector. Item vectors from the item pool are then used to compute similarity with this representation vector. Items from the item pool are ranked based on these calculated similarities. The higher an item ranks, the more likely it is to be recommended to the user.

These sequential recommendation models have certain drawbacks. Firstly, these methods are unable to capture the rich textual world knowledge that could enhance user preference modeling and the quality of recommendations. Secondly, many of these methods are designed for a specific domain and struggle to adapt to new recommendation domains. For example, the item IDs in movie recommendation data are unrelated to the item IDs in book recommendation data, making it difficult for representation learning based on ID embedding to generalize. Thirdly, these methods lack a human-understandable explanation as to why an item is recommended for a specific user. Such explanations can help improve the recommender system and enhance the user experience.

Unlike methods based on representation learning, LLM-based methods mainly focus on using prompt-based reasoners for sequential recommendations. These methods rely heavily on LLM like ChatGPT, which have demonstrated remarkable performance in numerous natural language processing (NLP) tasks and strong capabilities to solve these NLP tasks through task-specific prompts in a zero-shot setting or in a few-shot setting, without requiring any examples for further fine-tuning. As shown in Figure 1.2, this thesis primarily focuses on representation learning and LLM-based reasoning for sequential recommendations. In representation learning, a sequence encoder inputs the user sequence to generate a user representation, followed by a next-item predictor that predicts recommended items. Explanation methods, such as template-based text generation, can then create explanations for these recommended items. For LLM-based reasoning in sequential recommendations, prompts are crafted to utilize LLMs for item recommendations. The LLMs' textual output contains information about recommended items. Then, an answer extractor is used to extract the recommended items. A prompt-based explanation generator can be used to produce an high-quality textual explanation for these recommended items.

When using LLM-based reasoners, it is crucial to carefully design prompts or prepare demonstration examples for specific tasks, such as sequential recommendation in the movie domain or explanation generation for sequential recommendations. After obtaining these prompts or examples, we feed them into the LLM to obtain a textual output. This output is then processed by a post-processing module to provide the desired answer.

For sequential recommendation tasks, incorporating LLMs offers at least the following benefits. Firstly, their strong performance in complex reasoning tasks indicates an excellent inference ability. This can help to predict the next recommendation with explicit explanations, enhancing transparency and improving user experience. Secondly, LLMs contain extensive world knowledge which effectively compensates for the limited local knowledge of traditional sequential recommendation models. This provides additional information and helps to model more complex patterns, going beyond only sequence pattern modeling. Therefore, in this thesis, we will also focus on developing LLM-based sequential recommendation models.

## 1.3 Overview of Dissertation Works

In the following, we summarize the dissertation works into two parts and describe the key research ideas in each part.

### 1.3.1 Explanation Guided Representation Learning of Item Sequences

Compared to user representations derived from all interacted items in the user sequence, the representation derived from important items is more effective in capturing the dependencies between interacted items and the next item. For instance, in Figure 1.1(b), the dumbbell set and basketball from the user sequence are considered significant in predicting the next item, a basketball shirt. Meanwhile, the remaining items like lipstick and milk contribute less. In this thesis, we introduce the idea of modeling user representations considering important items. This requires the detection of important items using explanation methods and the leverage of these items to learn improved user representation vectors.

We propose to use contrastive learning to enhance user representations. The reasons for choosing contrastive learning include: (1) sequential recommendation models often suffer from data sparsity, making it challenging to learn high-quality user representations. Secondly, contrastive learning allows for the consideration of both important and unimportant items, which can further improve user representation when crafting positive and negative views. Specifically, we propose explanation-guided augmentations to infer the important items of a given user sequence using explanation methods and consider item importance in augmentation operations. This way, better positive views and negative views can be derived for contrastive learning. We also propose the explanation-guided contrastive learning for sequential recommendation framework to utilize the positive and negative views for self-supervised and supervised learning of user sequence representations, in addition to the standard recommendation loss function.

### 1.3.2 LLM-reasoning based Sequential Recommendation and Explanation

#### Zero and Few-Shot LLM-based Reasoning of Recommended Items

In the zero-shot LLM-based method, we must carefully design prompts to better utilize the rich world knowledge from LLMs and elicit their reasoning abilities for more effective next-item recommendations. There are two major challenges that need to be considered: (1) the recommendation space can be extremely large, but LLMs without training cannot easily limit the recommendation space; (2) LLMs are inherently unfamiliar with the user preferences required for recommendations. To address these issues, we first limit the recommendation space for a user to items within a candidate item set by using user or item filtering techniques. Then, we introduce a zero-shot prompt-based reasoner for sequential recommendation via a 3-step prompting method: (i) capturing user preferences (Step 1), (ii) selecting representative items from the user's interacted items (Step 2), and (iii) recommending a ranked list of items (Step 3).

As LLMs increase in model parameters and training corpus size, they gain an emergent ability known as in-context learning (ICL), allowing them to reason based on a few demonstration examples within a given context. In this thesis, we also explore the use of in-context learning for sequential recommendation. We study how in-context learning can enhance sequential recommendation via a preliminary empirical study. Specifically, we investigate the impact of four aspects of demonstrations, including the wording of prompts, task consistency between demonstrations and test instances, selection of demonstrations, and number of demonstrations. Through the preliminary study, we discover that increasing the number of demonstrations in ICL reduces recommendation accuracy. To cope with limited prompt length while introducing demonstration(s), we propose a method that incorporates multiple demonstration users into one aggregated demonstration.

#### LLM-based Post-Hoc Explanation

For improved user experience and system accountability, sequential recommendation models should support transparency and explainability in their recommendation results. This allows users to understand the reasons behind recommended items. As depicted in Figure 1.1(c), a basketball shirt recommendation can be explained by the presence of a dumbbell and a basketball in the user sequence.

Post-hoc explanations for representation learning-based sequential recommendations can use post-hoc explanation methods such as: 1) proxy models like LIME[114], 2) attention weights [15], 3) template-based textual explanation generation [71], and 4) cosine similarity over item representations drawn from the model itself [98]. These methods can help identify items in the user sequence and provide insight into why the recommendation models recommend a particular item. In this thesis, we mainly focus on developing prompt-based post-hoc explanation. The reason is that LLMs, like ChatGPT, have demonstrated exceptional ability in providing explanations for general tasks. This indicates their significant potential for generating high-quality and human readable explanations for sequential recommendations. Unlike the evaluation of recommendation accuracy, assessing the ability of LLMs to explain recommendation results comes with several challenges. Firstly, it is difficult to solicit explanations from the LLMs that allow us to evaluate their performance, as many LLMs do not provide detailed nor structured explanations. Secondly, and more importantly, it is difficult to obtain golden ground truth reasons for evaluating the explanations of each recommended item. Therefore, in this thesis, we aim to develop LLM-based methods that generate explanations for sequential recommendation results, and how to develop an automatic evaluation system for evaluating LLMs' ability to explain accuracy.

## 1.4 Dissertation Structure

We first discuss related works that are connected to the topics of this thesis in Chapter 2. The rest of this thesis consists of two parts — Part I: Representation Learning of Item Sequences and Part II: LLM-based Reasoning and Explanation of Recommended Items.

Part I contains Chapter 3, focusing on representation learning of item sequences, with an emphasis on building better user representation that allows sequential recommendation models to make better next-item recommendations. In Chapter 3, we first discuss the data sparsity problem in the sequential recommendation task and the false positive problem caused by the data augmentation of existing contrastive learning-based methods. We then formulate the problem and give readers a better sense of how the proposed method fundamentally differs from two strong contrastive learning for sequential recommendation baselines. Next, we introduce the proposed

**EC4SRec** on how to utilize explanation methods to perform Explanation Guided Augmentations (**EGA**) to aid contrastive learning-based sequential recommendation. We present experimental results on four representative sequential recommendation datasets and in-depth analyses of the proposed model to understand better why explanation methods can help obtain more accurate recommendation performance. This chapter is based on our work [139].

Part II, which includes Chapters 4, 5, and 6, focuses on leveraging LLMs' strong reasoning capabilities for making next-item recommendations. It also covers the use of LLMs for generating and automatically evaluating explanations.

In Chapter 4, we explore how to leverage the rich world knowledge and strong reasoning abilities of LLMs to make next-item recommendations. We present our proposed Zero-Shot Next-Item Recommendation (**NIR**) strategy. This strategy uses an external module for generating candidate items and a 3-step prompting method for capturing user preferences and making ranked recommendations. Finally, we present a comprehensive evaluation of the proposed NIR on two widely used benchmarks. NIR competes well with strong sequential recommendation models, offering new and exciting research possibilities for using LLMs in recommender systems.

In Chapter 5, we explore how to develop sequential recommendation models using in-context learning. We first conduct a preliminary empirical study to investigate the role of four aspects of demonstrations, including the wording of prompts, task consistency between demonstrations and test instances, selection of demonstrations, and number of demonstrations. Then, we introduce our proposed method called **LLMSRec-Syn**, which incorporates multiple demonstration users into one aggregated demonstration. We also conduct a comprehensive evaluation of the proposed **LLMSRec-Syn** on three recommendation datasets.

In Chapter 6, we study LLM's ability to explain recommended items in sequential recommendation tasks. Specifically, we introduce our proposed Framework for LLM-based EXplanation of Next-Item Recommendation (**FLEX**). We create two benchmark datasets for both CTF and CLF explanations using a subset of MovieLens dataset. Finally, we present an evaluation of the ability of LLM-based explanation using the FLEX framework on our benchmark datasets.

Finally, in Chapter 7, we summarize the thesis's contributions and suggest potential future directions.

## 1.5 Contributions

The technical contributions of this thesis are summarized as follows:

- We improve the state-of-the-art representation learning of user-item interaction sequences by exploiting explanation methods to determine the importance of items in the sequences when augmenting the sequences for contrastive learning. We incorporate this idea into our proposed EC4SRec model which demonstrates superior performance over existing sequential recommendation methods across multiple benchmark datasets.

- We propose one of the early zero-shot sequential recommendation methods using the LLM-based reasoning approach. To address the lack of item universe and user knowledge in LLMs, our method incorporates an external candidate item generation module and a 3-step prompting strategy to capture user preferences. We evaluate the proposed approach on both the MovieLens 100K and LastFM 2K datasets, and obtain promising accuracy results. This shows the potential of using LLMs in zero-shot recommendations.

- We explore in-context learning for sequential recommendation by systematically investigating the effect of instruction format, task consistency, demonstration selection, and number of demonstrations. We also propose a novel one-shot sequential recommendation method called LLMSRec-Syn which leverages on in-context learning and the novel concept of aggregated demonstration. The key idea behind aggregated demonstration is to incorporates multiple demonstration users into one aggregated demonstration to conserve prompt length while taking advantage of the multiple demonstration users for better in-context learning outcome.

- Finally, we tackle the explanation generation task and the evaluation of explanation results for sequential recommendation results. We proposed a Framework for LLM-based EXplanation of Next-Item Recommendation (FLEX) that supports an automatic evaluation of a LLM's ability to generate plausible post-hoc explanations, focusing on two prevalent perspectives: content filtering and collaborative filtering. The results on the introduced dataset show that LLM can achieve promising explanation performance, suggesting the potential to develop effective LLM-based explainers for a wide range of recommendation models across different domains.

# Chapter 2

# Related Work

In this chapter, we aim to present previous works related to this thesis. Following the research framework adopted in this dissertation work (see Section 1.2), we will divide them under representation learning approach, LLM-based reasoning approach, and LLM-based explanation approach. All sequential recommendation works using the representation learning approach focus on learning high-quality user and item representations for predicting the next-items by matching the user and next-items in the learned representation space. In this chapter, we will cover both the machine learning and deep learning methods for learning the representations of user-item interaction sequences and items (see Sections 2.1 and 2.2). Lately, researchers have turned to exploring the reasoning ability of LLMs in the development of next-generation recommender systems. We will thus report some latest works in this line of research in Section 2.3. Lastly, we review research on explanation methods for sequential recommendations (Section 2.4).

## 2.1 Machine Learning-based Sequential Recommendation

The popular machine learning methods for sequential recommendation encompass the following categories: markov chains and factorization-based methods.

**Markov Chains (MC).** MC models focus on modeling low-order transition relationships between user interactions. The first-order MC models [47] consider only the most recent item, while the low-order MC models [46] consider dependencies among the last few interacted items. The representation vectors in these MC methods can be derived by learnable matrices [47]. Although these MC models consider low-order dependencies among items, they struggle to capture complex relationships among items when the user-item interaction sequence is relatively long.

**Factorization-based Methods.** Matrix factorization (MF) methods model the user-item interaction sequences as a user-item matrix that is subsequently decomposed into two lower-rank matrices, one capturing the representation vectors of users and another capturing the representation vectors of items. BPR-MF [113] optimizes the two decomposed matrices using a pairwise ranking objective function. The MF-based sequential recommendation model [134], simplified from Factorization Machines [112], involves considering the dependencies between items that are not far apart and a candidate item for predicting recommendations. FPMC [111] combines MF and MC to learn the general preference of a user by factorizing the user-item matrix and to learn a transition graph over items based on the user's recent actions. FISM [57] performs matrix factorization on an item-item matrix, instead of learning explicit user representations. To overcome the sparsity problems prevalent in real-world datasets, Fossil [46] combines similarity-based methods with Markov Chains to make personalized sequential recommendations.

While these machine learning-based sequential recommendation methods have demonstrated promising results, they do not provide an effective strategy for modeling high-order and complex dependencies among items.

## 2.2 Deep Learning-based Sequential Recommendation

Due to recent advances in neural networks and their ability to model complex information, deep neural networks have been used to model complex sequential dependencies in sequential recommendations. In this section, we categorize the deep learning-based sequential recommendation methods into multi-layer perceptron methods, recurrent neural network methods, convolutional neural network methods, attention methods, and graph neural network methods. In representation learning methods using deep neural networks, a user's item sequence is fed into a sequence encoder based on a deep neural network. This encoder then generates a corresponding representation vector. The similarity between this vector and item vectors from the item pool is computed. Items are ranked based on these computed similarities. The higher an item's ranking, the more likely it is to be recommended to the user.

**Multi-layer Perceptron (MLP) methods.** MLPs, or feed-forward neural networks with multiple hidden layers, are good at learning the nonlinear relationships between input and output through nonlinear activation functions. This makes MLP-based sequential recommendation models highly capable of capturing complex and nonlinear relationships among items. For example, FMLP-Rec [185] employs an embedding layer to obtain item embedding and uses multiple stacked learnable filter-enhanced blocks, which consist of filter layers and feed-forward layers, to derive user representations.

**Recurrent Neural Network (RNN) methods.** The effectiveness of RNNs in sequence modeling has been widely demonstrated in the field of natural language processing (NLP). RNN-based methods typically use RNN and its variants to derive user representation from a given sequence of items for a user. To model long dependent relationships among user-item interactions, RNNs such as the Gated Recurrent Unit (GRU) [26] have been adapted for sequential recommendations to model user interactions [50, 129, 49]. For example, GRU4Rec [50] incorporates

GRU to model sequence-level patterns. This is further improved by replacing the GRU with a hierarchical RNN [107]. The main limitation of RNN-based sequential recommendation methods is that they could not cope with global dependencies for extremely long user-item sequences, and their training is costly, especially for lengthy sequences.

**Convolutional Neural Network (CNN) methods.** Originally developed for image processing, a typical CNN structure includes convolution layers, pooling layers, and fully connected feed-forward layers. The strength of CNN is its ability to effectively capture dependent relationships across local information, such as pixel correlations in specific regions of an image. In sequential recommendations, CNN-based models [130, 133] usually generate user representations by applying a CNN with sliding windows over the given user sequence and identify local features within a short user-item sequence. For instance, the model Caser [130] is proposed to use CNNs for modeling complex item-item relationships within the local regions of an user-item interaction sequence.

**Attention based methods.** The attention mechanism, inspired by human visual attention, has been widely used in computer vision [95, 159] and natural language processing tasks [5, 91]. The core concept of the attention mechanism in sequential recommendation models is to assess the importance of various segments of an input user sequence on the recommended item. These methods generate user representations either through a neural network like RNN, followed by an attention layer, or through a transformer encoder. Li et al. [66] propose a hybrid encoder, which combines RNN with an attention mechanism, to model the user's sequential behavior and capture the user's main purpose. Attention has also been incorporated into memory networks for sequential recommendation tasks, and they achieve excellent performance. The memory network is developed to store and update attribute-level preference information explicitly [52]. Due to the recent success of Transformers using self-attention to model global and long-term relationships in natural language text [135, 32], they have been used to model temporal patterns in sequential recommendation. For instance, SASRec [58] uses a self-attention layer to learn which items are most influential in the history sequences to characterize complex correlations between item transitions over time. Based on BERT [32], BERT4Rec [125] employs a bidirectional Transformer layer to model user-item interaction sequences. To support efficient model learning, LightSANs [34] introduces a low-rank decomposed self-attention to model the context of the items in user-item sequences. LSSA [158] includes a long- and short-term self-attention network that takes into account both long-term preferences and short-term sequential dynamics.

**Graph Neural Networks (GNNs).** GNNs in sequential recommendation methods are designed to learn representations of items by collectively aggregating neighboring node information based on the graph structure. GNN-based models have also been proposed to capture more complex patterns than sequential ones by combining them with self-attention networks [13, 151, 157]. For example, Wu et al. [151] uses GNNs for session-based recommendation, capturing more complex relationships between items in a sequence. In this model, each session is represented as a composition of long-term preference and short-term interests within a session, using an attention network.

While the above deep learning-based methods have contributed to improved recommendation accuracy through sequence representation learning, they face several drawbacks. These include (1) sole reliance on knowledge in the training data, without considering the open-world knowledge, (2) inability to cope with unbalanced user-item interactions and prevalent long-tailed items, and (3) poor prediction performance for cold-start users.

## 2.3 Large Language Model-based Sequential Recommendation

Recently, Large Language Models (LLMs) have astounded the world with their impressive performance across a wide range of NLP tasks [11, 105]. LLMs, such as ChatGPT [99], GPT-4 [100], LLaMa [132], and PaLM [24], are typically very large transformers with billions of parameters, are trained on enormous textual corpora. They display emergent skills such as chain-of-thought reasoning [145], instruction following [25], and in-context learning [11]. They also show strong abilities in dealing with complex scenarios [145, 62, 140]. Inspired by the success of LLMs in various domains, the recommender system community also begins to propose LLM-based recommendation methods [51, 138, 82, 150, 160, 9]. LLM-based recommender systems can be categorized into (a) LLM-augmented recommender systems [38, 154], and (b) LLM-only recommender systems [51, 29, 6, 171]. LLM-augmented recommender systems leverage LLM's reasoning and world knowledge to improve existing recommendation models, while LLM-only recommender systems directly treat LLMs as recommendation models. Since this section primarily focuses on LLM-based recommender systems that focus on reasoning, prompting, in-context learning, and LLM knowledge for sequential recommendation, it will leave out instruction tuning, parameter-efficient fine-tuning, and foundation models for recommendation.

**LLM-Augmented Recommender Systems.** This category of methods primarily uses the reasoning ability and world knowledge of LLMs to enhance the performance of existing recommendation models. Chat-REC [38] is an LLM-augmented recommender system with a conversational chat interface. The supervised learning recommender system generates a set of candidate items for recommendations. LLM then reranks these items and selects the top-k items as the final recommendations. Furthermore, Chat-REC can also provide explanations for these recommended items. Using prompting or in-context learning strategies, LLMs can generate auxiliary textual information for enhancing recommendations. For example, Du et al. [33] proposed to use LLMs to summarize user information and job requirements to enhance job recommendation accuracy. KAR [154] uses LLMs to obtain two types of external knowledge: user preference reasoning and item factual knowledge. KAR then converts them into augmented vectors, which can be directly used to improve the performance of any recommendation model. LLMRG [144] designs four modules powered by LLMs to construct personalized reasoning graphs: 1) a chain module for causal reasoning, 2) an expansion module based on user interests, 3) a verification module for reasoning procedures, and 4) a self-improvement module to store validated reasoning chains. The resulting reasoning graph is then encoded using graph neural networks, which serve as an additional input to improve conventional recommender systems.

**LLM-Only Recommender Systems.** These methods directly treat LLMs as recommender systems, requiring LLMs to generate recommended items, either from candidate item lists included in prompts or based on the general knowledge of LLMs. GenRec [56] leverages the context comprehension capabilities of LLMs to transform interaction histories into structured prompts for generating the next item, instead of computing a ranking score for each candidate item individually to determine which to recommend. GPT4Rec [69] starts by generating hypothetical "search queries" (user representation vectors) using a language model (GPT-2). This model uses the titles of items from a user's history, along with a generation prompt, as inputs. Then, GPT4Rec employs the BM25 algorithm to find the item most similar to these "search queries" from the entire list of items. Dai et al. [29] conducted an empirical analysis on ChatGPT's recommendation abilities. Hou et al. [51] explores LLMs (e.g., GPT-4) as ranking models in recommender systems, revealing promising zero-shot abilities. Liu et al. [82] introduced a one-shot in-context learning method that utilizes the previous user-item interactions of the target user as one demonstration example. Compared to the existing prompting methods using LLMs for sequential recommendation, which only use prompts with historical or given candidate items, our method detailed in Chapter 4.3 of this thesis employs user-filtering and item-filtering to derive a candidate item set and incorporates the set into a 3-step prompting strategy to leverage LLMs. Furthermore, we are the first to conduct a comprehensive empirical study on what makes in-context learning effective for sequential recommendation. We additionally propose a new demonstration construction strategy, i.e., demonstration aggregation.

## 2.4 Explanation Methods

Among the studies on explanation methods for recommendation tasks [39, 131, 177, 110], most focus on explaining why a recommendation algorithm recommends certain item for an input user's past interacted item sequence. Our thesis, on the other hand, focuses on the use of general local explanations to enhance the accuracy of sequential recommendation and the evaluation of LLM-based explanation for sequential recommendation. We thus review two types of related work: local explanation methods and evaluation of LLM-based explanations. Note that, in this section, most of the related work pertains to explanation, and may not be directly relevant to sequential recommendation.

**Local Explanation Methods.** Local explanation methods aim to elucidate how a model generates specific prediction for a given input. In other words, local explanation methods are originally designed to identify input features that can explain the prediction results, which is a subset of post-hoc explanation methods that are used to interpret and understand the decisions of models after they have been trained. The method detailed in Chapter 3 primarily uses local explanation methods to identify important items from a user's item sequence. Saliency [121], Gradient & Input (GI) [61, 3], Guided Backpropagation [124], and Integrated Gradient [126], are gradient methods that have been applied to a number of models in various fields. Saliency [121] derives an input feature's attribution score by returning the gradient with respect to the input feature. Integrated Gradient [126] takes derivatives of the value for the predicted label with respect to the input features. It outperforms Saliency but is less efficient. Attention can be used to understand or visualize the prediction of the model by extracting the attention heads from the attention module [5, 135]. However, attention as explanation is known to be unreliable [53, 147, 118] since different attention distributions can make the same prediction. In addition to using raw attention weights directly, more elaborate attention explanation mechanisms, such as Attention Rollout, have been proposed [1]. Unlike the raw attention weights, which show the importance given to different parts of the input for a single layer, Attention Rollout aggregates attention weights across all layers of the model. Originally developed for explaining image classification [4], the layer-wise relevance propagation (LRP) method has been applied to explaining predictions for NLP models [153]. Occlusion [170] is a perturbation based explanation method which computes input features' attribution scores by the difference between outputs of the original and perturbed input features. Beyond occlusion, perturbation-based methods can also be used to identify the most important parts of the input by observing changes in model confidence and Shapley values [89, 3] by using input reductions [36]. LIME [114] is a surrogate-based explanation method that first samples points around the input and uses the model's evaluations concerning these sampled points to train a surrogate model, such as logistic regression, to explain the important features. Apart from post-hoc explanation methods, multiple select-then-predict methods have been developed [65, 7, 55]. For example, FRESH [55] predicts binary importance labels over snippet inputs and then uses these highlighted snippets to make a final prediction.

**LLM-based Explanation.** LLMs such as GPT-4 have demonstrated impressive abilities to generate natural language explanations for their predictions [145, 74, 141]. However, it remains unclear whether these explanations truly assist humans in understanding the model's reasoning. Therefore, well-designed evaluation methods are needed to better assess the performance of explanation methods. Ideally, human annotators would label explanations as good or bad, but this is costly. A common method to evaluate the plausibility of an explanation involves removing a certain percentage (k%) of the most important or unimportant tokens, then observing the changes in the output [14, 96]. Recently, it is recommended to have evaluation datasets for specific purposes to assess factual editing in LLMs [181, 90]. Additionally, GPT-judge [79] use the GPT-3 model to measure whether a language model is truthful in generating answers. The above LLM-generated explanations are not directly relevant to explanations for sequential recommendation models. To the best of our knowledge, there is no previous work about LLM-based explanation for sequential recommendation. The work detailed in Chapter 6 explores how to leverage LLM to generate explanations for sequential recommendation and how to evaluate explanations for sequential recommendation.

# Chapter 3

# Explanation Guided Contrastive Learning for Sequential Recommendation

In this chapter, we cover the work **E**xplanation Guided **C**ontrastive Learning for **S**equential **Rec**ommendation (**EC4SRec**) on how to utilize explanation methods to perform Explanation Guided Augmentations (**EGA**) to aid contrastive learning-based sequential recommendation. The key idea behind EGA is to use explanation method(s) to determine items' importance in a user sequence and derive the positive and negative sequences accordingly. EC4SRec then combines both self-supervised and supervised contrastive learning over the positive and negative sequences generated by EGA operations to improve sequence representation learning for more accurate recommendation results. Extensive experiments on four real-world benchmark datasets demonstrate that EC4SRec outperforms the state-of-the-art sequential recommendation methods and two recent contrastive learning-based sequential recommendation methods, CL4SRec and DuoRec. Our experiments also show that EC4SRec can be easily adapted for different sequence encoder backbones (e.g., GRU4Rec and Caser), and improve their recommendation performance.[1]

[1]Code is available at https://github.com/demoleiwang/EC4SRec.


Figure 3.1: Motivation example: (a) A given user sequence with seven items and a red hair-dryer as the next item; (b) Two positive views generated by random mask operations on the given sequence and a negative view which is the sequence of another user. [M] represents a masked item.

The chapter is organised as follows. We first present the objective in Section 3.1. Then we review the existing works, including sequential recommendation models, contrastive learning methods, and explanation methods, in Section 3.2. In Section 3.3, we formulate the problem and give readers a better sense of how the proposed method differs fundamentally from two strong contrastive learning for sequential recommendation baselines. In Section 3.4, we present the proposed EC4SRec followed by the experiment Section 3.5. Finally, we conclude this chapter with a summary.

## 3.1 Objective

**Background.** Recommender systems have played an important role in today's online services [27, 43] to help users navigate the overwhelming amount of information and discover interesting items. Since sequential patterns of user-item interactions change with time, researchers thus [50, 129, 130, 58] pay much attention to sequential recommendation which focuses on short-term and long-term dependencies among items in user sequences to predict next user-item interaction(s).

For a sequential recommendation method to yield accurate results, it has to learn a high-quality user representation from the user's historical sequence and match the user representation against candidate items. Traditional methods model low-order dependencies between users and items via Markov Chain and Matrix Factorization [111, 46]. Recently, researchers have developed deep learning-based sequential recommendation methods using deep neural networks (such as recurrent neural networks [50], convolutional neural networks [130], transformer [58], and graph neural networks [13]) which learn higher-order dependencies to predict the next items. However, data sparsity is still a major challenge due to limited data about users and items in the long tail. The former refers to many users having very short item sequences. The latter refers to many items having very few user interactions. To cope with these challenges, contrastive learning-based (CL-based) sequential recommendation works [156, 184, 87] incorporate positive and negative views of original user sequences by augmentations and sampling so as to learn more robust user sequence representations, thus more accurately matching candidate items to improve recommendation performance.

**Motivating Example.** Figure 3.1 shows an example of the contrastive learning approach to sequential recommendation. From a given user sequence shown in Figure 3.1(a), we obtain two positive views of the user sequence using some augmentation operator(s), and select the sequence of another user as a negative view. For the positive views, we randomly mask as few items in the given user sequence as shown in Figure 3.1(b). To learn user sequence representations, contrastive loss(es) is introduced to make the representations of positive views to be close to each other, but far from that of the negative view [156, 184, 87].

Note that even as CL approach has been shown to improve sequential recommendation performance, its user sequence augmentation and sampling methods are performed with randomness (e.g., random crop, random mask, and random reorder) and is thus prone to produce for a given user sequence positive views that look very different or negative views that look quite similar. As a result, the learned sequence representations are non-ideal which subsequently affects the recommendation accuracy. For example, if we were to know that the red hair dryer is the next item, the hair care items in the original sequence will be considered more relevant (or important). The positive view 2 in Figure 3.1(b) will look more different from the original user sequence, while positive view 1 without any masked hair care items will look more similar. Attracting the representations of positive views 1 and 2 to be closer to each other is therefore inappropriate and may degrade the recommendation performance. By the same reasoning, the negative view may be inappropriately sampled if it shares many hair care items with the two positive views.

**Proposed idea.** The above motivating example suggests that we need to carefully choose positive and negative views in order to learn good high-quality user sequence representations. To begin this research, we thus conduct a small experiment to show that items important to the next-item of the predicted sequence should be treated differently from non-important items for CL-based sequential recommendation to achieve high accuracy. While this result is interesting, it is infeasible to know which items are important in a user sequence as the next-item is not given during model training. To determine the elusive "important items", we propose **explanation guided augmentations** (**EGA**) to infer the important items of a given user sequence using explanation methods and consider item importance in augmentation operations. This way, better positive views and negative views can be derived for contrastive learning. We also propose the **Explanation Guided Contrastive Learning for Sequential Recommendation (EC4SRec)** framework to utilise the positive and negative views for self-supervised and supervised learning of user sequence representations, combined with recommendation loss function. EGA and EC4SRec are also designed to accommodate different sequential recommendation backbones. In other words, they can be readily applied to existing self-supervised and supervised CL methods to improve their recommendation performance.

**Our contributions.** In summary, our contribution is three-fold:

- We propose a model-agnostic Explanation Guided Contrastive Learning for Sequential Recommendation (EC4SRec) framework that incorporates explanation methods into user sequence augmentations for generating positive and negative user sequences for both self-supervised and supervised contrastive learning. EC4SRec can be seen as an integration of CL4SRec and DuoRec, with an additional sampling of negative views for contrastive learning to more effectively separate the representations of positive views from that of the negative views. To our knowledge, EC4SRec is also the first that utilizes explanation methods to improve sequential recommendation.
- We propose several explanation guided augmentation operations to generate both positive and negative user sequences using importance score derived from explanation methods. With these operations, EC4SRec can effectively utilize augmented positive and negative user sequences in contrastive learning to obtain better sequence representations for recommendation.
- We evaluate the proposed augmentation method over two types of contrastive learning frameworks, with three different baseline sequential recommendation models, on four real-world datasets. Our experiment results show that EC4SRec significantly outperforms the vanilla CL4SRec and DuoRec, and other state-of-the-art sequential recommendation methods. We also demonstrate the generalizability of EC4SRec using different sequence encoders and combinations of explanation methods with consistent performance improvement by $4.2\%$ to $23.0\%$.

## 3.2 Existing Methods

### 3.2.1 Sequential Recommendation

Sequential recommendation aims to learn high-quality user and item representations to predict the next item of a given user sequence. Early works focused on modeling low-order transition relationships between items via Markov Chains as item-item features to be used for recommendation [111, 46, 157]. With the advances in neural networks, researchers turn to using neural networks [50, 66, 130, 58, 174, 127, 172, 88, 137, 67], such as RNN [50], CNN [130], Transformer [58], and GNN [13] to model high-order sequential dependencies hidden in historical user-item interactions. GRU4Rec [50], for example, incorporates GRU to model sequence-level patterns. This is further improved by replacing GRU by hierarchical RNN [107]. Caser [130] on the other hand uses CNN to model high-order item-item relationships. Inspired by the effectiveness of self-attention in NLP communities [148], Kang, et al. [58] applied self-attention in sequential recommendation named SASRec. GNN based models [87, 151] are also proposed to capture more complex patterns than sequential patterns. To improve sequential recommendation by both performance and interpretability, various works [52, 169, 20] began to determine items contributing to the next-item prediction. Explanation methods, such as attention weights [148], gradient-based methods [170, 126], and Occlusion [121] have been designed to determine features that explain the prediction labels. In our research, we explore the use of explanation methods to determine specific earlier items in the user sequence that explain the predicted next-item and in turn improve sequential recommendation accuracy under the EC4SRec framework.

### 3.2.2 Contrastive Learning

Contrastive learning has recently achieved great successes in various research domains including computer vision [18, 45, 41, 102], NLP [35, 37], recommendation [184, 156, 23, 149, 23, 183, 86, 166, 149, 136, 78], etc.. It aims to obtain high-quality representations by pulling positive views of the same instance closer while pushing the positive views and their negative views apart in the representation space. $S^3$Rec [184] pre-trains sequential recommendation by contrastive learning with four self-supervised tasks defined on historical items and their attributes. CL4SRec [156] combines recommendation loss with contrastive loss of self-supervised tasks to optimize the sequential recommendation model. CoSeRec [87] introduces two new augmentation operations, insert and replace, to train robust sequence representations. DuoRec [106] retrieves the positive view of a given user sequence by finding another user's sequence which shares the same next-item in its proposed supervised contrastive learning. In the following section, we will further elaborate CL4SRec and DuoRec. The above contrastive learning-based sequential recommendation methods, nevertheless, suffer the same pitfalls mentioned in our motivating example. In this research, we therefore seek to address these pitfalls by explanation-guided augmentations and explanation-guided contrastive learning framework.

### 3.2.3 Explanation Methods

While there are several works on explainable recommendation [39, 131, 177], they are designed to explain why an item is predicted by the recommendation algorithm. This work mainly focuses on general explanation methods [121, 170, 126] originally designed to determine features that explain the prediction results. By applying these methods to sequential recommendation methods, we can determine historical items in a user sequence that explain the predicted next-item, and assign importance scores to these historical items. For example, Saliency [170] derives an input feature's attribution score by returning the gradient with respect to the input feature. Integrated Gradient [126] takes derivatives of the value for the predicted label with respect to the input features. It outperforms Saliency but is less efficient. Occlusion [121] is a perturbation-based explanation method which computes input features' attribution scores by the difference between outputs of the original and perturbed input features. Prediction models incorporating attention mechanism compute attention weights as the relative importance of items. However, using attention weights as explanation is controversial [54, 148] since different attention distributions can produce the same model predictions.

## 3.3 Preliminaries

### 3.3.1 Problem Formulation

Suppose that we have a set of users $\mathcal{U}$ and items $\mathcal{V}$. For the sequential recommendation task, each user $u \in \mathcal{U}$ has a sequence of items the user has interacted with in the past. We denote this sequence by $s_u = [v_1^u, v_2^u, \ldots, v_{|s_u|}^u]$ where $v_i^u \in \mathcal{V}$ and $|s_u|$ denotes the sequence length. The items in the sequence are ordered by time. The goal of sequential recommendation is to predict the next item at time step, i.e., $v_*^u$, using the observed historical sequence $s_u$. Suppose $P(v|s)$ is a model that returns the probability of $v$ being the next item given a sequence $s$. The sequential recommendation task can be formulated as:

$$v_*^u = \arg\max_{v \in \mathcal{V}} P\Big(v_{|s_u|+1}^u = v \mid s_u\Big).$$

### 3.3.2 Contrastive Learning for Sequential Recommendation

In this section, we describe a set of basic augmentation operations to determine positive views of a given user sequence.

**Basic Augmentation Operations.** There are four basic augmentation operations [184, 156, 106] to generate positive views from an original user sequence, $s_u = [v_1^u, v_2^u, \ldots, v_{|s_u|}^u]$.

- **Random Crop** (crop)**:** It randomly selects a continuous sub-sequence from positions $i$ to $i + l_c$ from $s_u$ and removes it. $l_c$ is defined by $l_c = i + \lfloor \mu_c \cdot |s_u| \rfloor$ where

$\mu_c$ $(0 < \mu_c \leq 1)$ is a hyper-parameter. The cropped sequence is defined by:

$$s_u^{\mathrm{c}} = [v_i^u, v_{i+1}^u, \ldots, v_{i+l_c}^u] \tag{3.1}$$

- **Random Mask** (mask)**:** It randomly selects a proportion $\mu_m$ of items from $s_u$ to be masked. Let $g^m(1), g^m(2), \cdots, g^m(n_u^m)$ be the indexes of the items to be masked where $n_u^m = \lfloor \mu_m \cdot |s_u| \rfloor$ and $g^m(x) \in [1, |s_u|]$. An item $v_i$ is replaced with the mask item [m] if selected to be masked. The masked sequence is thus:

$$\begin{aligned} s_u^{\mathrm{mask}} = [&v_1^u, \cdots, v_{g^m(1)-1}^u, [\mathrm{m}], v_{g^m(1)+1}^u, \cdots, \\ &v_{g^m(n_u^m)-1}^u, [\mathrm{m}], v_{g^m(n_u^m)+1}^u, \ldots, v_{|s_u|}^u]. \end{aligned} \tag{3.2}$$

- **Random Reorder** (rord)**:** It first randomly selects a continuous sub-sequence $[v_i^u, v_{i+1}^u, \ldots, v_{i+l_r}^u]$ of length $l_r = \lfloor \mu_r * |s_u| \rfloor$ $(0 \leq \mu_r \leq 1)$. It then randomly shuffles the items in the sub-sequence. Suppose the reordered items, sorted by new positions, are $[\tilde{v}_i^u, \ldots, \tilde{v}_{i+l_r}^u]$. The reordered sequence is thus:

$$s_u^{\mathrm{rord}} = [v_1^u, \cdots, v_{i-1}^u, \tilde{v}_i^u, \tilde{v}_{i+1}^u, \cdots, \tilde{v}_{i+l_r}^u, v_{i+l_r+1}^u, \cdots v_{|s_u|}^u]. \tag{3.3}$$

- **Random Retrieval** (rtrl)**:** This operation randomly selects another user sequence $s_{u'}$ that shares the same target (or next) item as the input sequence $s_u$, i.e., $v_*^u = v_*^{u'}$. The retrieved sequence is thus:

$$s_u^{\mathrm{rtrl}} = s_{u'}, s.t.\ v_*^u = v_*^{u'} \tag{3.4}$$

**CL4SRec Method.** Consider a set of users in a batch $U_B = \{u_1, u_2, \ldots, u_{|U_B|}\}$. The loss function of CL4SRec is:

$$\mathcal{L}_{CL4SRec} = \sum_{u \in U_B} \mathcal{L}_{rec}(s_u) + \lambda \mathcal{L}_{cl}(s_u^{a_i}, s_u^{a_j}). \tag{3.5}$$

where $\mathcal{L}_{rec}(s_u)$ and $\mathcal{L}_{cl}(s_u^{a_i}, s_u^{a_j})$ are the recommendation loss and self-supervised contrastive loss respectively. $s_u^{a_i}$ and $s_u^{a_j}$ are positive views of original user sequence $s_u$ after applying augmentations $a_i$ and $a_j$ respectively. $a_i$ and $a_j$ are sampled from $\{\mathrm{crop}, \mathrm{mask}, \mathrm{rord}\}$. We denote the positive view pairs for the users in the batch $B$ as $S_B = \{s_{u_1}^{a_1}, s_{u_1}^{a_2}, s_{u_2}^{a_1}, s_{u_2}^{a2}, \cdots, s_{u_{|B|}}^{a_1}, s_{u_{|B|}}^{a_2}\}$. Thus, the recommendation loss for the user $u$ can be formulated as:

$$\mathcal{L}_{rec}\left(s_u\right) = -\log \frac{\exp\left(\operatorname{sim}\left(h_u, h_{v_*^u}\right)\right)}{\exp\left(\operatorname{sim}\left(h_u, h_{v_*^u}\right)\right) + \sum_{v^- \in V^-} \exp\left(\operatorname{sim}\left(h_u, h_{v^-}\right)\right)} \tag{3.6}$$

where $V^- = V - \{v_*^u\}$, and $h_{v^-}$ are the representations of the sequence $s_u$, the next item $v_u^*$, and a negative item $v^-$ respectively. The contrastive loss is:

$$\mathcal{L}_{cl}\left(s_u^{a_i}, s_u^{a_j}\right) = -\log \frac{\exp\left(\operatorname{sim}\left(h_u^{a_i}, h_u^{a_j}\right)\right)}{\exp\left(\operatorname{sim}\left(h_u^{a_i}, h_u^{a_j}\right)\right) + \sum_{s^- \in S_u^-} \exp\left(\operatorname{sim}\left(h_u^{a_i}, h^-\right)\right)}, \tag{3.7}$$

where $h_u^{a_i}$ and $h_u^{a_j}$ are the representations of $s_u$ after augmentations $a_i$ and $a_j$ respectively. $S_u^-$ denotes a set of negative sequences defined by $S_u^- = S_B - \{s_u^{a1}, s_u^{a2}\}$. $s^-$and $h^-$ denote a sequence that does not belong to $u$ in the batch $B$ and its representation respectively.

**DuoRec Method.** Given a user sequence $s_u$, we randomly sample a *retrieved-positive view* from other users' sequences that share the same next item $v_*^u$. We denote all user sequences and their corresponding retrieved-positive views by $S = \{s_{u_1}, s_{u_1}^{\mathrm{rtrl}}, s_{u_2}, s_{u_2}^{\mathrm{rtrl}}, \cdots, s_{u_{|B|}}, s_{u_{|U|}}^{\mathrm{rtrl}}\}$. In DuoRec, the representations of each user sequence $s_u$ and its retrieved-positive view $s_u^{\mathrm{rtrl}}$ are learned to be close to each other but far from other user sequences and their retrieved-positive views denoted by $S_u^- = S - \{s_u, s_u^{\mathrm{rtrl}}\}$.

The loss function of DuoRec consists of both recommendation loss and supervised contrastive loss functions:

$$\mathcal{L}_{DuoRec} = \sum_{u \in U_B} \mathcal{L}_{rec}(s_u) + \lambda \mathcal{L}_{sl}(s_u) \tag{3.8}$$

$$
\mathcal{L}_{sl}\left(s_u\right) = -\left(\log \frac{\exp\left(\operatorname{sim}\left(h_u, h_u^{\mathrm{rtrl}}\right)/\tau\right)}{\exp\left(\operatorname{sim}\left(h_u, h_u^{\mathrm{rtrl}}\right)/\tau\right) + \sum_{s^- \in S_u^-} \exp\left(\operatorname{sim}\left(h_u, h^-\right)/\tau\right)} + \right.
$$
$$
\left. \log \frac{\exp\left(\operatorname{sim}\left(h_u^{\mathrm{rtrl}}, h_u\right)/\tau\right)}{\exp\left(\operatorname{sim}\left(h_u^{\mathrm{rrrl}}, h_u\right)/\tau\right) + \sum_{s^- \in S_u^-} \exp\left(\operatorname{sim}\left(h_u^{\mathrm{rtrl}}, h^-\right)/\tau\right)}\right) \tag{3.9}
$$

where $\tau$ is the temperature ratio.

### 3.3.3 Experiment for Important Item Evaluation

As shown in Figure 3.1, random augmentation is prone to generate false positive pairs that possibly degrade the quality of learned representations. To evaluate this claim, we conduct an experiment comparing CL4SRec using the vanilla random augmentation operations and augmentation operations that are aware of important items. Our goal is to evaluate if the latter can contribute to better recommendation performance, suggesting that the item importance-aware approach generates higher quality representations.

To verify this assumption empirically, we construct a synthetic dataset[2], which provides ground truth of important items in every user sequence. Specifically, the dataset consists of 500 user sequences each with 10 historical items and 3 additional items at the end serving as the next-items. Among the historical items are 3 important items shared by the 3 next-items to be used for training, validation and test respectively.

We then experiment CL4SRec on this synthetic dataset with two types of mask operations to generate positive views. Each of them masks 4 historical items as follows: (i) *random masking* that randomly masks 4 historical items (4 is empirically chosen); and (ii) *oracle based masking* that masks only unimportant items of the user sequence.

As shown in Table 3.1, CL4SRec using oracle-based masking substantially

[2]Details of the synthetic data is available at https://github.com/demoleiwang/EC4SRec.

Table 3.1: Results of CL4SRec on synthetic dataset with ground truth important items.

| Masking Op. | Random | Oracle-based |
|---|---|---|
| HR@3 | 0.3560 | 0.5180 |
| NDCG@3 | 0.2573 | 0.3645 |

outperforms that using random masking by both HitRate@3 and NDCG@3. The former achieves more than 40% higher NDCG@3 than the latter. This motivates us to determine important items for effective augmentation and contrastive learning in sequential recommendation.

## 3.4 Explanation-Guided Contrastive Learning Framework for Sequential Recommendation

Our proposed **Explanation guided Contrastive Learning Framework for Sequential Recommendation** (**EC4SRec**), as shown in Figure 3.2, consists of a **sequence encoder** to represent a given user's sequence of historical items $s_u$ into a vector representation $h_u$ which is in turn matched with items from a common pool by a **next-item predictor** which returns the next recommended item.

Unlike the existing contrastive learning methods to sequential recommendation (e.g., CL4SRec, DuoRec), EC4SRec utilizes an **explanation method** at scheduled epoch(es) to determine the importance of each $s_u$'s items for an input user sequence with next-item returned by the sequence encoder and next-item predictor. Next, the **explanation guided augmentation** will utilize the item importance scores to generate positive and negative views of user sequences for further training the sequence encoder and next-item predictor. The right of Figure 3.2 shows the different loss and recommendation loss functions that are used to train the models under different explanation-guided contrastive learning methods.

### 3.4.1 Explanation Guided Importance Scores

The schedule of explanation method updating the item importance scores, also known as *update schedule*, is controlled by a hyperparameter $p$. For a model training with a total of $N$ epoches, we schedule the updates to be at epoch $l \cdot \lfloor \frac{N}{p+1} \rfloor$ for $1 \leq l \leq p$. For example, for $p = 3$ and $N = 100$, updates will scheduled at epochs $25$, $50$, and $75$. At each scheduled epoch for updating explanation methods, we will use local explanation methods to derive importance scores for user sequences. These importance scores will be used to derive explanation augmented positive or negative views for each user sequence. For epochs before the first scheduled update (i.e., 1 to $\lfloor \frac{N}{p+1} \rfloor - 1$), EC4SRec can adopt any reasonably good sequential recommendation model (e.g., CL4SRec or DuoRec) to train the initial sequence encoder and next-item predictor. In our experiments, we combine the losses of CL4SRec and DuoRec, i.e., $\sum_{u \in U} \mathcal{L}_{rec}(s_u) + \lambda \mathcal{L}_{cl}(s_u) + \lambda \mathcal{L}_{sl}(s_u)$, to train the initial model. During inference, we only need to feed the input user sequence to the sequence encoder which generates the sequence representation for next-item predictor to recommend the next-item.

The most important items in the sequential recommendation models are those that contribute the most to the prediction. General explanation methods, such as Saliency Maps [121], Integrated Gradient [126], and Occlusion [170], are agnostic to sequential recommendation algorithms, such as GRU4Rec [50], Caser [130], and SASRec [58]. To obtain explanation-guided importance scores for each item in the user sequence, we feed the input user sequence $s_u = [v_1^u, v_2^u, \cdots, v_{|s_u|}^u]$, the sequential encoder $SeqRec$, and its prediction probability for any next item $y_u$ into any model-agnostic explanation method $Expl(\cdot)$, which determine the **importance scores** of items in $s_u$ as $\text{score}(s_u) = Expl(y_u, s_u, SeqRec)$, where $\text{score}(s_u) = [\text{score}(v_1^u), \text{score}(v_2^u), \ldots, \text{score}(v_{|s_u|}^u)]$ and $\text{score}(v_i^u)$ denotes the importance score of item $v_i^u$.

Each explanation method employs a different strategy to identify these important items. Each method computes the importance score, $\text{score}\,(v_i^u)$, for each item $v_i^u$ to

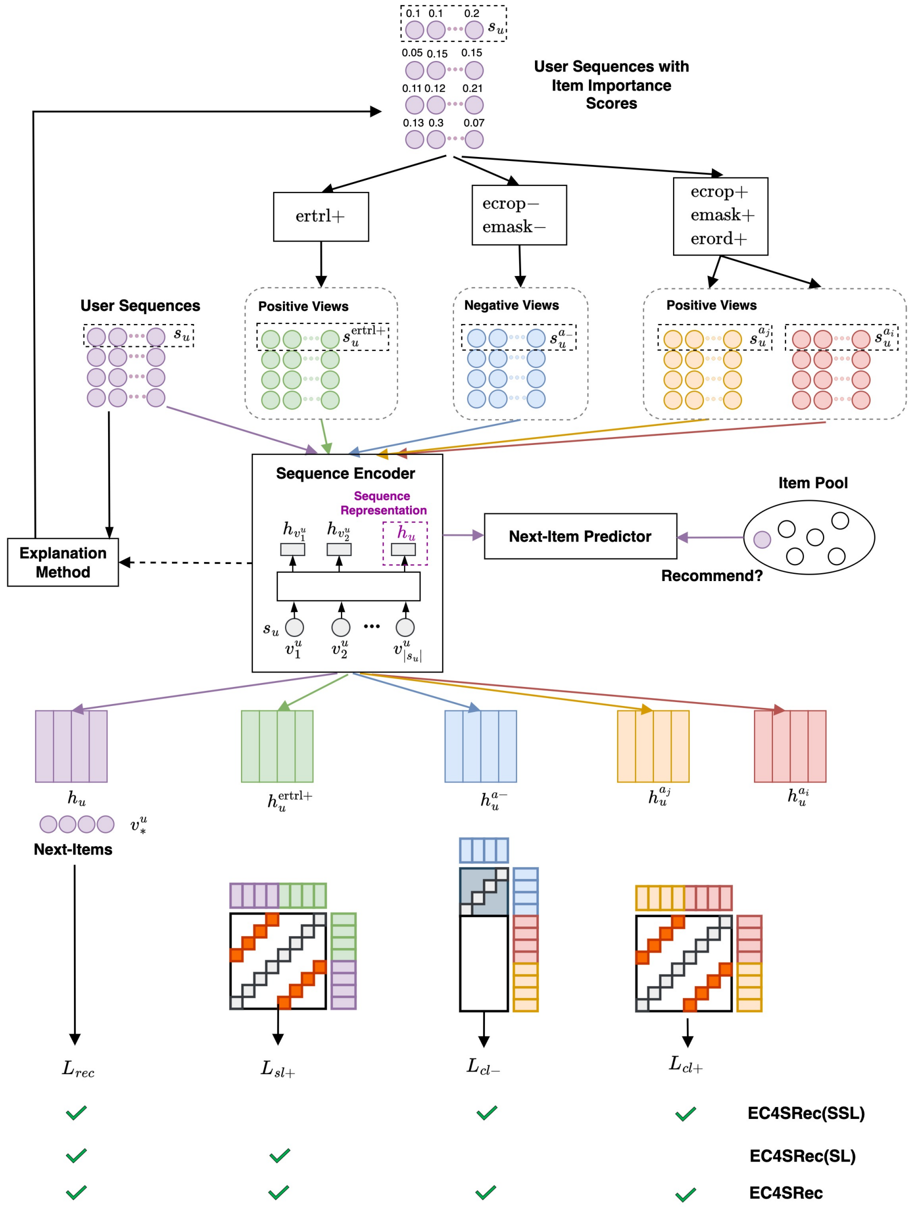


Figure 3.2: Proposed EC4SRec Framework.

help find the important items that contribute to a model's prediction.

**Importance Score using Saliency Map.** Saliency maps are used to determine which parts of the input sequence a model focuses on when making a decision. The basic idea is to compute the gradient of the output category with respect to the input sequence. This shows how much each item in the input sequence contributes to the final decision. The importance score of each item can be represented by the following formula:

$$\operatorname{score}\left(v_i^u\right)=\left\|\frac{\partial v_{|s_u|+1}^u}{\partial e_{v_i^u}}\right\|. \tag{3.10}$$

where:

- $v_{|s_u|+1}^u$ is the output of the model and $s_u$ is the input sequence.
- $e_{v_i^u}$ is the embedding vector of item $v_i^u$.
- $\frac{\partial v_{|s_u|+1}^u}{\partial e_{v_i^u}}$ is the gradient of the output with respect to the item $v_i^u$.

**Importance Score using Occlusion.** Occlusion involves systematically blocking different items of the input sequence and observing the impact on the model's output. This method helps to understand which items of the input sequence are crucial for the model's prediction by noting how the prediction changes when items of the input are obscured. The impact can be quantified by the change in output score. This method typically involves:

- Occluding item of the input sequence using a patch.
- Calculating the output of the model with the occluded item sequence.
- Observing the change in the output.

This can be summarized by:

$$\operatorname{score}(v_i^u)=\mathrm{O}(s_{u,i})=y(s_u)-y\left(s_{u,i}\right). \tag{3.11}$$

Where:

- $\mathrm{O}(s_{u,i})$ is the occlusion score.
- $y(s_u)$ is the output score with the full input sequence.
- $y\left(s_{u,i}\right)$ is the output score with part of the input sequence occluded by patch $i$.

**Importance Score using Attention Mechanisms.** Attention mechanisms, often used in sequence modeling tasks (like language processing), allow the model to weigh different items of the input sequence differently, focusing more on items deemed more important. The typical formula for a simple attention mechanism can be:

$$\text{score}(v_i^u) = \text{Attention}(v_{|s_u|}^u, v_i^u) = \frac{h_{v_{|s_u|}^u} h_{v_i^u}}{\sum_j^{|s_u|} h_{v_{|s_u|}^u} h_{v_j^u}}. \tag{3.12}$$

Where:

- $v_{|s_u|}^u$ and $v_i^u$ are the last item (query) and a specific item (key), respectively.
- $h_{v_{|s_u|}^u}$ and $h_{v_i^u}$ are the latent vectors of last item (query) and a specific item (key), respectively.
- $h_{v_{|s_u|}^u} h_{v_i^u}$ computes the alignment score between query and key.

$score(v_i^u)$ returns a value in $[0, 1]$ indicating how important is $v_i^u$ in the sequence $s_u$ for a specific given sequential recommendation algorithm. As $\sum_i score(v_i^u) = 1$, the importance score is relative and comparable only among items of the same sequence.

### 3.4.2 Explanation Guided Augmentation

We propose five explanation guided augmentation operations, three for generating positive views and two for generating negative views. While some of these operations are extensions of random augmentation, the operations for generating negative views (that is, $\text{ecrop}-$ and $\text{emask}-$) are unique to EC4SRec as both CL4SRec and DuoRec consider augmentations for positive views only. Our experiment results in Section 3.5.3 also show that these negative views can substantially improve recommendation accuracy.

- **Explanation Guided Crop for Positive and Negative View (**$\text{ecrop}+$**,** $\text{ecrop}-$**):** To perform $\text{ecrop}+$ (or $\text{ecrop}-$) on $s_u$, we select the $k$ (or $|s_u| - k$) items with the lowest (or highest) by importance score to be removed to generate the positive (or negative) view. $k$ is defined by $\lfloor \mu_e \cdot |s_u| \rfloor$ where $\mu_e$ $(0 < \mu_e \leq 1)$. Let $[v_{i_1}^u, \cdots, v_{i_k}^u]$ denote the sub-sequence of $k$ items in $s_u$ with the lowest importance scores. The explanation guided cropped positive and negative views are defined as:

$$s_u^{\text{ecrop}+} = s_u - [v_{i_1}^u, \cdots, v_{i_k}^u], \quad s_u^{\text{ecrop}-} = [v_{i_1}^u, \cdots, v_{i_k}^u]. \tag{3.13}$$

- **Explanation Guided Mask for Positive or Negative View (**$\text{emask}+$**,**$\text{emask}-$**):** To perform $\text{emask}+$ on $s_u$, we select the $k$ items with the lowest importance scores to be masked. Let $[v_{i_1}^u, \cdots, v_{i_k}^u]$ denote the sub-sequence of $k$ items in $s_u$ with the lowest importance scores. The explanation guided masked positive view is then defined as:

$$s_u^{\text{emask}+} = [v_1^u, \cdots, v_{i_1-1}^u, [\text{m}], v_{i_1+1}^u, \cdots, v_{i_k-1}^u, [\text{m}], v_{i_k+1}^u, \cdots, v_{|s_u|}^u] \tag{3.14}$$

  The explanation guided masked negative view $s_u^{\text{emask}-}$ is defined in a similar way except that the $k$ items to be masked are those with highest importance scores.

- **Explanation Guided Reorder for Positive View (**$\text{erord}+$**):** Let $[v_{i_1}^u, \cdots, v_{i_k}^u]$ denote the sub-sequence of $k$ items in $s_u$ with the lowest importance scores $(i_1 < i_2 < \cdots < i_k)$. We randomly reorder these items. Suppose the reordered items, sorted by new positions, are $[\tilde{v}_{i_1}^u, \cdots, \tilde{v}_{i_k}^u]$. The reordered positive view can be formulated as:

$$s_u^{\text{erord}+} = [v_1^u, \cdots, v_{i_1-1}^u, \tilde{v}_{i_1}^u, v_{i_1+1}^u, \cdots, v_{i_k-1}^u \tilde{v}_{i_k}^u, v_{i_k+1}^u, \cdots, v_{|s_u|}^u]. \tag{3.15}$$

  We leave out explanation guided reorder operation for negative views as it is not likely to generate discriminative negative views.

- **Explanation Guided Retrieval for Positive View** ($\text{ertrl+}$)**:** Like random retrieval, we first define the candidate sequences that share the same target (or next) item as the original user sequence $s_u$ as: $S_u^c = \{s_{u_1}, s_{u_2}, \ldots, s_{u_{|S_u^c|}}\}$. That is, $v_*^{u_k} = v_*^u$, $s_{u_k} \in |S_u^c|$, and $u_k \neq u$. Next, we compute the probability for each sequence in $S_u^c$ using importance scores:

$$P(s_{u_k}) = \frac{util(s_{u_k})}{\sum_{s_{u_j} \in S_u^c} util(s_{u_j})}. \tag{3.16}$$

  where

$$util(s_{u_k}) = \frac{|s_u \cap s_{u_k}|}{|s_u \cup s_{u_k}|} \sum_{v \in s_u \cap s_{u_k}} \text{score}(v) \tag{3.17}$$

  We then sample the explanation guided retrieved sequence user sequence $s_u^{\text{ertrl+}}$ from $S_u^c$ with the probability distribution.

### 3.4.3 Explanation Guided Contrastive Learning

Based on the EC4SRec framework, we can derive different proposed models depending the type of explanation guided contrastive learning used for model training. In the following, we introduce three proposed models based on explanation guided self-supervised contrastive learning, explanation guided supervised contrastive learning, and combined explanation guided contrastive learning.

- **Explanation Guided Self-Supervised Learning**.

  This model can be seen as an extension of CL4SRec with explanation guided augmentation operations generating both positive and negative views for contrastive learning. The loss function consists of three components: (i) *recommendation loss*, (ii) *contrastive loss for explanation guided positive views*, and (iii) *contrastive loss* *for explanation guided negative views*:

$$\mathcal{L}_{EC4SRec(SSL)} = \sum_{u \in U_B} \mathcal{L}_{rec}(s_u) + (\lambda_{cl+}\mathcal{L}_{cl+}(s_u) + \lambda_{cl-}\mathcal{L}_{cl-}(s_u)). \tag{3.18}$$

The $\mathcal{L}_{rec}(s_u)$ here has been defined earlier in Equation 3.6. Let $A^+ = \{a_{\text{ecrop}+}, a_{\text{emask}+}, a_{\text{erord}+}\}$ and $A^- = \{a_{\text{ecrop}-}, a_{\text{emask}-}\}$. To obtain $\mathcal{L}_{cl+}(s_u)$, we generate two positive views $s_u^{a_i}$ and $s_u^{a_j}$ by sampling $a_i$ and $a_j$ ($a_i \neq a_j$) from $A^+$ and applying them on $s_u$. We repeat this for all other users and obtain a set of a set of positive views from *all* users denoted as $S^+$. Let $S_u^+$ be $\{s_u^{a_i}, s_u^{a_j}\}$. To get the representations of $s_u^{a_i}$ and $s_u^{a_j}$ closer to each other but farther away from other users' positive views, we define:

$$\mathcal{L}_{cl+}(s_u) = -\log \frac{\exp\left(\text{sim}\left(h_u^{a_i}, h_u^{a_j}\right)\right)}{\exp\left(\text{sim}\left(h_u^{a_i}, h_u^{a_j}\right)\right) + \sum_{s_{u'}^a \in S^+ - S_u^+} \exp\left(\text{sim}\left(h_u^{a_i}, h_{u'}^{a}\right)\right)}. \tag{3.19}$$

To obtain $\mathcal{L}_{cl-}(s_u)$, we generate a negative view $s_u^{a-}$ by applying an augmentation operator $a-$, sampled from $A^-$, on $s_u$. Here, we would like this negative view to be closer to other users' negative views (as we do not need distinctive representations for these negative views) and farther away from the all users' positive views, borrowing a similar idea from [59]. Let $S^-$ denote the set of negative views after repeating the above on *all* users. We define:

$$\mathcal{L}_{cl-}(s_u) = -\frac{1}{|S^-| - 1} \sum_{s_{u'}^a \in S^- - \{s_u^{a-}\}} \log \frac{\exp\left(\text{sim}\left(h_u^{a-}, h_{u'}^{a}\right)\right)}{\sum_{s \in S^+ \cup \{s_{u'}^a\}} \exp\left(\text{sim}\left(h_u^{a-}, h\right)\right)}, \tag{3.20}$$

where $h$ is the representation of the sequence $s$. By setting $\beta = 0$, we can obtain a model variant that considers positive views only.

- **Explanation Guided Supervised Contrastive Learning**.

This model extends DuoRec to use explanation guided augmentation. The loss function is:

$$\mathcal{L}_{EC4SRec(SL)} = \sum_{u \in U_B} \mathcal{L}_{rec}(s_u) + \lambda \mathcal{L}_{sl+}(s_u), \tag{3.21}$$

where

$$\begin{aligned} &\mathcal{L}_{sl+}(s_u) = \\ &- \left( \log \frac{\exp\left(\text{sim}\left(h_u, h_u^{\text{ertrl}+}\right)/\tau\right)}{\exp\left(\text{sim}\left(h_u, h_u^{\text{ertrl}+}\right)/\tau\right) + \sum_{s^- \in S_u^-} \exp\left(\text{sim}\left(h_u, h^-\right)/\tau\right)} + \right. \\ &\left. \log \frac{\exp\left(\text{sim}\left(h_u^{\text{ertrl}+}, h_u\right)/\tau\right)}{\exp\left(\text{sim}\left(h_u^{\text{ertrl}+}, h_u\right)/\tau\right) + \sum_{s^- \in S_u^-} \exp\left(\text{sim}\left(h_u^{\text{ertrl}+}, h^-\right)/\tau\right)} \right). \end{aligned} \tag{3.22}$$

$h_u^{ertrl+}$ is the representation of the augmented sequence for the user $u$ generated by explanation guided retrieval operation, i.e., $\text{ertrl}+$.

- **Combined Explanation Guided Contrastive Learning**.

  To leverage both self-supervised contrastive learning and supervised contrastive learning, we combine the two types of contrastive learning losses as:

$$\mathcal{L}_{EC4SRec} = \sum_{u \in U_B} \mathcal{L}_{rec}(s_u) + \lambda_{cl+} \mathcal{L}_{cl+}(s_u) + \lambda_{cl-} \mathcal{L}_{cl-}(s_u) + \lambda_{sl+} \mathcal{L}_{sl+}(s_u). \tag{3.23}$$

## 3.5 Experiment

### 3.5.1 Experimental Settings

**Datasets and Data Preprocessing**. We conduct experiments on four widely used real-world datasets, i.e., Beauty, Clothing, Sports, and ML-1M (Movielens 1M). The first three are from Amazon[3] [93], and ML-1M[4] [43] is a very large benchmark dataset for movie recommendation. Following the previous works [58, 184, 156,

[3] http://jmcauley.ucsd.edu/data/amazon/
[4] https://grouplens.org/datasets/movielens/1m/

Table 3.2: Dataset Statistics After Preprocessing.

| Dataset | Beauty | Clothing | Sports | ML-1M |
|---|---|---|---|---|
| Users | 22,363 | 39,387 | 35,598 | 6,041 |
| Items | 12,101 | 23,033 | 18,357 | 3,417 |
| User-item Interactions | 198,502 | 278,677 | 296,337 | 999,611 |
| Avg Sequence Length | 8.9 | 7.1 | 8.3 | 165.5 |
| Sparsity | 99.93% | 99.97% | 99.95% | 95.16% |

Table 3.3: Performance comparison with sequential recommendation methods without contrastive learning. (The best and second best results are boldfaced and underlined.

| Dataset | Metric | BPR-MF | GRU4Rec | Caser | SASRec | BERT4Rec | $S^3Rec_{MIP}$ | **EC4SRec** |
|---|---|---|---|---|---|---|---|---|
| Beauty | HR@5 | 0.0120 | 0.0164 | 0.0191 | 0.0365 | 0.0193 | 0.0327 | **0.0569** |
| | HR@10 | 0.0299 | 0.0365 | 0.0335 | 0.0627 | 0.0401 | 0.0591 | **0.0862** |
| | NDCG@5 | 0.0065 | 0.0086 | 0.0114 | 0.0236 | 0.0187 | 0.0175 | **0.0364** |
| | NDCG@10 | 0.0122 | 0.0142 | 0.0160 | 0.0281 | 0.0254 | 0.0268 | **0.0458** |
| Clothing | HR@5 | 0.0067 | 0.0095 | 0.0049 | 0.0168 | 0.0125 | 0.0163 | **0.0209** |
| | HR@10 | 0.0094 | 0.0165 | 0.0092 | 0.0272 | 0.0208 | 0.0237 | **0.0320** |
| | NDCG@5 | 0.0052 | 0.0061 | 0.0029 | 0.0091 | 0.0075 | 0.0101 | **0.0119** |
| | NDCG@10 | 0.0069 | 0.0083 | 0.0043 | 0.0124 | 0.0102 | 0.0132 | **0.0155** |
| Sports | HR@5 | 0.0092 | 0.0137 | 0.0121 | 0.0218 | 0.0176 | 0.0157 | **0.0331** |
| | HR@10 | 0.0188 | 0.0274 | 0.0204 | 0.0336 | 0.0326 | 0.0265 | **0.0514** |
| | NDCG@5 | 0.0053 | 0.0096 | 0.0076 | 0.0127 | 0.0105 | 0.0098 | **0.0203** |
| | NDCG@10 | 0.0085 | 0.0137 | 0.0103 | 0.0169 | 0.0153 | 0.0135 | **0.0262** |
| ML-1M | HR@5 | 0.0164 | 0.0763 | 0.0816 | 0.1087 | 0.0733 | 0.1078 | **0.1672** |
| | HR@10 | 0.0354 | 0.1658 | 0.1593 | 0.1904 | 0.1323 | 0.1952 | **0.2533** |
| | NDCG@5 | 0.0097 | 0.0385 | 0.0372 | 0.0638 | 0.0432 | 0.0616 | **0.1102** |
| | NDCG@10 | 0.0158 | 0.0671 | 0.0624 | 0.0910 | 0.0619 | 0.0917 | **0.1380** |

106], we remove repeated items, and preprocess the four datasets with the 5-core strategy (i.e., removing users and items with fewer than 5 interactions). The dataset statistics are summarized in Table 3.2. The datasets are very sparse as their sparsity indices (defined by $1 - \frac{\text{num. of interactions}}{\text{num. of users} \cdot \text{num. of items}}$) are very high.

**Evaluation Protocols**. Following [156, 106], we use the last interacted item of each user sequence for test, the second last item for validation, and all the earlier items for training. The predicted next-item come from the pool of *all* items without any candidate filter. We employ two performance metrics, *Hit Ratio at $k$ (HR@$k$)* and *Normalized Discounted Cumulative Gain at $k$ (NDCG@$k$)*, which are widely used in previous work [58, 184, 156, 106]. We report the average of results with running 3 times with 3 random seeds.

Table 3.4: Performance comparison with sequential recommendation methods with contrastive learning. (The best and second best results are boldfaced and underlined. *: significant improvement of EC4SRec(SSL) over CL4SRec with $p$-value$= 0.05$. **: significant improvement of EC4SRec(SL) over DuoRec with $p$-value$= 0.01$.)

| Dataset | Metric | CL4SRec | EC4SRec(SSL)** | DuoRec | EC4SRec(SL)* | **EC4SRec** |
|---|---|---|---|---|---|---|
| Beauty | HR@5 | 0.0495 | 0.0569 | 0.0548 | **0.0585** | <u>0.0569</u> |
| | HR@10 | 0.0810 | 0.0853 | 0.0832 | **0.0867** | <u>0.0862</u> |
| | NDCG@5 | 0.0299 | 0.0358 | 0.0345 | <u>0.0361</u> | **0.0364** |
| | NDCG@10 | 0.0401 | 0.0450 | 0.0436 | <u>0.0455</u> | **0.0458** |
| Clothing | HR@5 | 0.0187 | 0.0201 | 0.0196 | <u>0.0205</u> | **0.0209** |
| | HR@10 | 0.0305 | <u>0.0314</u> | 0.0296 | 0.0311 | **0.0320** |
| | NDCG@5 | 0.0104 | 0.0113 | 0.0112 | <u>0.0115</u> | **0.0119** |
| | NDCG@10 | 0.0142 | <u>0.0149</u> | 0.0144 | <u>0.0149</u> | **0.0155** |
| Sports | HR@5 | 0.0277 | <u>0.0323</u> | 0.0310 | 0.0317 | **0.0331** |
| | HR@10 | 0.0455 | <u>0.0497</u> | 0.0480 | 0.0491 | **0.0514** |
| | NDCG@5 | 0.0167 | <u>0.0201</u> | 0.0191 | 0.0194 | **0.0203** |
| | NDCG@10 | 0.0224 | <u>0.0256</u> | 0.0246 | 0.0249 | **0.0262** |
| ML-1M | HR@5 | 0.1583 | **0.1699** | 0.1672 | <u>0.1682</u> | 0.1672 |
| | HR@10 | 0.2423 | **0.2543** | 0.2507 | 0.2526 | <u>0.2533</u> |
| | NDCG@5 | 0.0996 | 0.1095 | 0.1076 | **0.1104** | <u>0.1102</u> |
| | NDCG@10 | 0.1267 | 0.1368 | 0.1345 | <u>0.1375</u> | **0.1380** |

**Baselines**. We compare EC4SRec(SSL), EC4SRec(SL) and EC4SRec with the following three groups of baseline methods:

- **Non-sequential recommendation method**: We use BRP-MF [113], a popular matrix factorization model.
- **Sequential recommendation methods without contrastive learning:** These methods include GRU4rec [50], a RNN-based method; Caser [130], a CNN-based method; two self-attention based methods SASRec [58] and BERT4Rec [125]; and a self-supervised learning method $S^3Rec_{MIP}$ [184].
- **Sequential recommendation methods with contrasitve learning**: These include CL4SRec [156] and DuoRec [106].

**Implementation Details.** For BPR-MF and $S^3Rec_{MIP}$, we use results reported in the CL4SRec paper. We implemented the baselines GRU4Rec, Caser, SASRec, and BERT4Rec using the RecBole library [5] [182]. For CL4SRec and DuoRec, we made

[5] https://github.com/RUCAIBox/RecBole

Table 3.5: Results of EC4SRec with different Sequential Recommendation Backbones.

| Backbone | | Beauty | | Clothing | | Sports | |
|---|---|---|---|---|---|---|---|
| | | NDCG@5 | NDCG@10 | NDCG@5 | NDCG@10 | NDCG@5 | NDCG@10 |
| GRU4Rec | CL4SRec | 0.0270 | 0.0341 | 0.0065 | 0.0089 | 0.0154 | 0.0200 |
| | EC4SRec(SSL) | 0.0314 | 0.0382 | 0.0082 | <u>0.0109</u> | 0.0167 | 0.0213 |
| | DuoRec | 0.0318 | 0.0388 | 0.0078 | 0.0102 | 0.0163 | 0.0207 |
| | EC4SRec(SL) | <u>0.0327</u> | <u>0.0401</u> | <u>0.0086</u> | 0.0108 | <u>0.0173</u> | <u>0.0218</u> |
| | EC4SRec | **0.0332** | **0.0412** | **0.0089** | **0.0115** | **0.0182** | **0.0233** |
| Caser | CL4SRec | 0.0108 | 0.0157 | 0.0036 | 0.0049 | 0.0071 | 0.0096 |
| | EC4SRec(SSL) | 0.0137 | 0.0189 | <u>0.0039</u> | 0.0055 | 0.0088 | 0.0121 |
| | DuoRec | 0.0129 | 0.0183 | 0.0031 | 0.0046 | 0.0082 | 0.0110 |
| | EC4SRec(SL) | <u>0.0161</u> | <u>0.0218</u> | <u>0.0039</u> | <u>0.0056</u> | <u>0.0097</u> | <u>0.0127</u> |
| | EC4SRec | **0.0172** | **0.0232** | **0.0041** | **0.0060** | **0.0105** | **0.0139** |

some changes to the codes provided by the authors of DuoRec to mainly correct some bugs. Our CL4SRec and DuoRec results are generally similar to that reported in the original works. For each baseline, we set the embedding dimension to be $64$ and keep all other hyper-parameter settings the same as those reported in their original papers. For EC4SRec and its variants, we use SASRec and Occlusion as the default backbone sequential recommendation method and explanation method respectively. The hyper-parameter settings, such as batch size, embedding dimension, number of layers, number of attention heads, follow those reported in CL4SRec. We tune $\mu_e$, a hyperparameter to control the proportion of important items in augmentation from $0.1$ to $0.9$ with step size $= 0.1$. We also tune the temperature $\tau$ within $[0.5, 1.0, 1.5]$, and the coefficients $\lambda_{cl+}$, $\lambda_{cl-}$, and $\lambda_{sl+}$ within $[0.1, 0.2, 0.3, 0.4, 0.5]$.

### 3.5.2 Overall Results

**EC4SRec versus Baselines**. As shown in Table 3.3 and Table 3.4, the overall experiment results show that:

- Our proposed EC4SRec and its variants consistently outperform the state-of-the-art methods, including the latest contrastive learning-based models CL4SRec and DuoRec, for all datasets by all metrics. Specifically, EC4SRec achieves 12.4% (4.9%) improvement over CL4SRec (DuoRec) on average across all datasets by all metrics. EC4SRec generally performs better than EC4SRec(SSL) and EC4SRec(SL). The above findings as well as the significant improvement of EC4SRec(SSL) over CL4SRec and EC4SRec(SL) over DuoRec demonstrate that self-supervised and supervised contrastive learning benefit substantially from explanation guided augmentation. Higher-quality positive views and negative views for contrastive leaning have clearly resulted in better user sequence representations.

- Among the baselines, non-sequential recommendation recommendation methods (i.e., BPR-MF) unexpectedly yield the worst performance. It indicates that the sequential patterns are important in this task. Among the sequential recommendation methods, SASRec and BERT4Rec consistently outperform GRU4Rec and Caser. It shows that self attention can model more complex patterns than left-to-right patterns.

- Consistent with earlier results in [156, 106], contrastive learning methods CL4SRec and DuoRec outperform $S^3Rec_{MIP}$ and SASRec. Our experiment also shows that the former also outperform BERT4Rec. The above demonstrates the the strength of contrastive learning. With supervised contrastive learning, DuoRec performs better than CL4SRec but the gap is reduced between EC4SRec(SL) and EC4SRec(SSL). This could be explained by the additional loss $\mathcal{L}_{cl-}$ added to EC4SRec(SSL).

**EC4SRec with Different Backbone Sequential Recommendation Methods.** Instead of using the default self-attention based backbone SASRec, we evaluate EC4SRec, its variants, CL4SRec and DuoRec using other backbones, namely RNN-based GRU4Rec and CNN-based Caser to study the impact of explanation guided augmentation and contrastive learning. Due to space constraint, we only show the result on three datasets. As shown in Table 3.5, the relative performance ranking between EC4SRec, EC4SRec(SSL), EC4SRec(SL), CL4SRec and DuoRec remains unchanged when using GRU4Rec and Caser backbones. EC4SRec still achieves the best performance using different backbones. EC4SRec(SSL) and EC4SRec(SL)

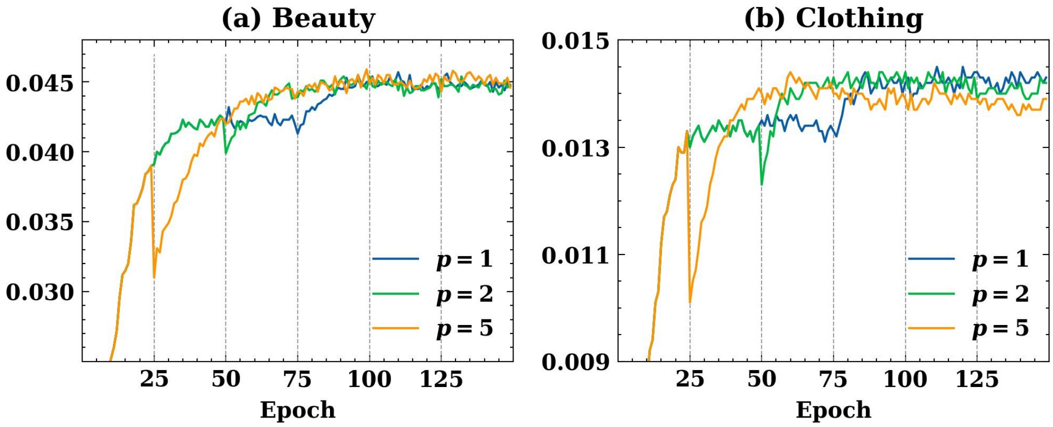


Figure 3.3: Changes of NDCG@5 for EC4SRec using different update schedules over 150 training epoches ($p$: number of importance score updates in training)

outperforms CL4SRec and DuoRec respectively. These encouraging results indicate the generalizability of the effectiveness of explanation guided approach.

### 3.5.3 Detailed Analysis

In this section, we conduct detailed analysis of EC4SRec and its variants. We show the results on Beauty and Clothing datasets only due to space constraint.

**Effect of Update Schedule of Important Scores.** As mentioned in Section 3.4, the parameter $p$ controls the number of item importance updates scheduled during the training of EC4SRec and its variants. First, we want to study how the updates affect the performance of these models during the training epoches. For illustration, we plot the NDCG@5 of EC4SRec only on validation data in Figure 3.3. With 150 training epoches, the update occurs only at epoch 75 for $p = 1$, at epoches 50 and 100 for $p = 2$, and at epoches 25, 50, 75, 100 and 125 for $p = 5$. The figure shows that EC4SRec experiences drops of NDCG@5 at the first update. This is because EC4SRec switches from random augmentation and a loss function combining that of CL4SRec and DuoRec to explanation guided augmentation and explanation guided contrastive loss at the first update. EC4SRec however recovers quickly and continues to improve until it converges. Interestingly, the drop in performance is not noticeable for subsequent scheduled updates.

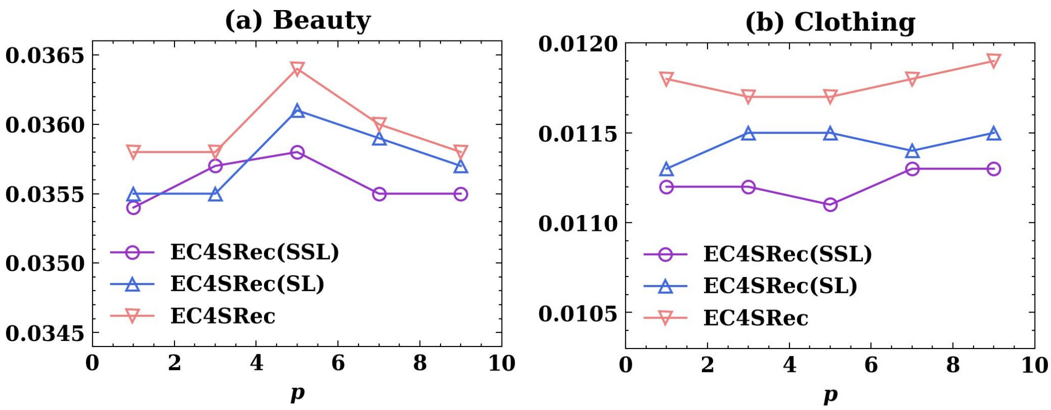


Figure 3.4: NDCG@5 Results with different update schedule settings ($p$: number of updates in model training).

We also show the effect of update schedule on the trained EC4SRec and variants when evaluated against test data in Figure 3.4. Generally, the NDCG@5 performance does not change much for different $p$ settings. $p = 5$ and $= 9$ yield best results for Beauty and Clothing respectively. As every update incurs additional overheads, there is clearly a trade-off between performance and efficiency when choosing the update schedule which we shall leave to future research.

**Ablation of Loss Functions.** We study the effect of different contrastive losses on EC4SRec performance by evaluating the model using different combinations of losses as shown in Figure 3.5. The figure shows the NDCG@5 of EC4SRec using recommendation loss $\mathcal{L}_{rec}$ and seven combinations of the three contrastive losses, $\mathcal{L}_{cl+}$, $\mathcal{L}_{cl-}$ and $\mathcal{L}_{sl+}$. For illustration, we conduct this ablation study on Beauty and include CL4SRec and DuoRec for comparison.

Figure 3.5 shows that EC4SRec with $\mathcal{L}_{cl+}$ and EC4SRec with $\mathcal{L}_{cl-}$ outperform CL4SRec. EC4SRec with $\mathcal{L}_{sl+}$ also outperforms DuoRec. These indicate that each of the 3 explanation guided contrastive losses can effectively improve performance. Moreover, combining them together can yield even higher performance with the exception of $\mathcal{L}_{sl+} + \mathcal{L}_{cl+}$ which could be explained by having only $\mathcal{L}_{cl+}$ without $\mathcal{L}_{cl-}$ does not help to improve representations much and may conflict with the supervised contrastive learning loss using retrieved positive views.

**Influence of Different Augmentation.** Our approach consists of four explana-

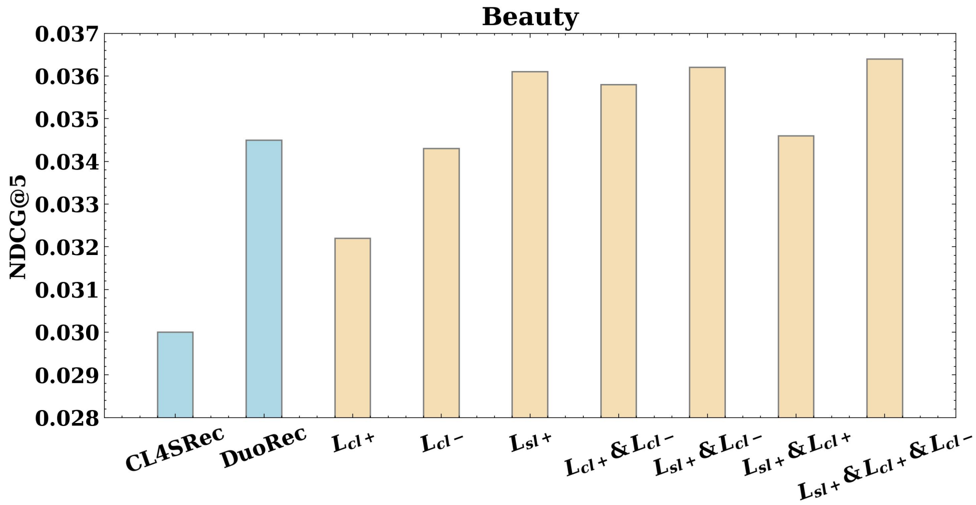


Figure 3.5: Ablation study of EC4SRec with different combinations of loss functions on Beauty dataset. (EC4SRec results are shown in yellow bars. As $\mathcal{L}_{rec}$ is included by default, EC4SRec(SSL) = EC4SRec with $\mathcal{L}_{cl+} + \mathcal{L}_{cl-}$; EC4SRec(SL) = EC4SRec with $\mathcal{L}_{sl+}$, and EC4SRec = one with all three losses.)

Table 3.6: NDCG@5 Results of EC4SRec(SSL), abbreviated by E(SSL), with the removal of augmentation operation on Beauty, Clothing and Sports.

| | Beauty | | Clothing | | Sports | |
|---|---|---|---|---|---|---|
| | CL4SRec | E(SSL) | CL4SRec | E(SSL) | CL4SRec | E(SSL) |
| None | 0.0299 | 0.0358 | 0.0104 | 0.0113 | 0.0167 | 0.0201 |
| −rord | 0.0307 | 0.0344 | 0.0103 | 0.0110 | 0.0169 | 0.0181 |
| −mask | 0.0311 | 0.0350 | 0.0101 | 0.0116 | 0.0169 | 0.0200 |
| −crop | 0.0282 | 0.0353 | 0.0086 | 0.0116 | 0.0147 | 0.0200 |

tion guided augmentations: ecrop, emask, erord, and ertrl. We earlier show that EC4SRec(SL) using explanation guided retrieval (i.e., ertrl) significantly outperforms DuoRec as shown in Table 3.4. In this section, we evaluate how EC4SRec performs when not using one of the three augmentation operations to investigate the effect of each augmentation operation.

As shown in Table 3.6, the recommendation accuracy drops substantially when any one of augmentations is removed. Besides, EC4SRec(SSL) consistently achieves better performance than CL4SRec even with one augmentation operation removed. It indicates the effectiveness of each proposed operation. Interestingly, for Clothing dataset, the removal of some augmentation operation can slightly improve the EC4SRec(SSL) performance.

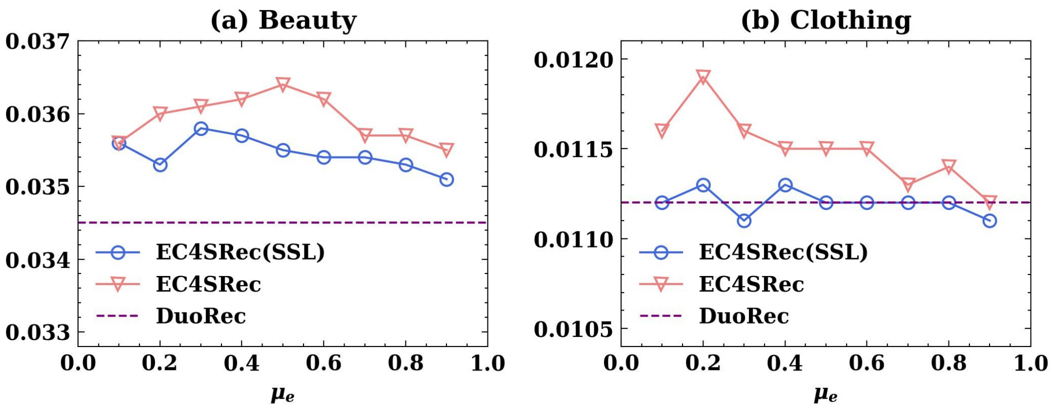


Figure 3.6: NDCG@5 of EC4SRec with different $\mu_e$ settings.

**Study of Hyper-Parameter $\mu_e$.** The hyper-parameter $\mu_e$ determines the number of items with highest scores would be augmented for positive views and negative views under explanation guided augmentation. In this study, we vary $\mu_e$ from 0.1 to 0.9 and show the NDCG@5 of EC4SRec and EC4SRec(SSL) on Beauty and Clothing datasets as shown in Figure 3.6. We observe that $\mu_e$ substantially affects the performance of EC4SRec and EC4SRec(SSL). For Beauty dataset, the NDCG@5 of EC4SRec changes from $0.0364$ when $\mu_e = 0.5$ to $0.0355$ when $\mu_e = 0.9$. Second, EC4SRec and EC4SRec(SSL) perform best on Beauty when $\mu_e = 0.5$ and $\mu_e = 0.3$ respectively. For Clothing dataset, the NDCG@5 of EC4SRec changes from $0.0119$ when $\mu_e = 0.2$ to $0.0112$ when $\mu_e = 0.9$. And both EC4SRec and EC4SRec(SSL) perform best on Clothing when $\mu_e = 0.2$. These findings indicate that EC4Srec and its variants have different optimal value $\mu_e$ on different datasets. Besides, even EC4SRec with the worst performing $\mu_e$ still outperforms DuoRec.

**Effect of Different Explanation Methods.** In our earlier results, we use Occlusion as the default explanation method. In this experiment, we evaluate EC4SRec and its variants using other explanation methods for comparison. Figure 3.7 shows the NDCG@5 results of the above models using Saliency, Occlusion, and Attention based explanation methods on Beauty and Clothing datasets. The NDCG@5 of CL4SRec and DuoRec are also shown as reference.

Figure 3.7 shows that Occlusion performs well in most cases. The three expla-

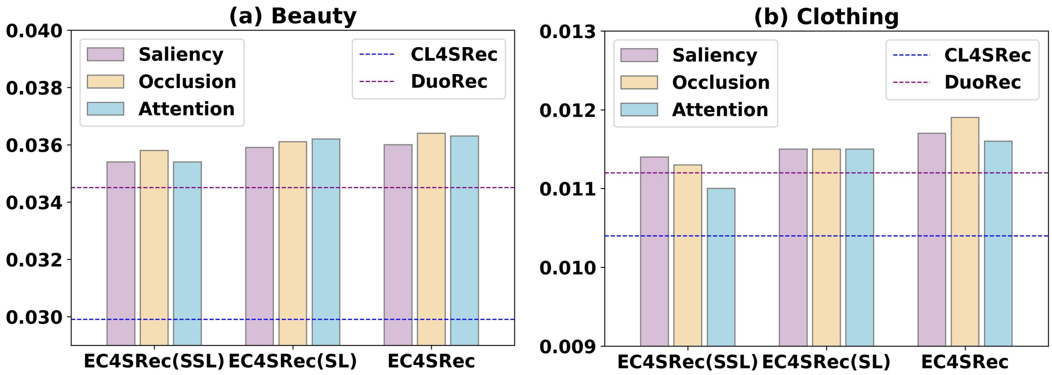


Figure 3.7: NDCG@5 using different explanation methods.

nation methods generally help EC4SRec and variants outperform CL4SRec and DuoRec except Attention which could not help EC4SRec(SSL) outperforms DuoRec. Considering robustness, performance, and efficiency, we prefer to use occlusion as the explanation method to get better views for contrastive learning.

## 3.6 Summary

In this chapter, we study how to utilize explanation methods to produce high-quality views for contrastive learning in sequential recommendation task. We propose a model-agnostic Explanation Guided Contrastive Learning for Sequential Recommendation (EC4SRec) framework. We introduce several proposed explanation-guided augmentations to generate positive and negative views of given user sequences and propose both self-supervised and supervised contrastive learning. Our extensive experiments on four real-world benchmark datasets demonstrate the effectiveness, generality, and flexibility of our proposed explanation guided approach. Our results also outperform the state-of-the-art contrastive learning based models. To our knowledge, this work represents the first that combines sequential recommendation with explanation methods. For our future research, we will conduct more extensive analysis of the importance score functions and training efficiency. Explanation guided supervised contrastive learning in particular could be inefficient as it involves selection of retrieved positive views using importance score function. One future research direction is thus to address such overheads by developing appropriate indexing or hashing techniques. One another direction is to meet the challenge of designing augmentations for very long sequences in sequential recommendations. Some users may have very long sequences as they have interacted with many items over a long period. Some useful information or sequence patterns may be hidden in old items in the sequences. Nevertheless, the current neural network-based sequential recommendation models do not directly process very long sequences. Moreover, long sequence modeling is computational cost. It is thus important to address these challenges for long user sequences.

# Chapter 4

# Zero-Shot Next-Item Recommendation using Large Pretrained Language Models

In this chapter, we explore how to leverage the rich world knowledge and strong reasoning abilities of Large Language Models (LLMs) to make next-item recommendations in the zero-shot setting. LLMs have demonstrated impressive performance in various natural language processing tasks when given appropriate input prompts, without requiring fine-tuning on specific training data. However, their application in next-item recommendation remains under-explored. Two major challenges need to be addressed for LLMs to act effectively as recommenders. Firstly, the recommendation space can be unknown to LLMs (e.g., books that can be found at a small brick-and-mortar bookstore) or extremely large to LLMs (e.g., all books that have been published). Secondly, LLMs are inherently unfamiliar with the user preferences required for recommendations. Some strategies are required to ensure that they can derive a good set of candidate items and infer user preferences for making recommendations. In this chapter, we propose a strategy that uses an external module for generating candidate items and a 3-step prompting method for capturing user preferences and making ranked recommendations.

This chapter is structured as follows. We first present the objective in Section 4.1. In Section 4.2, we review the existing Large Language Model-based (LLM) sequential recommendation methods. Next, in Section 4.3, we introduce our proposed Zero-Shot Next-Item Recommendation (NIR) strategy. Finally, we present a comprehensive evaluation of the proposed NIR on two widely used benchmarks in Section 4.4. The NIR competes well with strong sequential recommendation models, offering new and exciting research possibilities for using LLMs in recommender systems.

## 4.1 Objective

Large language models (LLMs) [11, 173, 24], such as GPT-3 [11], have demonstrated impressive results in various natural language processing (NLP) tasks. Nevertheless, the state-of-the-art LLMs are usually very large and often accessible only via some API services. Hence, they cannot be fine-tuned like the earlier pre-trained language models (PTMs) [32, 108]. Many works have also demonstrated that LLMs are capable of solving many known NLP problems through task-specific prompts under the zero-shot setting, i.e., without any examples or further fine-tuning [11, 24]. Nevertheless, using LLMs to perform next-item recommendations is still a relatively new research topic which awaits investigation.

Unlike NLP tasks that rely on the inherent textual knowledge of LLMs, recommendation tasks require LLMs to utilize a user's past item interactions to make item recommendations. Direct methods, such as the Simple Prompting method in Section 4.4, yield poor recommendations [178]. Moreover, LLMs struggle to contribute to recommendations without prior knowledge of the items. In this research, we assume that recommended items should be included in the pre-training data of LLMs (e.g., reviews, Wikipedia pages, etc.). Examples of such items include movies, artists, songs, etc.. For illustration and evaluation, we focus on movie and artist recommendations using GPT-3.5.

In this paper, we introduce an approach for next-item recommendation called Next-Item Recommendation (NIR) prompting. It first limits the recommendation space for a user to a candidate item set using user or item filtering techniques. Secondly, NIR recommends items using a 3-step prompting method: (i) capturing user preferences (Step 1), (ii) selecting representative items from the user's interacted items (Step 2), and (iii) recommending a ranked list of items (Step 3). Finally, we use a formatting technique in Step 3 to ensure easier extraction of recommended items. Our experiments on MovieLens 100K and LastFM 2k with GPT-3.5 (`text-davinci-003`) indicate that NIR prompting is competitive compared to strong supervised learning baselines.

## 4.2 Related LLM-based Recommendation Research

Among the early efforts in pre-trained language model (PTM) based recommendation [73, 178, 120, 28, 40, 152], Zhang et al. [178] proposed to use GPT-2 [108] or BERT [32] as the backbone recommender, making the next-movie prediction based on five previously watched movies by the target user. However, the unknown or huge recommendation space as well as inadequate user preference modeling make the PTMs perform poorly. With newer LLMs such as GPT-3 [11], OPT [173], and PaLM [24] which have shown significantly improved results in various NLP tasks, our work chooses GPT-3 to be the LLM for developing more effective zero/few-shot recommendation methods.

The existing LLM-based recommendation research can be categorized into (a) LLM-augmented recommender systems [38, 154], and (b) LLM-only recommender systems [51, 29, 6, 171]. KAR [154] leverages LLMs for open-world knowledge and improving recommendation accuracy and versatility. Chat-REC is a LLM-based recommender system with conversational chat interface [38]. It utilizes a supervised learning recommender system to select a small set of candidate items and uses LLMs to rerank them for the target user. Chat-REC also provides explanation to the

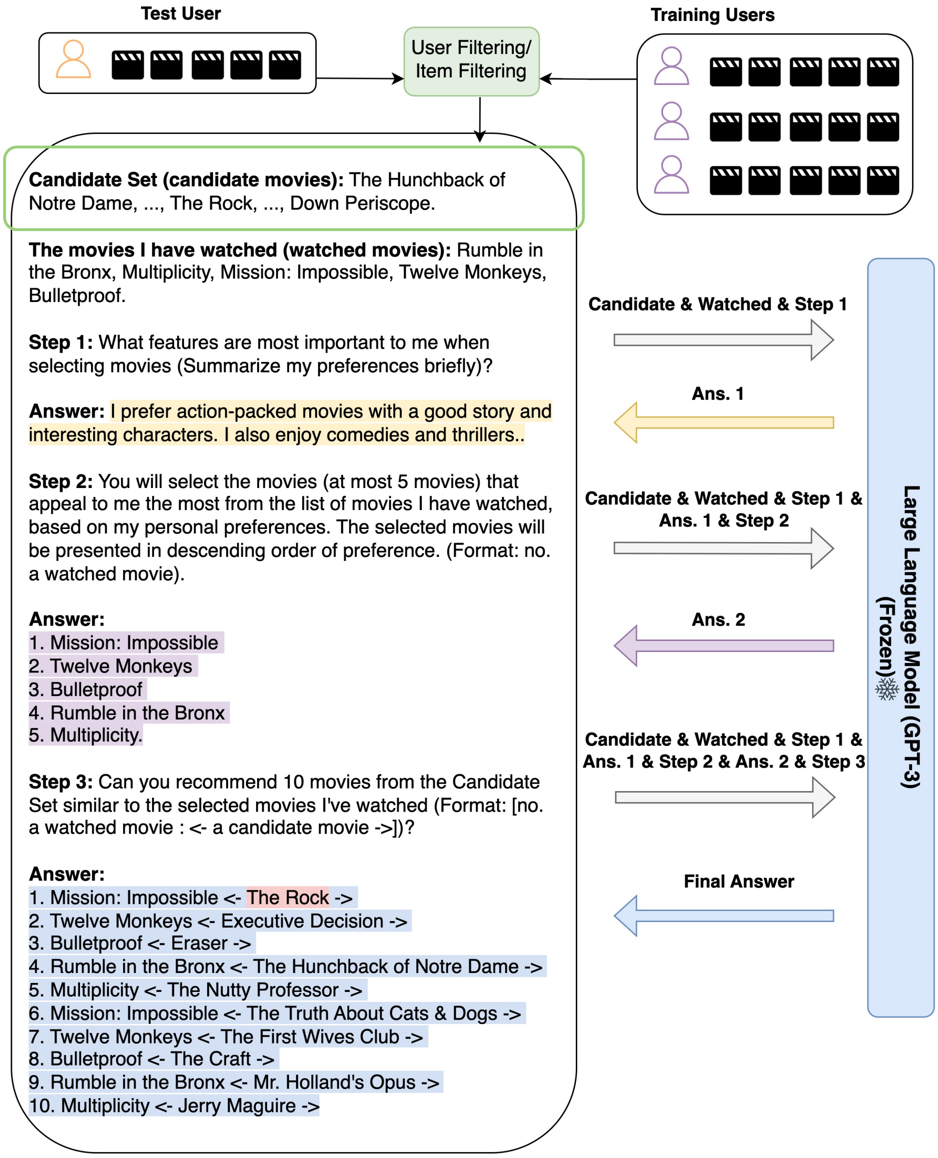


Figure 4.1: Zero-Shot NIR prompts. The ground truth movie in this example is The Rock.

recommended items. Hence, Chat-REC still requires fully supervised learning which could incur significant overhead. For LLM-only recommender systems, Dai et al. [29] conduct an empirical analysis on ChatGPT's recommendation abilities in three ranking policies. Hou et al. [51] explore LLMs (e.g., GPT-4) as ranking models in recommender systems, revealing the promising zero-shot abilities of LLMs. Instead of designing the prompting strategy from scratch, our proposed NIR prompting strategy incorporates user-filtering and item-filtering to derive a candidate item set. This way, it mimics well-known recommendation techniques and leverages its item knowledge and reasoning capability to deliver more accurate recommendation results.

## 4.3 Zero-Shot NIR Prompting Strategy

This section presents our proposed zero-shot NIR prompting strategy. As shown in Figure 4.1, the proposed method has three main components:

**Candidate set construction:** This component performs user-filtering or item-filtering to create a candidate item set for the target user using the training data, which effectively narrows down the recommendation space. These candidate items are then used in the three-step prompting.

**Three-step prompting:** This component involves three instruction prompts corresponding to three subtasks. In the first subtask (*user preference subtask*), we design a user preference prompt (Step 1 prompt) to summarize the target user's preferences based on the previously interacted items. In the second subtask (*representative items subtask*), we then define the Step 2 prompt to combine the user preference prompt and its answer to request the LLM to list representative items based on user preference. In the third subtask (*item recommendation subtask*), we direct the LLM to recommend $k$ items similar to the representative ones.

**Answer extraction:** This component extracts the recommended items from the textual results of the three-step GPT-3.5 prompting using a simple extraction rule.

### 4.3.1 Candidate Set Construction

In Section 4.1, we highlight the challenge of unknown or large recommendation space for LLM-based recommendation. Due to the prompt length limit, and not all items can be fed to the LLM. For instance, 1,683 movies from the MovieLens 100K are too large to be fed into a prompt. Thus, in our approach, we build a candidate item set for the user based on the relevance to the user. Specifically, we employ *user filtering* and *item filtering* to determine these candidate items.

**User-Filtering.** This principle assumes that the candidate items should also be liked by other users similar to the target user. Hence, we first represent every user by a multi-hot vector of their watched items. Users similar to the target user are then derived by cosine similarity between the target user's vector and vectors of other users. Next, we select the $m$ most similar users and the candidate item set of size $n_s$ is constructed by selecting the most popular items among the interacted items by the similar users.

**Item-Filtering.** Similar to user filtering, we represent each item by a multi-hot vector based on its interacted users. Using cosine similarity between two items, we select the $n_m$ most similar items for each item in the target user's interaction history. We then generate a candidate item set of size $n_s$ based on the "popularity" of these similar items among items in the target user's interaction history.

The constructed candidate item set is then incorporated into the prompts for recommendation using the prompt phrase: "Candidate Set (candidate `items`):" as shown in Figure 4.1. Following the candidate set, the prompts also include the list of target user's previously interacted items.

### 4.3.2 Three-Step Prompting

**Step 1: User Preference Prompting.** To capture the user's preferences, we include the sentence "Step 1: What features are most important to me when selecting items (summarize my preferences briefly)?" into the first prompt. As shown in Figure 4.1, the answer returned by LLM summarizes the target user preference (highlighted in pink).

**Step 2: Representative Item Selection Prompting.** As the second step, this prompt includes the previous prompt text appended with the answer of Step 1, followed by the prompt phrase: "Step 2: You will select the items ... that appeal to me the most ... presented in descending order of preference (...)" to determine the previously interacted items that best reflect the target user's preferences. Figure 4.1 shows the

| Method | MovieLens 100K | | LastFM 2K | |
|---|---|---|---|---|
| | HR | NDCG | HR | NDCG |
| POP | 0.0519 | 0.0216 | 0.0755 | 0.0458 |
| FPMC | 0.1018 | 0.0463 | 0.0872 | 0.0449 |
| GRU4Rec | 0.1230 | 0.0559 | 0.0890 | 0.0480 |
| SASRec | **0.1241** | **0.0573** | **0.1101** | **0.0539** |
| Simple Prompting | 0.0297 | 0.0097 | 0.1032 | 0.0410 |
| CS-Random-IF | 0.0805 | 0.0352 | 0.0851 | 0.0440 |
| CS-Random-UF | 0.0954 | 0.0457 | 0.0869 | 0.0378 |
| NIR-Single-IF | 0.0975 | 0.0501 | **0.1198** | **0.0624** |
| NIR-Single-UF | 0.1135 | 0.0529 | 0.1140 | 0.0621 |
| NIR-Multi-IF | 0.1028 | 0.0505 | 0.1013 | 0.0512 |
| NIR-Multi-UF | **0.1187** | **0.0546** | 0.0936 | 0.0492 |

Table 4.1: HR@10 (HR) and NDCG@10 (NDCG) on the test sets of MovieLens 100K and LastFM. (Best results in each group of methods are **boldfaced** and overall best results are underlined.

LLM's answers highlighted in purple.

**Step 3: Recommendation Prompting.** Again, this prompt includes the previous text appended with the answers of Step 2, followed by the prompt phrase "Step 3: Can you recommend 10 items from the Candidate Set similar to ...". This prompt explicitly instructs LLM to generate 10 recommended items from the candidate set as highlighted in blue.

## 4.4 Experiments and Results

### 4.4.1 Experiment Setup

We empirically investigate the performance of the zero-shot NIR strategy against fully trained and zero-shot baselines using the MovieLens 100K dataset [43] (943 users and 1,682 movies) and Last.FM 2k dataset [12] (1,143 users and 11,854 artists) for movie and artist recommendations, respectively. We also experiment on GPT-3.5, the state-of-the-art LLM.

We evaluate our proposed NIR-based methods including: (i) **Zero-Shot NIR-Single-IF/NIR-Single-UF** (that combines the 3 steps into a single prompt leav-

| CSet | UPref | RItem | ML100K | LastFM2K | Average |
|---|---|---|---|---|---|
| – | – | – | 0.0297 | 0.1032 | 0.0664 |
| ✓ | – | – | 0.1019 | 0.1093 | 0.1056 |
| ✓ | ✓ | – | 0.1081 | 0.1112 | 0.1096 |
| ✓ | – | ✓ | 0.1060 | 0.1102 | 0.1081 |
| ✓ | ✓ | ✓ | **0.1135** | **0.1140** | **0.1137** |

Table 4.2: Ablation study of the impact of Candidate Set (CSet), User Preference (UPref), and Representative Items (RItem) in the proposed NIR-Single-UF prompting on MovieLens100K (ML100K) and LastFM datasets. HR@10 is adopted for this evaluation.

ing out the intermediate answers, and prompts GPT-3.5 only once to generate $n$ recommended items from IF/UF-based candidate set.); (ii) **Zero-Shot NIR-Multi-IF/NIR-Multi-UF** (that uses three separate prompts to guide GPT-3.5 step-by-step and incorporates intermediate answers to the subsequent prompts with the IF/UF-based candidate set.). NIR-Single can save some prompting cost compared with NIR-Multi.

The *strong next-item recommendation baselines* to be compared include: (i) **POP** (that recommends most popular items), (ii) **FPMC** [111] (that combines matrix factorization and Markov chains), (iii) **GRU4Rec** [50] (a GRU-based sequential recommendation model), and **SASRec** [58] (a sequential recommendation model based on self-attention). As FPMC and GRU4Rec are fully trained models, they are expected to outperform zero-shot methods. The zero-shot baseline methods to be compared include: (i) **Simple Prompting** (that prompts LLMs to recommend $n$ items directly), (ii) **CS-Random-IF** (that randomly selects $n$ items from the item filtering-based candidate set), and (iii) **CS-Random-UF** (that randomly selects $n$ items from the UF-based candidate set).

We utilize the GPT-3.5 `text-davinci-003` (175B) with public APIs[1], setting the temperature to 0 for consistent results. For $*$**-UF**'s, default values are: most similar users ($m$) as 12, and candidate items ($n_s$) as 19. For $*$**-IF**'s, we use: most similar items ($n_m$) as 10 and candidate items ($n_s$) as 19. We apply a leave-one-out

[1]https://beta.openai.com/docs/models/gpt-3

strategy for performance measurement: the last item in each user sequence is test data, the penultimate is validation, and others form the training set. Evaluation metrics include *Hit Ratio (HR) at 10* and *Normalized Discounted Cumulative Gain (NDCG) at 10*, following the same set of metrics used in [58].

### 4.4.2 Experiment Results

**Main results.** Table 4.1 reveals that our zero-shot NIR-based methods significantly outperform POP. Interestingly, Zero-Shot NIR-Single-UF, NIR-Multi-IF, and NIR-Multi-UF even outperform the fully trained FPMC. Although the three Zero-Shot NIR-based methods perform slightly worse than the strong sequential recommendation model SASRec, they still compete strongly with SASRec. Among zero-shot methods, CS-Random-UF(IF) surpasses Simple Prompting, demonstrating that candidate sets enhance recommendation performance. Our NIR-based prompts outperform Simple Prompting and CS-Random-IF/UF, indicating that combining user preferences and other strategies enrich LLM recommendations. Moreover, Multi-IF(UF) yield better results than Single-IF(UF) on MovieLens 100K, but not LastFM 2K. Simple prompting leads in HR@10 on LastFM but in NDCG@10. UF-based NIR prompts generally perform better than IF-based ones, although IF-based methods deliver better results in the zero-shot setting on LastFM 2K.

**Effects of NIR Prompt Components.** Our proposed methods, NIR-Single-UF/IF and NIR-Multi-UF/IF, involve candidate set construction and a three-step prompting process. We evaluate the effectiveness of these components on MovieLens 100K and LastFM 2K datasets with HR@10. The results in Table 4.2 reveal that each step contributes to better recommendation accuracy. The Simple Prompting method, which employs a candidate set, generally performed better than the one without it (HR@10=0.1056 vs. HR@10=0.0664), highlighting the importance of the candidate set. Our findings show that integrating candidate sets and specific prompting steps further improve performance, suggesting that a narrowed recommendation space and

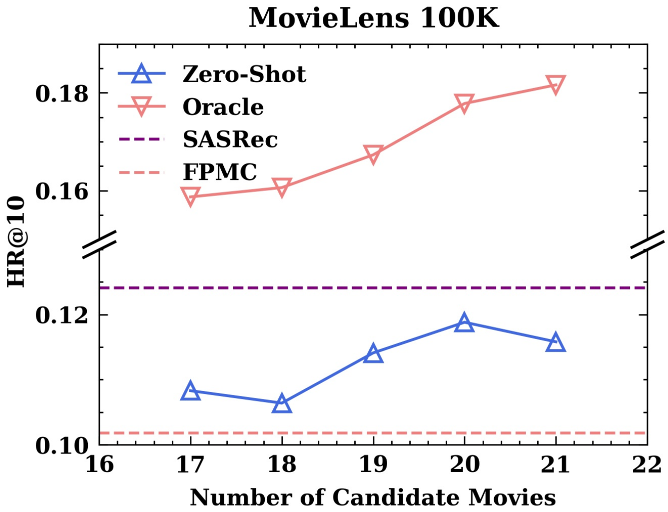


Figure 4.2: HR@10 of Full-Trained SASRec, FPMC and NIR-Single-UF prompting with varying number of candidate movies $n_s$ on MovieLens 100K.

clear guidelines improve GPT-3.5's output.

**Impact of Candidate Set Size $n_s$.** In this study, we examine how the candidate set size affects the performance of NIR-based methods on the MovieLens 100K dataset. We tested the NIR-Single-UF method with candidate set sizes ranging from 17 to 21. The results, depicted in Figure 4.2, show that an optimal candidate set size is around 20; both smaller and larger sizes diminish performance, though it remains between the levels of SASRec and FPMC. Moreover, we observe the oracle's performance continues to improve with larger candidate set ($n_s = 21$). Nevertheless, NIR-Single-UF could not exploit this for performance improvement. Furthermore, while an oracle model, which returns the true item when present in the candidate set, improves its performance with a larger candidate set, NIR-Single-UF does not. This indicates potential for further enhancements in the zero-shot NIR approach. We thus believe there are ample room for the zero-shot NIR approach to further improve.

**Impact of Backbone LLMs.** In this study, we investigate the impact of LLM model size and capability on NIR-based prompting methods for recommendations using various models, such as different versions of GPT-3.5, accessed via OpenAI API on MovieLens 100K. Figure 4.3 ranks these models by capability, from `GPT-3 ada` (lowest) to ChatGPT (highest). Testing on a subset of

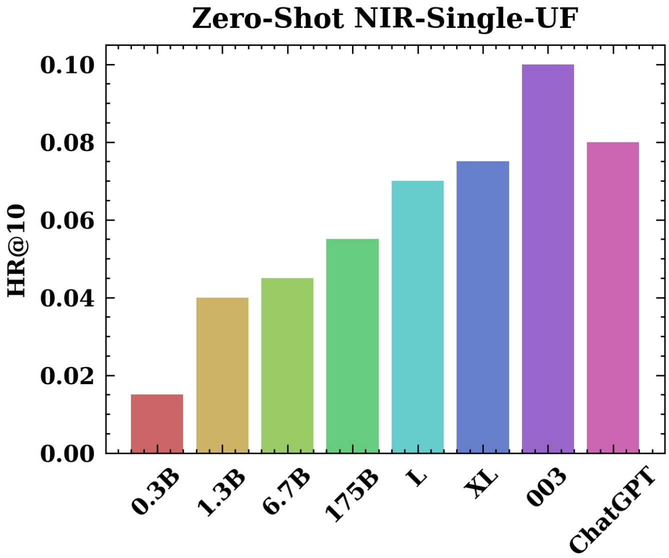


Figure 4.3: HR@10 of NIR-Single-UF prompting using backbone LLMs with different sizes. 0.3B: GPT-3 `ada`, 1.3B: GPT-3 `babbage`, 6.7B: GPT-3 `curie`, 175B: GPT-3 `davinci`, X: Instruct GPT-3 `text-davince-001`, XL: Instruct GPT-3 `text-davinci-002`, 003: GPT-3.5 `text-davinci-003`.

200 examples from the MovieLens 100K dataset shows an improvement in performance from `ada` to `text-davinci-003`. However, ChatGPT underperforms `text-davinci-003`, possibly due to ChatGPT's flexible generation nature. These results indicate that more capable LLMs typically yield better recommendation results.

## 4.5 Summary

In this paper, we propose a three-step prompting strategy called Next-Item Recommendation (NIR) for LLM to make next-item recommendation for user-item interaction sequences. We evaluate our approach using GPT-3.5 as the LLM on both Movielen 100K and LastFM 2K datasets, and obtain promising accuracy. Our results show the potential of using LLMs in zero-shot recommendation and call for further exploration of using LLMs in recommendation tasks. This work can be extended in several directions, including the few-shot approach (instead of zero-shot), choice of LLMs, recommendation of proprietary items, and explainable LLM-based recommendations. Our proposed prompting method partially relies on handcrafted prompts when writing the prompting questions. However, handcrafted prompts are usually based on the personal knowledge and experience of the exports, which can introduce subjective biases.

# Chapter 5

# The Whole is Better than the Sum: Using Aggregated Demonstrations in In-Context Learning for Sequential Recommendation

As LLMs grow in model parameters and training corpus size, they gain emergent abilities by appropriate prompting as covered in Chapter 4. LLMs can further leverage in-context learning (ICL) to perform novel tasks by reasoning based on one or more demonstration example within a given context. In this chapter, we thus focus on developing sequential recommendation models using in-context learning.

This chapter is organized as follows. We first present the objective in Section 5.1, followed by introducing ICL and existing LLM-based sequential recommendation methods (Section 5.2). In Section 5.3, we discuss the important factors to be considered when introducing in-context learning to improve sequential recommendation. We conduct a preliminary empirical study to investigate the role of four aspects of demonstrations. The factors include the wording of prompts, task consistency between demonstrations and test instances, selection of demonstrations, and number of demonstrations. Subsequently, we propose a novel method called LLMSRec-Syn

(Section 5.4). This method incorporates multiple demonstration users into one aggregated demonstration. It is motivated by a finding that increasing the number of demonstrations in ICL does not improve accuracy despite using a long prompt. Next, we present a comprehensive evaluation of the proposed LLMSRec-Syn on three recommendation datasets in Section 5.5.

## 5.1 Objective

**Motivation.** Large language models (LLMs) are known to perform well as a zero-shot solution for many natural language processing tasks [10, 24, 99, 105]. Recently, there are some works that focus on using LLMs to perform recommendation with promising accuracy results [51, 138, 82, 6, 38] and to provide explanations [162, 141]. Most of these works developed LLM prompts for zero-shot sequential recommendation.

To investigate whether LLM can serve as a strong zero-shot sequential recommender, Hou et al. [51] devised a prompt that is filled with historical items in chronological order, candidate items, and instruction to rank the candidate items. In Chapter 4.3, we proposed a three-step prompting method, where LLMs first summarizes the user preference based on the user's past interacted items [138]. It then identifies representative items from the past interacted items that capture the user preference, and finally recommends items among the candidate items which are aligned with the representative items. Among the very few one-shot sequential recommendation works, Liu et al. [82] and Hou et al. [51] explored in-context learning using the test user's second last item as the ground truth next-item and all earlier interacted items as input to create self-demonstrations. Nevertheless, previous experiments have shown that in-context learning (ICL) based sequential recommendation methods perform poorly compared with the supervised learning-based methods (e.g., SASRec) due to the complex recommendation task definition [82, 51, 138]. The illustrative comparison of these two methods is shown in Figure 5.1.

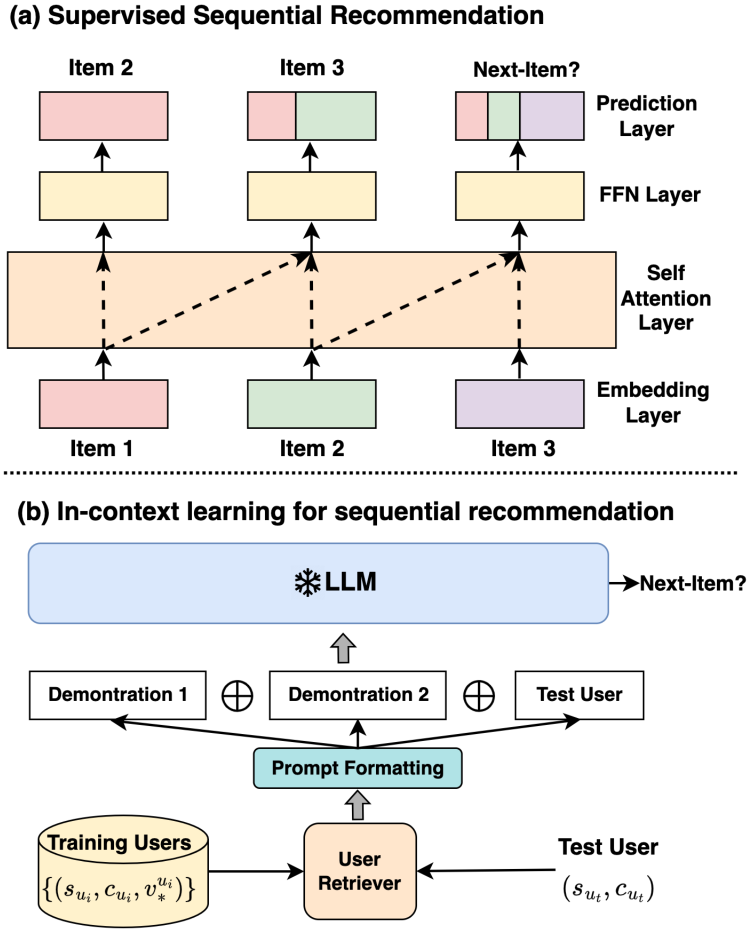


Figure 5.1: The illustrative comparison of (a) supervised sequential recommendation method and (b) in-context learning based sequential recommendation method.

To develop an effective in-context learning approach for LLMs to perform sequential recommendation, we first define the sequential recommendation problem as follows.

**Problem definition for LLM-based sequential recommendation.** We denote each user input instance $u$ as a $(s_u, v_*^u)$ tuple, where $s_u$ represents the sequence of past interacted items (excluding the ground truth next-item $v_*^u$) by $u$. The objective of the sequential recommendation method is to predict $v_*^u$ based on the given $s_u$. In the setting of our LLM-based sequential recommendation method, we denote $c_u$ as the candidate items to be recommended ($|c_u| = M$), and $v_*^u \in c_u$ represents the ground truth next-item included in $c_u$. The LLM-based sequential recommendation in this chapter is required to assign a rank $rank(d) \in [1, M]$ to each item $d$ in $c_{u_t}$ for a test

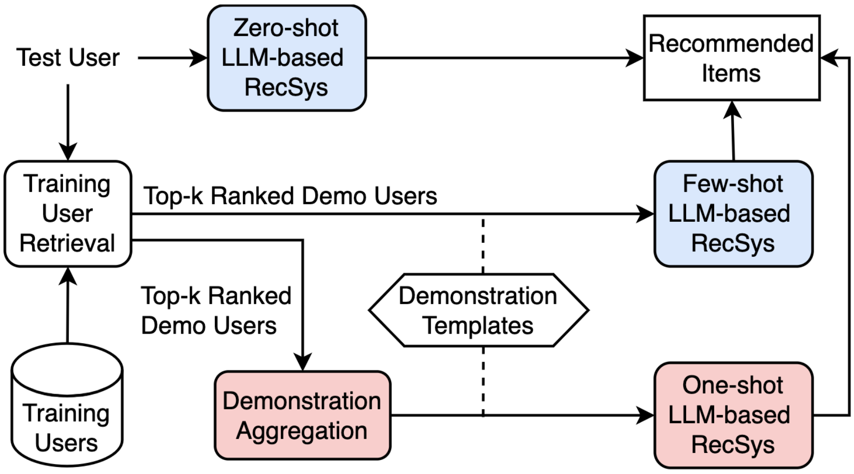


Figure 5.2: The overall framework of zero-shot, few-shot, and aggregated one-shot LLM-based sequential recommender systems.

user $u_t$. Our objective is to ensure that the rank of $v_*^{u_t}$, i.e., $rank(v_*^{u_t})$, can be as high as possible for all users $u_t$'s.

The above problem definition, also adopted in Hou et al. [51], includes $c_u$ as input as it is usually infeasible for LLMs to take all items as input due to limited prompt length. Moreover, if $c_u$ is randomly selected, it would not introduce bias that favors any methods in the evaluation. We also assume a training dataset consisting of user-item interaction sequences from which we can construct demonstrations for ICL. Finally, as LLMs are too large and costly for pretraining or finetuning, we will focus solely on ICL only.

**Overview of our study.** Past works has shown that the effectiveness of ICL in adapting LLMs to new tasks heavily depends on the instruction wording [92, 162], label design [167, 146], selection of demonstrations [81, 119, 179], and number of demonstrations [17, 180]. Our study thus begins by systematically investigating how the instruction format, task consistency (between test and demonstration), demonstration selection, and the number of demonstrations affect ICL-based sequential recommendation. Through our preliminary experiments, we obtain four findings including the one that observes degradation of recommendation accuracy when the number of demonstrations increases. As each demonstration takes up significant length, it is also easy for multiple demonstrations to exceed the prompt limit of

LLMs. Moreover, as LLMs are known to miss out relevant information in a long input prompt [84], we thus embark on a follow-up study on designing a more efficient ICL scheme based on *aggregated demonstration*.

Figure 5.2 shows a comparison of the frameworks for zero-shot, few-shot, and aggregated one-shot LLM-based sequential recommender systems. The key idea in aggregated demonstration is to combine multiple training users into one demonstration. This reduces the repetition of instruction text in the ICL prompt. It also seeks to summarize multiple training users relevant to the test instance in a compact manner. We also develop a novel ICL method using aggregated demonstration for sequential recommendation known as **LLMSRec-Syn**. The length of LLMSRec-Syn prompt increases only gradually with number of demonstration users, LLMSRec-Syn can cope with more relevant information from the demonstration users within a concise input context. We finally show LLMSRec-Syn outperforms other zero-shot and one-shot ICL methods in an extensive set of experiments.

**Contribution.** Our contributions can be summarized as follows: (1) We systematically explore the ICL approach to sequential recommendation by investigating the effect of instruction format, task consistency, demonstration selection, and number of demonstrations to recommendation results; (2) We propose a new in-context learning method for sequential recommendation called LLMSRec-Syn which leverages on a novel concept of aggregated demonstration; (3) We experiment on three popular recommendation datasets and show that LLMSRec-Syn outperforms previous LLM-based sequential recommendation methods.

## 5.2 Works on In-Context Learning

Several works show that LLMs can effectively adapt to different NLP and multimodal tasks, including machine translation [2], visual question answering [164], and foreground segmentation [179]. The adaptation is achieved by learning from a few task-relevant demonstrations, commonly known as in-context learning (ICL) [10].

Table 5.1: Dataset statistics after removing duplicate interactions and users or items with fewer than 5 interactions.

| Datasets | ML-1M | LastFM-2K | Games |
|---|---|---|---|
| # Users | 6,040 | 1,143 | 50,547 |
| # Items | 3,706 | 11,854 | 16,859 |
| # User-item Interactions | 1,000,209 | 68,436 | 389,718 |
| Avg. interacted items per user | 165.59 | 59.92 | 7.71 |
| Avg. interacted users per item | 269.88 | 5.77 | 23.11 |

Despite the above successes, ICL's performance is still significantly affected by the wording of instructions [92, 162], label design [167, 146], demonstration selection [81, 119, 179], and number of demonstrations Chen et al. [17], Zhao et al. [180]. To the best of our knowledge, ICL has not yet been studied in LLM-based sequential recommendation. As sequential recommendation is distinct from the pretraining tasks of LLMs and also different from the above-mentioned tasks, new designs of demonstration(s) and ICL prompt is necessary.

## 5.3 What Makes In-Context Learning Work for Sequential Recommendation

In this section, we conduct a preliminary empirical study to investigate the role of various aspects of demonstrations. These aspects include the wording of prompts, task consistency between demonstrations and test instances, selection of demonstrations, and number of demonstrations. While previous studies have explored the use of LLM as sequential recommenders in a zero-shot manner [51, 138], this is the first study to comprehensively discuss how in-context learning can improve sequential recommendation.

### 5.3.1 Experiment Setup

We implement zero-shot, one-shot, and few-shot methods in this study, using three widely used recommendation datasets: the movie rating dataset *MovieLens-1M* (ML-1M) dataset, the category of *Games* from the Amazon Review dataset [93], and the music artist listening dataset *LastFM-2K* [12]. The data statistics are summarized in Table 5.1. Taking into account cost-effectiveness of LLMs, we select 50 data examples from each of the three datasets to carry out all experiments for analysis in Section 3. Following the previous works [51, 138], we use a leave-one-out strategy for evaluation, i.e., predicting the last interacted item of each user sequence and using the earlier interacted items as input. For each user sequence, we hide the last item, keeping it for testing. The rest of the sequence is used for training and validation. To evaluate the ranking results for each user $u_i$ over a set of candidate items $c_i$, we adopt the widely used **NDCG@N** (**N** = 10, 20) as the evaluation metric. For MovieLens-1M and Games, we directly use the candidate sets utilized in an earlier work [51]. For LastFM, we follow [51] and randomly select candidate items from the item universal set for each user sequence. We then insert the ground truth next item into the candidate item set.

We use ChatGPT (`GPT-3.5-Turbo`) as the default LLM due to its excellent performance and cost-effectiveness. To ensure the reliability of findings, we repeat each experiment 9 times and report the average results. Without exception, we use ML-1M as an example for discussion.

### 5.3.2 In-Context Learning for Sequential Recommendation

In ICL for sequential recommendation, one or a few training users are used as demonstrations that are included in the LLM prompt. Each demonstration includes a training user $u_i$'s historical item interactions $s_{u_i}$, a set of candidates $c_{u_i}$, and ground truth next-item $v_*^{u_i}$. We denote the prompt capturing the demonstration user $u$ by $\mathcal{T}(x_{u_i}, c_{u_i}, y_{u_i})$. The following shows the concatenation of $n$ demonstrations $\mathcal{C}$ which

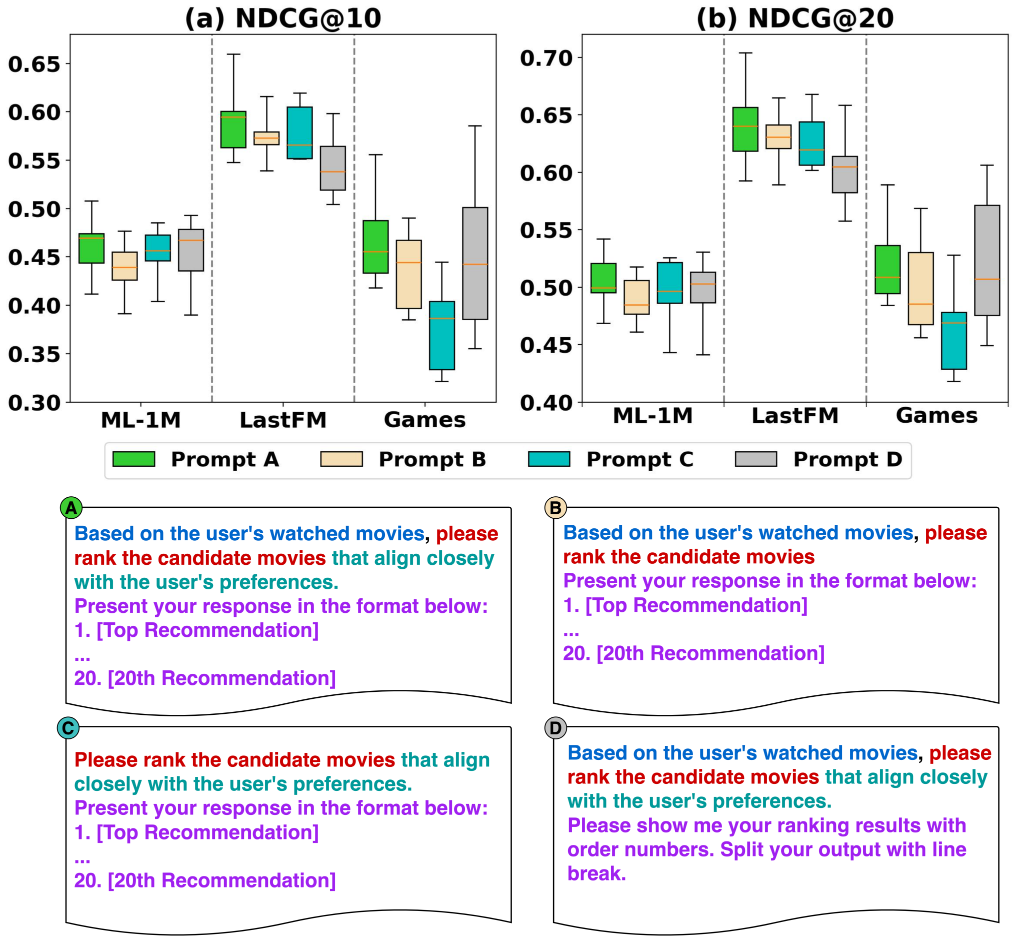


Figure 5.3: Instruction Format options with mention components shown in different colors: (A) Full, (B) w/o preference alignment, (C) w/o watched movie focus, (D) w/o rank result format.

is appended by the instruction prompt for the test user $\mathcal{T}(x_{u_t}, c_{u_t})$ for prediction.

$$\mathcal{C} = \mathcal{T}\left(x_1, c_{u_1}, v_*^{u_1}\right) \oplus \cdots \oplus \mathcal{T}\left(x_{u_n}, c_{u_n}, v_*^{u_n}\right) \tag{5.1}$$

$$v^{u_t} \sim \mathcal{P}_{LLM}\left(\cdot \mid \mathcal{C} \oplus \mathcal{T}\left(x_{u_t}, c_{u_t}, \cdot\right)\right) \tag{5.2}$$

### 5.3.3 Wording of Instructions

LLMs have been found to be sensitive to wording of the prompt [92, 162]. For example, prompts (or instructions) that are semantically similar may yield significantly different results [62, 186, 140, 175]. To examine the impact of instruction wording and exclude the influence of other factors such as demonstration labels and selection, we employ LLM as a zero-shot solver for sequential recommendation.

This concern is also mentioned in works on prompt optimization [162, 31, 104]. However, determining the optimal prompt wording typically requires feedback (such as validation set results [162]), carefully designed reward function [31], or textual feedback from large language models to iteratively update the initial prompt [104]. Instead of a direct application of prompt optimization, this paper focuses on introducing a reasonably well-designed prompt for studying the strategies to determine good demonstrations for sequential recommendation.

We discuss four different options for the instruction format to investigate the sensitivity of the LLM to the wording of the instruction. Considering the prompts used in earlier LLM-based zero-shot recommendation models [51, 138], we derive instructions with four possible mention components (coded with colors consistent with those in Figure 5.3): (a) candidate item ranking, (b) user preference alignment, (c) historical interacted items, and (d) ranked result format. As recommendation is formulated as a ranking task, component (b) is mandatory. The *full* instruction covers all the four components. To explore better instructions, we derive other instruction options by leaving out one of the remaining components. We thus have four instruction options: (A) full instruction $\mathcal{T}^{\mathrm{A}}$, (B) full instruction without (b) $\mathcal{T}^{\mathrm{B}}$, (C) full instruction without (c) $\mathcal{T}^{\mathrm{C}}$, and (D) full instruction with (d) replaced by textual result table description $\mathcal{T}^{\mathrm{D}}$ as shown in Figure 5.3.

As shown in the boxplots at the top of Figure 5.3, we observe that ChatGPT's performance degrades when the instruction does not make reference to interacted items or user preferences across three datasets. This suggests that explicit inclusion of watched movies or user preferences can improve its ability to leverage the user's historical items effectively. While Instruction (A) shows similar average performance as Instruction (D) on ML-1M and LastFM, the former enjoys a smaller variance and outperforms the latter on LastFm. This suggests that LLM prefers explicit output formats over textual description of output format.

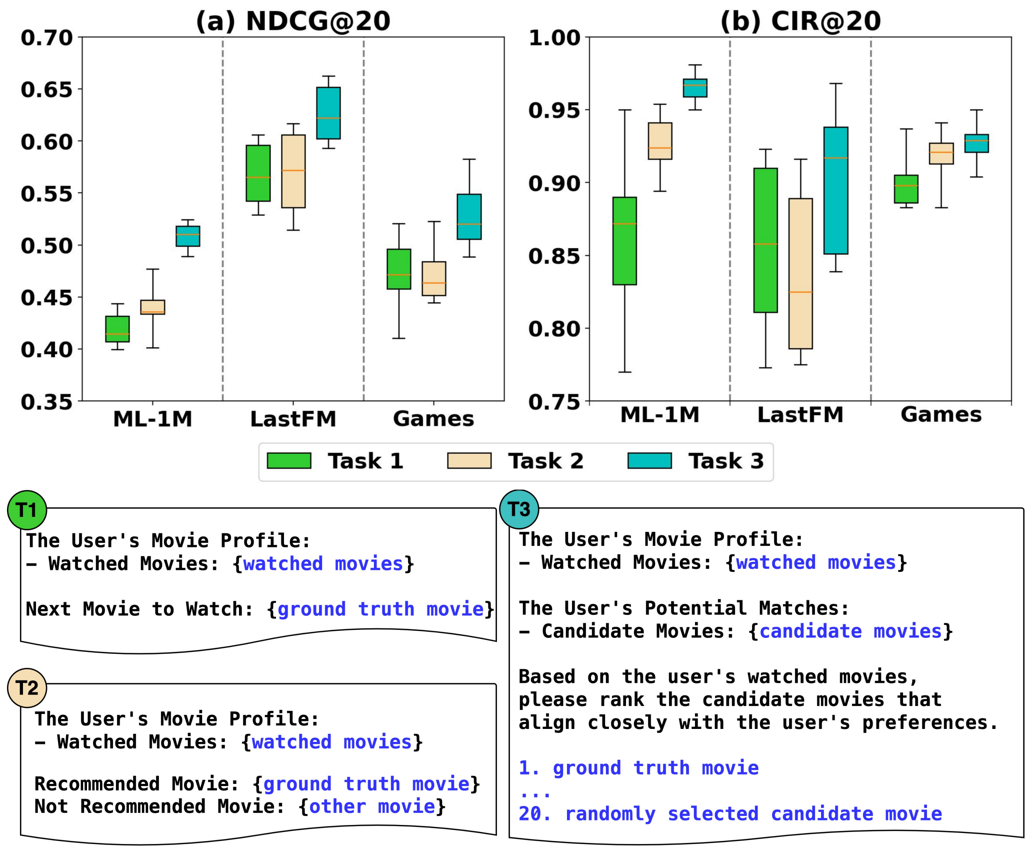


Figure 5.4: Impact of task consistency between demonstrations and test instances. CIR: Candidate Inclusion Ratio of Demonstration Templates: (T1) Next-Item option; (T2) Contrasting Item Pair option; (T3) Ranked Items option.

> ***Finding* 1.** For sequential recommendation, ChatGPT prefers explicit mentions of instructions and explicit mentions of interacted items, user preference alignment and ranked result format.

### 5.3.4 Task Consistency

LLMs are capable of learning new tasks at test time by understanding the relationship between the input of a demonstration and its corresponding output label [167, 146]. In sequential recommendation, LLM is required to rank the ground truth target item at the top followed by other candidate items. However, in a demonstration example from the training set, we observe only one labeled next item but not the ranking of other candidate items. Hence, when constructing demonstrations for in-context learning, we have to answer the important questions: How to prepare the

Table 5.2: Further comparison of different tasks with candidates. We have developed two methods with the same information as T3: (1) T1 w/ Candidate (adding candidate items in the T1 prompt) and (2) T2 w/ Candidate (adding candidate items in the T2 prompt). We evaluate these methods on the ML-1M dataset.

| Methods | T1 | T1 w/ Candidate | T2 | T2 w/ Candidate | T3 |
|---|---|---|---|---|---|
| NDCG@10 | 0.3640 | 0.3766 | 0.3776 | 0.3972 | 0.4584 |
| NDCG@20 | 0.4193 | 0.4420 | 0.4384 | 0.4510 | 0.5077 |

input-label correspondence for a demonstration to be consistent with the sequential recommendation task? To eliminate other factors that may influence the results, such as the number of demonstrations and instructions, we employ instruction (A) as it has proven to be the most effective and robust across three datasets in our previous experiments. We randomly select only one demonstration example for all experiments in this study.

In traditional sequential recommendation, next-item prediction [123, 103], positive and negative item comparison [113, 58, 156], and reranking [161] are commonly utilized objectives to train models. Hence, we develop three different prediction tasks for demonstrations for in-context learning. These tasks include: (T1) predicting the next item, (T2) contrasting item pairs, and (T3) ranking candidate items. The prompts corresponding to these prediction tasks are shown in Figure 5.4. T1 uses the ground truth next-item directly in the demonstration. T2 uses the ground truth next item and another randomly selected item as the positive and negative items respectively. T3 ranks the ground truth next item at the first position and randomly shuffles the remaining candidate items to fill the other positions. Among the task prediction task options, T3 is the only one that aligns closely with the instruction for the test user, i.e., (A).

Figure 5.4 shows the results of these three tasks across three datasets. T3 consistently outperforms T1 and T2 on all three datasets, suggesting that task consistency between demonstration and test user benefits in-context learning for sequential recommendation. Moreover, as the recommended items may not be found among the provided candidates, we also report *candidate inclusion ratio* (CIR) which measures

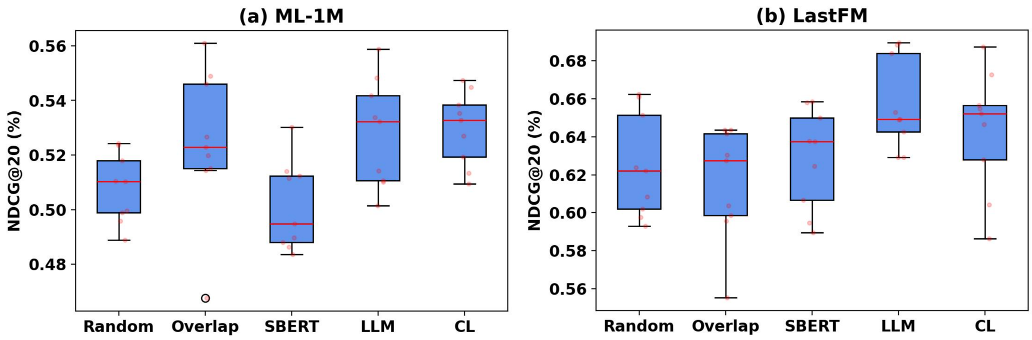


Figure 5.5: Demonstration selection: (1) random selection; (2) overlapping interacted items; (3) cosine similarity between the SBERT embeddings of interacted item sequences; (4) cosine similarity between the LLM embeddings of interacted item sequences; (5) cosine similarity using CL embeddings of interacted item sequences.

the proportion of the candidate items that appear in the ranked item results. As shown in Figure 5.4, we observe that the CIR generally correlates with the NDCG results. The inconsistent demonstration task options (e.g., T1 and T2 coupled with test instruction option (A)) are more likely to cause the LLM to generate non-candidate items in the results. As shown in Table 5.2, We observed that T1 with candidate items (T1 w/ candidate) in the prompt performs better than T1. The same observation applies to T2 w/ candidate and T2. These results show that including more information in the prompt will enhance the performance. However, even with this consideration, T3 still outperforms T1 with candidate items in the prompt and T2 with candidate items in the prompt. T3 thus achieves the overall best performance.

> ***Finding* 2.** Maintaining task consistency between demonstrations and test users is beneficial for in-context learning in sequential recommendation.

### 5.3.5 Selection of Demonstrations

In this work, we evaluate five different demonstration selection methods to determine their impact to in-context learning for sequential recommendation. These methods include: (1) random selection; (2) overlapping historical items of demonstration user and test user; (3) text similarity scores using Sentence-BERT embedding [109]

(SBERT); (4) text similarity scores using OpenAI embedding[1] (LLM); and (5) trained retriever using contrastive learning [156, 75] (CL). Other than random selection, the other four options are also called the retrieval-based methods. In Option (5), positive examples are obtained by data augmentation applied to the anchor user sequence, while negative examples are randomly selected user-item interaction sequences [156]. It has been observed that the performance of in-context learning greatly depends on selecting suitable demonstrations [81]. Utilizing examples that are semantically similar to the test sample can provide more informative and task-relevant knowledge to LLMs. Following Liu et al. [81], there are several follow-up works [117, 119, 179, 75] to develop methods for selecting better demonstrations. In this work, we evaluate five different demonstration selection methods to determine their impact to in-context learning for sequential recommendation. These methods include: (1) random selection; (2) overlapping historical items of demonstration user and test user; (3) text similarity scores using Sentence-BERT embedding [109] (SBERT); (4) text similarity scores using OpenAI embedding[2] (LLM); and (5) trained retriever using contrastive learning [156, 75] (CL). In Option (5), positive examples are obtained by data augmentation applied to the anchor user sequence, while negative examples are randomly selected user-item interaction sequences [156].

Figure 5.5 compares the five selection methods on ML-1M and LastFM as they are used in one-shot sequential recommendation. The results show that selection methods (4) and (5) generally outperform the rest. As method (4) appears to be more robust than (5) and it does not require additional training, we thus use that as the default retriever model in the subsequent experiments.

> ***Finding* 3.** Retrieval-based methods are better than random selection, and stronger LLMs can serve as stronger retrievers without any training.

[1] `text-embedding-ada-002` (https://platform.openai.com/docs/models/moderation)
[2] `text-embedding-ada-002` (https://platform.openai.com/docs/models/moderation)

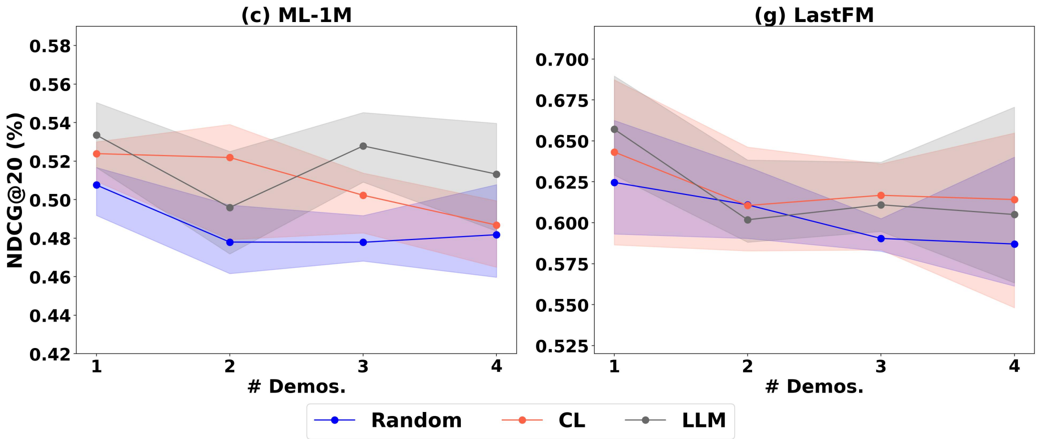


Figure 5.6: Varying number of demonstrations.

### 5.3.6 Number of Demonstrations

When training a model, having more training data examples usually leads to better model performance. However, it is interesting to note that Zhao et al. [180] discover that increasing the number of demonstrations for in-context learning does not necessarily result in improved performance. Chen et al. [17] also show that using only one demonstration may not perform worse than using more demonstrations. In our case, we evaluate the impact of the number of demonstrations on ML-1M and LastFM using random selection, LLM, and CL demonstration selection methods. We conduct experiments by varying the number of demonstrations from 1 to 4, as exceeding 4 demonstrations would exceed the prompt length limit of ChatGPT (`GPT-3.5-Turbo`). Figure 5.6 shows a clear decreasing trend of performance as we increase the number of demonstrations.

> ***Finding* 4.** Increasing demonstrations of in-context learning for sequential recommendation would result in performance degradation and potentially breach the length limit of LLMs.

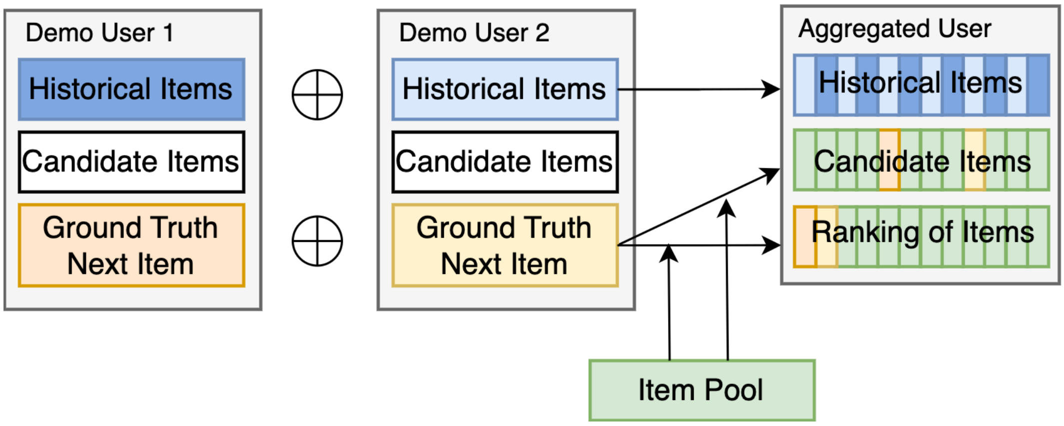


Figure 5.7: Aggregated demonstration construction for sequential recommendation.

## 5.4 In-Context Learning with Aggregated Demonstrations

Finding 4 suggests LLMs have difficulties coping with multiple demonstrations in sequential recommendation. A similar finding by Liu et al. [84] also suggests that the current language models often struggle to utilize information in long input contexts. In particular, their performance tends to significantly degrade when the relevant information is located in the middle of long contexts, also known as the "lost in the middle" phenomenon. The in-context learning prompts for sequential recommendation can easily exceed the prompt length limit of LLM when more than 4 demonstrations are to be accommodated. Such prompts not only suffer from "lost in the Middle", but also incur additional costs of calling LLM APIs.

To address the above challenge, we propose *aggregated demonstration* which combines $K$ ($K > 1$) demonstration users into one for in-context learning. This simple yet effective in-context learning method for sequential recommendation is called **LLMSRec-Syn**. As the prompt length of aggregated demonstration only increases marginally when we increase $K$, LLMSRec-Syn can accommodate more member demonstration users.

Based on Finding 3, LLMSRec-Syn begins with selecting $K$ demonstration users that are similar to the test user. We use similarity between the LLM embeddings of demonstration and test users. We also follow Finding 1 and adopt instruction template (A) for the test user. Based on Finding 2, we also adopt demonstrate template (T3) for the aggregated demonstration to maintain consistency with the task for test user. Next, we construct the aggregated demonstration's historical item-interactions, candidate items, and the desired ranking of the candidate items from its member demonstrations, as shown in Figure 5.7. We will also elaborate these components as follows.

**Historical item-interactions**. Let $H$ denote the historical item-interactions and $H$ is empty initially. We first rank the $K$ selected demonstration users by similarity score. We then add the most recent interacted item from the most similar demonstration to $H$. We repeat the same step for the remaining demonstrations in their similarity order. When we run out of most recent interacted items from $K$ selected demonstrations, we continue to add the next recent interacted items of these demonstrations to $H$ until the number of historical items reaches $MAX_H$.

**Candidate Items**. Let $C$ denote the candidate items of the aggregated demonstration and $C$ is empty initially. We first gather all the ground truth next items from the $K$ selected demonstrations and add them to $C$. Next, we randomly add other items from the item pool to $C$ so as to meet the required number of candidate items.

**Ranking of Candidate Items**. To rank the candidate items in $C$, we place the ground truth next item of the most similar demonstration at rank 1, followed by that of next similar demonstration until we run out of the ground truth next items of all $K$ selected demonstrations. Next, we assign random ranks to the remaining items in $C$.

Once the aggregated demonstration is constructed, it is added to the prompt the same way a training user is added as a demonstration. we add it to the corresponding test user and use them as input for the LLM.

$$\mathcal{C}^{\textbf{agg}} = \mathcal{T}^{\text{A}} \left( \text{Agg}^{\text{T3}} \left( x_{\sigma_1}, c_{\sigma_1}, y_{\sigma_1}, \cdots, x_{\sigma_n}, c_{\sigma_n}, y_{\sigma_n} \right) \right), \tag{5.3}$$

$$y_{\text{test}} \sim \mathcal{P}_{LLM} \left( \cdot \mid \mathcal{C}^{\textbf{agg}} \oplus \mathcal{T}^{\text{A}} \left( x_{\text{test}}, c_{\text{test}} \right) \right), \tag{5.4}$$

where $\sigma_i$ represents the $i^{\text{th}}$ ranked selected users returned by the retrieval model.

Finally, the LLM generates a ranked list of candidate items as the recommendation result.

There are several advantages of the proposed LLMSRec-Syn. They are:

- Unlike the standard demonstration which has one ground truth next-item in the ranked list of recommended items, the aggregated demonstration of LLMSRec-Syn includes more ground truth next-items at topmost positions in the ranked list of recommended items. This approach can avoid sparse signals and provide more guidance to LLMs for recommending to the test user;
- LLMSRec-Syn is less sensitive to the number of demonstrations;
- Cost of LLMSRec-Syn does not increase much with the number of demonstrations; and
- LLMSRec-Syn keeps to the prompt length limit of LLMs.

## 5.5 Experiments and Results

### 5.5.1 Methods for Comparison

To evaluate the performance of LLMSRec-Syn, we conduct an extensive set of experiments on ML-1M, Games, and LastFM-2K datasets. Following the experiment setup of Hou et al. [51], we select 200 data examples from each of the three datasets to carry out all experiments. We use an experiment setup similar to that mentioned in Section 5.3.1 except that we now uses more LLMs and reports the NDCG@N results where N=5,10, and 20. We compare LLMSRec-Syn with 10 methods categorized into 3 types:

**Supervised methods**: Most Popular (Recommending items based on their overall popularity among all users in the training data), GRU4Rec [50] (using GRUs to model user's item sequences), and SASRec [58] (employing a self-attention mechanism to learn user preferences from their item sequences).

**Zero-shot methods**: BM25 [116] (ranking candidate items based on their textual similarity with the test user's interacted items), LLMSeqSim [44] (ranking candidate items by semantically similarity using OpenAI embeddings (`text-embedding-ada-002`)), LLMRank-Seq [51] (using ChatGPT to rank candidate items with crafted prompts), and LLMSRec (a zero-shot version of the proposed LLMSRec-Syn using the instruction prompt $\mathcal{T}^{\mathrm{A}}$).

**One-shot methods**: These include LLMRank-His [51] (using historical items of the test user to form a demonstration), and two variants of LLMSRec-Syn, LLMSRec-Fixed and LLMSRec-Nearest. LLMSRec-Fixed uses a randomly selected demonstration for all test users, while LLMSRec-Nearest finds the most similar training user as the demonstration.

As Section 5.3.6 shows that more than one demonstration in in-context learning for sequential recommendation does not yield better performance, we do not include few-shot methods in this set of experiments. We however will study how many member demonstrations $K$ is ideal for aggregated demonstration (see Section 5.5.2).

We implement LLMSRec-Syn using three different LLMs, LLaMa2 [132], ChatGPT [99] (LLMSRec-Syn), and GPT-4 [100] (LLMSRec-Syn-4). For the LLMSRec-Syn-4 experiment, which is shown in the last row of Table 5.3 to Table 5.5, we use GPT-4 as the base LLM. For all other experiments, including preliminary studies, in-depth analysis, and method comparisons presented in Table 5.3 to Table 5.5, we use the same ChatGPT (`GPT-3.5-Turbo`). To ensure the reliability of our findings, each experiment is conducted 9 times, and the average results are reported.

However, we found LLaMa2 unable to follow recommendation instructions and is prone to generating historical interacted items or in-context examples. Hence, we exclude the LLaMa2 results from the evaluation results. In LLMSRec-Syn, we set the number of member users in the aggregated demonstration as {1,2,3,4,5,6,7} and conduct a brute force search to determine the optimal number for each dataset. We set the number of historical items $MAX_H = 50$ and number of candidate items to 20. Other than reporting the NDCG@N results, we also analyse specific test cases of

LLMSRec-Syn-4 in Section 5.5.5.

### 5.5.2 Main Results

Table 5.3: Main results. We report NDCG@5, NDCG@10 and NDCG@20 on the ML-1M dataset. (Best results in each group of methods are **boldfaced** and overall best results are <u>underlined</u>).

| Setting | Method | NDCG@5 | NDCG@10 | NDCG@20 |
|---|---|---|---|---|
| Supervised | Most Popular | 0.3673 | 0.4623 | 0.4748 |
| | GRU4Rec | 0.7205 | 0.7494 | 0.7610 |
| | SASRec | <u>**0.7322**</u> | <u>**0.7595**</u> | <u>**0.7702**</u> |
| Zero-shot | BM25 | 0.1314 | 0.2053 | 0.3370 |
| | LLMSeqSim | 0.3250 | 0.4037 | **0.4723** |
| | LLMRank-Seq | **0.3344** | 0.3882 | 0.4612 |
| | LLMSRec | 0.3339 | **0.4087** | **0.4723** |
| One-shot | LLMRank-His | 0.3919 | 0.4444 | 0.5074 |
| | LLMSRec-Fixed | 0.3590 | 0.4193 | 0.4793 |
| | LLMSRec-Nearest | 0.3842 | 0.4382 | 0.5017 |
| | LLMSRec-Syn | 0.4267 | 0.4813 | 0.5334 |
| | LLMSRec-Syn-4 | **0.5112** | **0.5685** | **0.5936** |

Table 5.4: Main results. We report NDCG@5, NDCG@10 and NDCG@20 on the LastFM-2K dataset. (Best results in each group of methods are **boldfaced** and overall best results are <u>underlined</u>).

| Setting | Method | NDCG@5 | NDCG@10 | NDCG@20 |
|---|---|---|---|---|
| Supervised | Most Popular | 0.4055 | 0.4205 | 0.4803 |
| | GRU4Rec | 0.3382 | 0.3971 | 0.4784 |
| | SASRec | **0.4081** | **0.4680** | **0.5303** |
| Zero-shot | BM25 | 0.1215 | 0.1393 | 0.3354 |
| | LLMSeqSim | 0.4090 | 0.4662 | 0.5293 |
| | LLMRank-Seq | 0.5084 | 0.5545 | **0.6070** |
| | LLMSRec | **0.5126** | **0.5602** | 0.6057 |
| One-shot | LLMRank-His | 0.5318 | 0.5725 | 0.6212 |
| | LLMSRec-Fixed | 0.4961 | 0.5425 | 0.5984 |
| | LLMSRec-Nearest | 0.5249 | 0.5697 | 0.6197 |
| | LLMSRec-Syn | 0.5554 | 0.5918 | 0.6371 |
| | LLMSRec-Syn-4 | <u>**0.6544**</u> | <u>**0.6799**</u> | <u>**0.7017**</u> |

The main experiment results are shown in Table 5.3 to Table 5.5. From the results table, we obtain the following findings:

Table 5.5: Main results. We report NDCG@5, NDCG@10 and NDCG@20 on the Games dataset. (Best results in each group of methods are **boldfaced** and overall best results are <u>underlined</u>).

| Setting | Method | NDCG@5 | NDCG@10 | NDCG@20 |
|---|---|---|---|---|
| Supervised | Most Popular | 0.2746 | 0.3905 | 0.4496 |
| | GRU4Rec | 0.6747 | 0.7002 | 0.7278 |
| | SASRec | <u>**0.6828**</u> | <u>**0.7189**</u> | <u>**0.7311**</u> |
| Zero-shot | BM25 | 0.2285 | 0.3108 | 0.4055 |
| | LLMSeqSim | **0.4269** | **0.4830** | **0.5360** |
| | LLMRank-Seq | 0.3063 | 0.3607 | 0.4074 |
| | LLMSRec | 0.4070 | 0.4555 | 0.5103 |
| One-shot | LLMRank-His | 0.4191 | 0.4667 | 0.5206 |
| | LLMSRec-Fixed | 0.3744 | 0.4400 | 0.4899 |
| | LLMSRec-Nearest | 0.3975 | 0.4388 | 0.4994 |
| | LLMSRec-Syn | 0.4989 | 0.5334 | 0.5869 |
| | LLMSRec-Syn-4 | **0.5647** | **0.6019** | **0.6277** |

**ICL one-shot methods with appropriate demonstrations out-perform zero-shot methods**. As shown in Table 5.3 to Table 5.5, the one-shot LLMSRec-Syn out-perform the NDCG@5, NDCG@10, and NDCG@20 of the zero-shot LLMSRec by 19.5%, 13.5%, and 11.0% respectively, when averaged across three benchmark datasets. It is significantly better than the strong zero-shot baseline such as LLMRank-Seq by 33.2% , 26.2%, and 21.6% respectively, averaged across all benchmark datasets. Furthermore, LLMRank-His, LLMSRec-Fixed, and LLMSRec-Nearest using one training user as demonstration outperform LLMRank-Seq on three datasets, except for LLMSRec-Fixed which performs slightly worse than LLMRank-Seq on LastFM-2K. This result suggests that ICL can enhance the LLM's ability to perform a complex task such as sequential recommendation.

**Aggregated demonstration, combining multiple member users, allows LLM to effectively gather useful task specific information about the test user within a concise context**. Compared to other ICL baselines (i.e., LLMRank-His, LLMSRec-Fixed, and LLMSRec-Nearest), LLMSRec-Syn achieves the superior one-shot performance across all datasets as shown in Table 5.3 to Table 5.5. For example, the one-shot LLMSRec-Syn outperforms the NDCG@5, NDCG@10, and NDCG@20 scores of the strongest baseline, LLMRank-His, by 10.8%, 8.6%, and 6.8% respectively, when averaged across three benchmark datasets. While Figure 5.6 shows that having more demonstrations may hurt ICL for sequential recommendation, the idea of incorporating multiple demonstration users into an aggregated demonstration enhances the performance of LLMSRec-Syn. These results illustrate the advantage of aggregated demonstration in accommodating multiple training users within a limited prompt length.

**LLMSRec-Syn is competitive against supervised methods when the amount of training data is limited**. LLMSRec-Syn easily outperforms the simple supervised baseline, Most Popular. While it does not outperform GRU4Rec and SASRec on ML-1M and Games, LLMSRec-Syn surprisingly outperforms all supervised baselines on LastFM-2K. One possible reason is that LastFM-2K has sparse information about items after removing duplicate user-item interactions and users/items with less than 5 interactions, making it challenging to train a good supervised model.

**LLMSRec-Syn using more powerful LLMs may outperform supervised methods in the future**. With rapid advancement of LLM research, LLMSRec-Syn can be further enhanced when more powerful LLM is used. Our results in Table 5.3 to Table 5.5 shows that LLMSRec-Syn-4 significantly outperforms the NDCG@5, NDCG@10, and NDCG@20 scores of LLMSRec-Syn on all the 3 datasets by 16.9%, 15.2%, 9.4% respectively.

### 5.5.3 Analysis of Aggregated Demonstrations

In this section, we study the recommendation performance when varying the settings of aggregated demonstrations. Analysis of ordering of users and label in the aggregated demonstration can be found in the Section 5.5.4.

**Impact of number of users in the aggregated demonstration**. We evaluate the impact of $K$ (the number of member users) in the aggregated demonstration on LLMSRec-Syn's performance. We empirically vary $K$ from 2 to 7. As shown

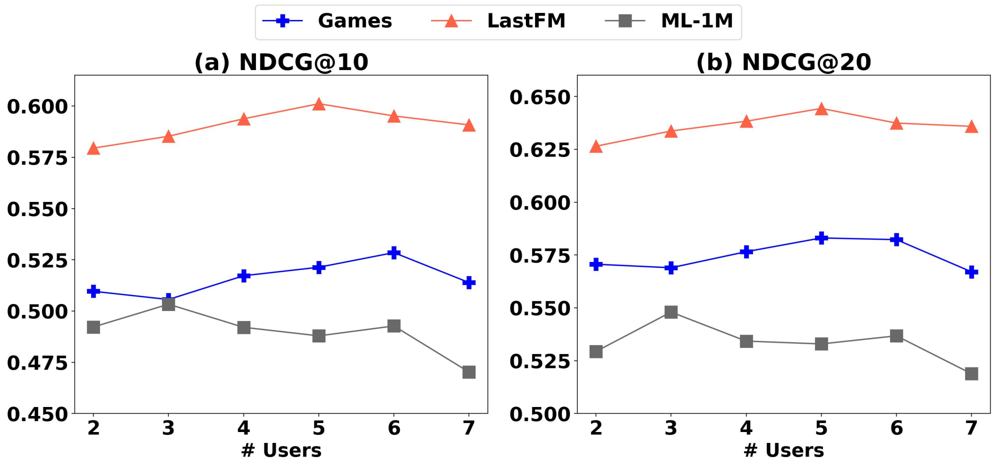


Figure 5.8: Varying number of users ($K$) in aggregated demonstration.

in Figure 5.8, an approximate inverted U-shaped relationship exists between $K$ and NDCG@10/20 performance. Initially, as $K$ increases, there is a noticeable performance increase, suggesting that LLMSRec-Syn benefits from aggregated demonstration. However, beyond some $K$ value, more member users in aggregated demonstration leads to lower performance. This can be explained by more irrelevant training users being incorporated into the aggregated demonstration.

**Impact of number of aggregated demonstrations.** We evaluate the impact of the number of aggregated demonstrations to LLMSRec-Syn by varying the number of aggregated demonstrations from 1 to 4 such that each demonstration involves 2 users (see Figure 5.9(a)) or 3 users (see Figure 5.9(b)). For the Games dataset, experimentation with 3 aggregated demonstrations was not possible due to GPT-3.5-Turbo's input limit. The results show that a single aggregated demonstration outperforms multiple ones, except in the LastFM-2K dataset, where two aggregated demonstrations slightly excel.

### 5.5.4 More In-Depth Analysis

**Impact of user order in aggregated demonstrations**. We experiment with 3 possible orders of member users: (i) Random (randomly selects historical items and next-items from the selected users to construct the aggregated demonstration), (ii)

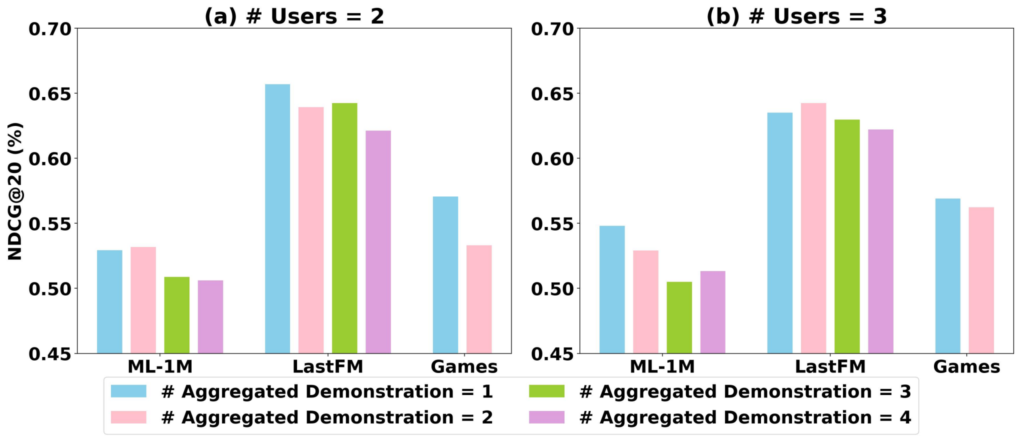


Figure 5.9: Varying number of aggregated demonstrations each with: (a) 2 member users, and (b) 3 member users.

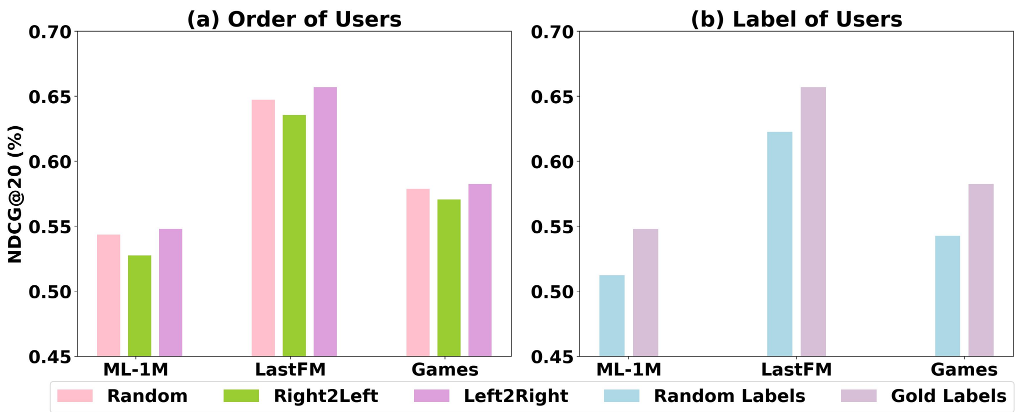


Figure 5.10: (a) Ordering of member users in the aggregated demonstration. (b) Ground truth vs random next-items in aggregated demonstrations.

Right2Left (the reverse order of demonstration users in constructing an aggregated demonstration in LLMSRec-Syn in Section 5.4), and (iii) Left2Right (the user order used in the LLMSRec-Syn). Figure 5.10(a) illustrates that Left2Right and Right2Left are the most and least ideal orders respectively. The performance of Random is naturally sandwiched in between.

**Impact of labeled next-items in the aggregated demonstration**. According to Min et al. [94], ground truth labels are not important for in-context learning. To investigate this claim for ICL-based sequential recommendation, we compare LLMSRec-Syn using ground truth next-items in the aggregated demonstration (referred to as "Gold Labels") with that using randomly chosen non-ground truth next-items (referred to as "Random Labels"). Our results in Figure 5.10(b) clearly indicate that ground

Table 5.6: Results of fine-tuned LLaMa2 with LoRA for in-context sequential recommendation. Regular means LLaMa2-LoRA-Regular. Aggregated means LLaMa2-LoRA-Aggregated.

| ML-1M | Regular | Aggregated | LLMSRec-Syn |
|---|---|---|---|
| NDCG@10 | 0.3640 | 0.3766 | 0.3776 |
| NDCG@20 | 0.4193 | 0.4420 | 0.4384 |

truth next-items are required to yield better performance contradicting the claim by Min et al. [94]. This could possibly be explained by the complexity of sequential recommendation task.

**Could fine-tuned LLaMa2 improve the performance of in-context sequential recommendation?** We initially used a training dataset of 150 data examples to train LLaMa2 with LoRA, which we referred to as LLaMa2-LoRA-Regular. For each training data example in this training dataset, the target output is the ranking of the candidate items for a training user. The input consists of a regular demonstration example, as well as historical items and candidate items from the training user. After training, we evaluated the performance of LLaMa2-LoRA-Regular using the same 50 test users as ChatGPT-based LLMSRec-Syn (0.5283 NDCG@10).

As shown in Table 5.6, the results showed that LLaMa2-LoRA-Regular achieved a NDCG@10 score of 0.2344. To investigate whether aggregated demonstration helps to train a better model compared to regular demonstrations, we prepared a training dataset using aggregated demonstrations instead of regular demonstrations. We trained LLaMa2 with LoRA using this dataset, which we call LLaMa2-LoRA-Aggregated. LLaMa2-LoRA-Aggregated achieved a NDCG@10 score of 0.3432 on the same test set. Although the initial study indicates that LLaMa-LoRA performs worse than ChatGPT, the fine-tuned LLaMa2-LoRA appears to have the potential to enable in-context learning-based sequential recommendation and aggregated demonstration can help to train a better model.

### 5.5.5 Case Study Examples

In this section, we provide comparative examples of one-shot LLMSRec-Syn (Table 5.7), one-shot LLMSRec-Nearest (Table 5.8), one-shot LLMSRec-Fixed (Table 5.9), and zero-shot LLMSRec (Table 5.10). Observations show that LLMSRec-Syn ranks the ground truth movie higher than the other methods. Compared to Nearest and Fixed demonstrations, the aggregated demonstration allows the LLM to better identify a user's interests and align the ranking with those interests. Without demonstration, zero-shot LLMSRec relies solely on the LLM's knowledge and performs poorly. This suggests that LLMs can learn from demonstrations to improve in areas where they might not originally be good at.

Table 5.7: Example of the one-shot LLMSRec-Syn on the ML-1M dataset. Ground truth recommendation is highlighted in Maroon.

```
Aggregated Demonstration Example:

The User's Movie Profile:
- Watched Movies: ['0. Caddyshack', '1. Glory', '2. A
Bug's Life', '3. Star Trek VI: The Undiscovered Country', '4.
Indiana Jones and the Last Crusade', '5. The Color of Money',
'6. Raging Bull', '7. Edward Scissorhands', '8. Kramer Vs.
Kramer', '9. Roger & Me', '10. Romancing the Stone', '11.
Full Metal Jacket', '12. The Shining', '13. Easy Rider',
'14. Glory', '15. The Color Purple', '16. Die Hard', '17.
Who Framed Roger Rabbit?', '18. Ghostbusters', '19. The
Right Stuff', '20. No Way Out', '21. The Breakfast Club',
'22. Dead Poets Society', '23. One True Thing', '24. Full
Metal Jacket', '25. U2: Rattle and Hum', '26. Caddyshack',
'27. Jaws', '28. Get Shorty', '29. A Fish Called Wanda',
'30. Star Trek IV: The Voyage Home', '31. Brazil', '32. The
Fugitive', '33. Willy Wonka and the Chocolate Factory', '34.
Ghostbusters', '35. The Blues Brothers', '36. Stripes', '37.
Stand by Me', '38. Pleasantville', '39. Bull Durham', '40.
Risky Business', '41. Indiana Jones and the Last Crusade', '42.
Star Wars: Episode VI - Return of the Jedi', '43. Witness',
'44. Brazil', '45. Predator', '46. The Terminator', '47.
Grosse Pointe Blank', '48. Contact', '49. The Princess
Bride']
The User's Potential Matches:
- Candidate Movies: ['0. Excalibur', '1. The Fox and the
Hound', '2. The Cable Guy', '3. Cat on a Hot Tin Roof', '4.
Slingshot, The (Kådisbellan )', '5. Institute Benjamenta,
or This Dream People Call Human Life', '6. Prefontaine', '7.
Star Wars: Episode IV - A New Hope', '8. 8 Seconds', '9. Two
Family House', '10. American Graffiti', '11. Seven Days in
May', '12. The English Patient', '13. Jurassic Park', '14.
Meet the Parents', '15. Weekend at Bernie"s', '16. Kurt &
Courtney', '17. Children of a Lesser God', '18. The Haunting',
'19. Love and Basketball']
Based on the user's watched movies, please rank the candidate
movies that align closely with the user's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.
Present your response in the format below:
1. [Top Recommendation (Candidate Movie)]
2. [2nd Recommendation (Candidate Movie)]
...
20. [20th Recommendation (Candidate Movie)]
Answer:
1. Children of a Lesser God
2. Excalibur
3. Jurassic Park
4. American Graffiti
5. Meet the Parents
6. The Haunting
7. Institute Benjamenta, or This Dream People Call Human Life
8. Cat on a Hot Tin Roof
```

9. Slingshot, The (Kådisbellan )
10. Prefontaine
11. The English Patient
12. Love and Basketball
13. The Fox and the Hound
14. The Cable Guy
15. Two Family House
16. Star Wars: Episode IV – A New Hope
17. Weekend at Bernie"s
18. Seven Days in May
19. 8 Seconds
20. Kurt & Courtney

---

**Test User:**

Learn from the above demonstration examples to solve the following test example.

**The User's Movie Profile:**

- Watched Movies: ['0. Nikita (La Femme Nikita)', '1. Wallace & Gromit: The Best of Aardman Animation', '2. Unforgiven', '3. Batman: Mask of the Phantasm', '4. The Fugitive', '5. Out of Sight', '6. Twelve Monkeys', '7. GoodFellas', '8. Fight Club', '9. Groundhog Day', '10. A Bug"s Life', '11. Tombstone', '12. Aladdin', '13. Beauty and the Beast', '14. Grosse Pointe Blank', '15. Election', '16. Leaving Las Vegas', '17. Total Recall', '18. A Few Good Men', '19. Pleasantville', '20. Jerry Maguire', '21. Pretty Woman', '22. Contact', '23. True Lies', '24. Waking Ned Devine', '25. Romeo Must Die', '26. Mission: Impossible 2', '27. Mission to Mars', '28. Killer, The (Die xue shuang xiong)', '29. Blade Runner', '30. The Princess Bride', '31. Brazil', '32. Henry V', '33. Amadeus', '34. The Right Stuff', '35. The Terminator', '36. Stand by Me', '37. Back to the Future', '38. This Is Spinal Tap', '39. Gandhi', '40. Star Trek: The Wrath of Khan', '41. Ghostbusters', '42. Mad Max 2 (a.k.a. The Road Warrior)', '43. A Fish Called Wanda', '44. Trading Places', '45. Chariots of Fire', '46. Time Bandits', '47. Who Framed Roger Rabbit?', '48. Indiana Jones and the Last Crusade', '49. Die Hard']

**The User's Potential Matches:**

- Candidate Movies: ['0. Sorority House Massacre', '1. Arguing the World', '2. Cop Land', '3. The Inkwell', '4. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)', '5. Airplane!', '6. Pokémon the Movie 2000', '7. Johnny 100 Pesos', '8. Happy, Texas', '9. Arlington Road', '10. The Prophecy II', '11. Prom Night', '12. The Commitments', '13. Federal Hill', '14. Another Stakeout', '15. The Postman Always Rings Twice', '16. The Crow', '17. Shanghai Surprise', '18. Patriot Games', '19. Modulations']

Based on the user's watched movies, please rank the candidate movies that align closely with John's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.
Present your response in the format below: 1. [Top Recommendation (Candidate Movie)] 2. [2nd Recommendation (Candidate Movie)] ... 20. [20th Recommendation (Candidate Movie)]

**Answer:**

1. The Commitments
2. Cop Land
3. **Airplane!**
4. The Crow
5. Patriot Games
6. The Postman Always Rings Twice
7. Shanghai Surprise
8. Arlington Road
9. Happy, Texas
10. Another Stakeout
11. The Inkwell
12. Modulations
13. Johnny 100 Pesos
14. Sorority House Massacre
15. Pokémon the Movie 2000
16. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)
17. The Prophecy II
18. Federal Hill
19. Prom Night
20. Arguing the World

Table 5.8: Example of the one-shot LLMSRec-Nearest on the ML-1M dataset.

```
Nearest Demonstration Example:

The User's Movie Profile:
- Watched Movies: ['E.T. the Extra-Terrestrial', 'Gladiator',
'Raiders of the Lost Ark', 'Brazil', 'Aliens', 'Full Metal
Jacket', 'The Right Stuff', 'The Terminator', 'Down by Law',
'Blade Runner', 'The Princess Bride', 'Mystery Train', 'Stand
by Me', 'Dangerous Liaisons', 'Year of Living Dangerously',
'Poltergeist', 'Crimes and Misdemeanors', 'Never Cry Wolf', 'Mad
Max 2 (a.k.a.  The Road Warrior)', 'Women on the Verge of a
Nervous Breakdown', "Ferris Bueller's Day Off", 'Who Framed
Roger Rabbit?', 'Koyaanisqatsi', 'Ghostbusters', 'A Fish Called
Wanda', 'The Gods Must Be Crazy', 'Chariots of Fire', 'The
Hidden', 'Diva', 'Gandhi', 'F/X', 'Sex, Lies, and Videotape',
'Reds', 'Back to the Future', 'Driving Miss Daisy', 'Body Heat',
'Indiana Jones and the Last Crusade', 'When Harry Met Sally...',
'Batman', 'The Untouchables', 'Predator', 'Risky Business', 'The
Blues Brothers', 'Star Trek IV: The Voyage Home', 'U2:  Rattle
and Hum', 'No Way Out', 'The Color Purple', 'Romancing the
Stone', 'The Color of Money', 'Caddyshack']
The User's Potential Matches:
- Candidate Movies:  ['Children of a Lesser God', 'Kicking and
Screaming', 'Harlem', 'The Good Mother', 'Oscar and Lucinda
(a.k.a.  Oscar & Lucinda)', 'Meet Me in St.  Louis', 'Sinbad
and the Eye of the Tiger', 'Gremlins', "Freddy's Dead:  The
Final Nightmare", 'One Magic Christmas', 'The Scarlet Letter',
'Alaska', 'Red Firecracker, Green Firecracker', 'Bulletproof',
'The Crimson Pirate', 'Clockers', 'The Stepford Wives', 'Twelve
Monkeys', 'Bitter Moon', 'Friday the 13th:  The Final Chapter']
Based on the user's watched movies, please rank the candidate
movies that align closely with the user's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.
Present your response in the format below:
1.  [Top Recommendation (Candidate Movie)]
2.  [2nd Recommendation (Candidate Movie)]
...
20.  [20th Recommendation (Candidate Movie)]
Answer:
1.  Children of a Lesser God
2.  One Magic Christmas
3.  Alaska
4.  Gremlins
5.  The Good Mother
6.  Bitter Moon
7.  Bulletproof
8.  Freddy's Dead:  The Final Nightmare
9.  Red Firecracker, Green Firecracker
10.  The Scarlet Letter
11.  Sinbad and the Eye of the Tiger
12.  Kicking and Screaming
13.  Meet Me in St.  Louis
14.  Oscar and Lucinda (a.k.a.  Oscar & Lucinda)
15.  Twelve Monkeys
```

16. Clockers
17. Harlem
18. The Stepford Wives
19. The Crimson Pirate
20. Friday the 13th: The Final Chapter

---

**Test User:**

Learn from the above demonstration examples to solve the following test example.
**The User's Movie Profile:**
- Watched Movies: ['0. Nikita (La Femme Nikita)', '1. Wallace & Gromit: The Best of Aardman Animation', '2. Unforgiven', '3. Batman: Mask of the Phantasm', '4. The Fugitive', '5. Out of Sight', '6. Twelve Monkeys', '7. GoodFellas', '8. Fight Club', '9. Groundhog Day', '10. A Bug"s Life', '11. Tombstone', '12. Aladdin', '13. Beauty and the Beast', '14. Grosse Pointe Blank', '15. Election', '16. Leaving Las Vegas', '17. Total Recall', '18. A Few Good Men', '19. Pleasantville', '20. Jerry Maguire', '21. Pretty Woman', '22. Contact', '23. True Lies', '24. Waking Ned Devine', '25. Romeo Must Die', '26. Mission: Impossible 2', '27. Mission to Mars', '28. Killer, The (Die xue shuang xiong)', '29. Blade Runner', '30. The Princess Bride', '31. Brazil', '32. Henry V', '33. Amadeus', '34. The Right Stuff', '35. The Terminator', '36. Stand by Me', '37. Back to the Future', '38. This Is Spinal Tap', '39. Gandhi', '40. Star Trek: The Wrath of Khan', '41. Ghostbusters', '42. Mad Max 2 (a.k.a. The Road Warrior)', '43. A Fish Called Wanda', '44. Trading Places', '45. Chariots of Fire', '46. Time Bandits', '47. Who Framed Roger Rabbit?', '48. Indiana Jones and the Last Crusade', '49. Die Hard']
**The User's Potential Matches:**
- Candidate Movies: ['0. Sorority House Massacre', '1. Arguing the World', '2. Cop Land', '3. The Inkwell', '4. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)', '5. Airplane!', '6. Pokémon the Movie 2000', '7. Johnny 100 Pesos', '8. Happy, Texas', '9. Arlington Road', '10. The Prophecy II', '11. Prom Night', '12. The Commitments', '13. Federal Hill', '14. Another Stakeout', '15. The Postman Always Rings Twice', '16. The Crow', '17. Shanghai Surprise', '18. Patriot Games', '19. Modulations']
Based on the user's watched movies, please rank the candidate movies that align closely with John's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.
Present your response in the format below: 1. [Top Recommendation (Candidate Movie)] 2. [2nd Recommendation (Candidate Movie)] ... 20. [20th Recommendation (Candidate Movie)]
**Answer:**
1. Arlington Road
2. Cop Land
3. The Crow
4. Patriot Games
5. The Postman Always Rings Twice
6. The Commitments

7. **Airplane!**
8. Another Stakeout
9. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)
10. Shanghai Surprise
11. Happy, Texas
12. Modulations
13. The Inkwell
14. Johnny 100 Pesos
15. Sorority House Massacre
16. Arguing the World
17. Prom Night
18. Federal Hill
19. Pokémon the Movie 2000
20. The Prophecy II

Table 5.9: Example of the one-shot LLMSRec-Fixed on the ML-1M dataset.

**Fixed Demonstration Example:**

**The User's Movie Profile:**
- Watched Movies: ['Total Recall', 'Aliens', 'Star Wars: Episode VI - Return of the Jedi', 'E.T. the Extra-Terrestrial', 'Forbidden Planet', 'Brazil', 'Star Trek: First Contact', 'Star Trek: The Wrath of Khan', 'Sneakers', 'Galaxy Quest', 'Contact', 'Village of the Damned', 'Being John Malkovich', 'Waiting for Guffman', 'Clerks', 'American Beauty', 'Toy Story 2', 'Shakespeare in Love', 'Toy Story', 'Flirting With Disaster', 'Smoke Signals', 'Pulp Fiction', 'Erin Brockovich', 'Chicken Run', 'Shanghai Noon', 'Gladiator', 'The Wizard of Oz', 'The Producers', "Singin' in the Rain", 'The Sound of Music', 'Snow White and the Seven Dwarfs', 'Fantasia', 'Sleeping Beauty', 'West Side Story', 'Cinderella', 'The Little Mermaid', 'Holiday Inn', 'James and the Giant Peach', 'Dumbo', 'The Lion King', 'The Nightmare Before Christmas', 'The Rocky Horror Picture Show', 'Oliver!', 'Pocahontas', 'Everyone Says I Love You', 'South Pacific', 'Victor/Victoria', 'Tank Girl', 'His Girl Friday', 'The Maltese Falcon']

**The User's Potential Matches:**
- Candidate Movies: ['The Shop Around the Corner','Kicking and Screaming','Harlem','The Good Mother','Oscar and Lucinda (a.k.a. Oscar & Lucinda)','Roman Holiday','Sinbad and the Eye of the Tiger','Gremlins', "Freddy's Dead: The Final Nightmare",'One Magic Christmas','The Scarlet Letter','Alaska','Red Firecracker, Green Firecracker','Bulletproof','The Crimson Pirate','Clockers','The Stepford Wives','The Crow','Bitter Moon','Friday the 13th: The Final Chapter']

Based on the user's watched movies, please rank the candidate movies that align closely with the user's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.

Present your response in the format below:
1. [Top Recommendation (Candidate Movie)]
2. [2nd Recommendation (Candidate Movie)]
...
20. [20th Recommendation (Candidate Movie)]

**Answer:**
1. The Shop Around the Corner
2. One Magic Christmas
3. Alaska
4. Gremlins
5. The Good Mother
6. Bitter Moon
7. Bulletproof
8. Freddy's Dead: The Final Nightmare
9. Red Firecracker, Green Firecracker
10. The Scarlet Letter
11. Sinbad and the Eye of the Tiger
12. Kicking and Screaming
13. Roman Holiday
14. Oscar and Lucinda (a.k.a. Oscar & Lucinda)
15. The Crow

16. Clockers
17. Harlem
18. The Stepford Wives
19. The Crimson Pirate
20. Friday the 13th: The Final Chapter

---

**Test User:**

Learn from the above demonstration examples to solve the following test example.
**The User's Movie Profile:**
- Watched Movies: ['0. Nikita (La Femme Nikita)', '1. Wallace & Gromit: The Best of Aardman Animation', '2. Unforgiven', '3. Batman: Mask of the Phantasm', '4. The Fugitive', '5. Out of Sight', '6. Twelve Monkeys', '7. GoodFellas', '8. Fight Club', '9. Groundhog Day', '10. A Bug"s Life', '11. Tombstone', '12. Aladdin', '13. Beauty and the Beast', '14. Grosse Pointe Blank', '15. Election', '16. Leaving Las Vegas', '17. Total Recall', '18. A Few Good Men', '19. Pleasantville', '20. Jerry Maguire', '21. Pretty Woman', '22. Contact', '23. True Lies', '24. Waking Ned Devine', '25. Romeo Must Die', '26. Mission: Impossible 2', '27. Mission to Mars', '28. Killer, The (Die xue shuang xiong)', '29. Blade Runner', '30. The Princess Bride', '31. Brazil', '32. Henry V', '33. Amadeus', '34. The Right Stuff', '35. The Terminator', '36. Stand by Me', '37. Back to the Future', '38. This Is Spinal Tap', '39. Gandhi', '40. Star Trek: The Wrath of Khan', '41. Ghostbusters', '42. Mad Max 2 (a.k.a. The Road Warrior)', '43. A Fish Called Wanda', '44. Trading Places', '45. Chariots of Fire', '46. Time Bandits', '47. Who Framed Roger Rabbit?', '48. Indiana Jones and the Last Crusade', '49. Die Hard']
**The User's Potential Matches:**
- Candidate Movies: ['0. Sorority House Massacre', '1. Arguing the World', '2. Cop Land', '3. The Inkwell', '4. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)', '5. Airplane!', '6. Pokémon the Movie 2000', '7. Johnny 100 Pesos', '8. Happy, Texas', '9. Arlington Road', '10. The Prophecy II', '11. Prom Night', '12. The Commitments', '13. Federal Hill', '14. Another Stakeout', '15. The Postman Always Rings Twice', '16. The Crow', '17. Shanghai Surprise', '18. Patriot Games', '19. Modulations']
Based on the user's watched movies, please rank the candidate movies that align closely with John's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.
Present your response in the format below: 1. [Top Recommendation (Candidate Movie)] 2. [2nd Recommendation (Candidate Movie)] ... 20. [20th Recommendation (Candidate Movie)]
**Answer:**
1. The Crow
2. Shanghai Surprise
3. The Commitments
4. Arlington Road
5. The Postman Always Rings Twice

6. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)
7. Cop Land
8. Happy, Texas
9. Another Stakeout
10. **Airplane!**
11. Federal Hill
12. Johnny 100 Pesos
13. The Prophecy II
14. Modulations
15. Patriot Games
16. Pokémon the Movie 2000
17. Sorority House Massacre
18. Prom Night
19. Arguing the World
20. The Inkwell

Table 5.10: Example of the zero-shot LLMSRec on the ML-1M dataset.

**Test User:**

Learn from the above demonstration examples to solve the following test example.
**The User's Movie Profile:**
- Watched Movies: ['0. Nikita (La Femme Nikita)', '1. Wallace & Gromit: The Best of Aardman Animation', '2. Unforgiven', '3. Batman: Mask of the Phantasm', '4. The Fugitive', '5. Out of Sight', '6. Twelve Monkeys', '7. GoodFellas', '8. Fight Club', '9. Groundhog Day', '10. A Bug"s Life', '11. Tombstone', '12. Aladdin', '13. Beauty and the Beast', '14. Grosse Pointe Blank', '15. Election', '16. Leaving Las Vegas', '17. Total Recall', '18. A Few Good Men', '19. Pleasantville', '20. Jerry Maguire', '21. Pretty Woman', '22. Contact', '23. True Lies', '24. Waking Ned Devine', '25. Romeo Must Die', '26. Mission: Impossible 2', '27. Mission to Mars', '28. Killer, The (Die xue shuang xiong)', '29. Blade Runner', '30. The Princess Bride', '31. Brazil', '32. Henry V', '33. Amadeus', '34. The Right Stuff', '35. The Terminator', '36. Stand by Me', '37. Back to the Future', '38. This Is Spinal Tap', '39. Gandhi', '40. Star Trek: The Wrath of Khan', '41. Ghostbusters', '42. Mad Max 2 (a.k.a. The Road Warrior)', '43. A Fish Called Wanda', '44. Trading Places', '45. Chariots of Fire', '46. Time Bandits', '47. Who Framed Roger Rabbit?', '48. Indiana Jones and the Last Crusade', '49. Die Hard']
**The User's Potential Matches:**
- Candidate Movies: ['0. Sorority House Massacre', '1. Arguing the World', '2. Cop Land', '3. The Inkwell', '4. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)', '5. Airplane!', '6. Pokémon the Movie 2000', '7. Johnny 100 Pesos', '8. Happy, Texas', '9. Arlington Road', '10. The Prophecy II', '11. Prom Night', '12. The Commitments', '13. Federal Hill', '14. Another Stakeout', '15. The Postman Always Rings Twice', '16. The Crow', '17. Shanghai Surprise', '18. Patriot Games', '19. Modulations']
Based on the user's watched movies, please rank the candidate movies that align closely with John's preferences.
- You ONLY rank the given Candidate Movies.
- You DO NOT generate movies from Watched Movies.
Present your response in the format below: 1. [Top Recommendation (Candidate Movie)] 2. [2nd Recommendation (Candidate Movie)] ... 20. [20th Recommendation (Candidate Movie)]
**Answer:**
1. The Commitments
2. Patriot Games
3. Arlington Road
4. Cop Land
5. The Crow
6. The Postman Always Rings Twice
7. Shanghai Surprise
8. Another Stakeout
9. Modulations

10. Pokémon the Movie 2000
11. The Inkwell
12. Happy, Texas
13. The Prophecy II
14. Johnny 100 Pesos
15. Lovers of the Arctic Circle, The (Los Amantes del Círculo Polar)
16. Arguing the World
17. Federal Hill
18. Prom Night
19. Sorority House Massacre
20. **Airplane!**

## 5.6 Summary

This paper investigates in-context learning (ICL) for LLM-based sequential recommendation. Our study identifies key factors such as instruction format and demonstration selection that influence ICL's effectiveness. We further introduce the LLMSRec-Syn method which utilizes our proposed aggregated demonstration to efficiently incorporate relevant information from multiple training users. In our experiments conducted on three datasets, LLMSRec-Syn consistently outperforms existing LLM-based sequential recommendation methods.

# Chapter 6

# Post-hoc Explanation of Next-Item Recommendation with Large Language Models

In this chapter, we explore the evaluation of LLM's ability to explain recommended items in sequential recommendation tasks. We first present the motivation of this study as well as the objective (Section 6.1). After reviewing the existing approaches (Section 6.2), we introduce our proposed Framework for LLM-based EXplanation of Next-Item Recommendation (FLEX) (Section 6.3) which supports an automatic evaluation of an LLM's ability to generate plausible post-hoc explanations. FLEX focuses on two prevalent perspectives of explanation: content filtering (CTF) and collaborative filtering (CLF). In Section 6.4, we describe two benchmark datasets created for both CTF and CLF explanations using MovieLens. Finally, we present the evaluation of the proposed framework on our created benchmark datasets (Section 6.5).

## 6.1 Motivation and Objective

Explainability is an important feature of a recommender system that measures how well the system elicits user satisfaction and appreciation of system transparency [181, 22]. Recently, various methods have been developed to provide recommendation explanations. These include *template-based explanation* [176, 19], *generation-based natural language explanations* [70, 72], *significant information* (such as *attributes* [21, 77, 76] and *important items* [83], and *important features* [163]), and *knowledge graph reasoning* methods [143, 155]. We shall elaborate these methods in Section 6.2.

As Large Language Models (LLMs), such as ChatGPT and GPT-4, show remarkable capabilities in NLP and reasoning tasks [105, 165], using LLMs to perform recommender systems has become an important research topic [160]. There are at least two advantages to using LLMs for recommendation. Firstly, we can tap on the vast knowledge of LLMs to improve ID-based recommendation models [68, 97]. Secondly, the inference ability of LLMs can be exploited to infer the to-be-recommended items [138, 51]. The LLM-based recommendation works focus on: (a) enhancing existing recommendation models by incorporating semantic information from text corpora [68], (b) using LLMs as recommenders [138, 51], and (c) utilizing LLMs as agents to make decisions and interact with existing recommendation models [38].

While LLMs show promising recommendation performance[51, 138], very few works have explored the use of LLMs to explain recommendation results. These LLMs include existing general-purpose LLMs and specially pre-trained or fine-tuned LLMs. Small LLMs are also trained to have strong explanatory abilities. For instance, the CoT Collection [60] introduces a large instruction tuning dataset across over 1000 tasks. Each data example includes step-by-step rationales. This instruction tuning dataset is used to train a smaller LLM that can generate rationales followed by an answer. In this chapter, we do not cover the methods of training models to generate post-hoc explanations as part of this dissertation. They will be explored in our future work. Unlike the evaluation of recommendation accuracy, the evaluation of LLM's ability to explain the recommended items comes with several challenges which we will try to address in this chapter. Firstly, it is challenging to solicit explanations from the LLMs that allow us to evaluate their performance. As many LLMs are not able to return detailed enough and "structured" explanations, we will introduce specially designed prompts to guide the LLMs to perform explanation in steps. Secondly and more importantly, there is a lack of a single and clear ground truth for evaluating the explanation of each recommendation. For example, a user A may be recommended an action movie X due to A's preference for action movies, or X having been watched by several A's friends recently, or X enjoying high review ratings. It may be the case one of them is correct, some of them are correct, or none of them is correct.

Instead of focusing on a single ground truth explanation that approximates the subjectivity and imagination of each human mind [22], we introduce a **F**ramework for **L**LM-based **EX**planation of Next-Item Recommendation (**FLEX**) that supports an automatic evaluation of a LLM's ability to generate *plausible* post-hoc explanations. We say a post-hoc explanation is plausible if it is based upon some known recommendation principles. In this work, we adopt both content filtering (CTF) and collaborative filtering (CLF) principles to determine if an explanation is plausible. Specifically, LLM-based content filtering explanation attempts to answer the question: *Can one explain whether the attributes of the interacted items could have induced the user to choose the given next item?* To explain from the collaborative filtering perspective, LLMs need to answer the question: *Can one explain whether the similarity between the items user A has interacted with and the items user B has interacted with could have induced user B's to choose the given next item?* To explain next-item recommendation from the CTF perspective, we ask LLMs to predict the genres that are common between the genres of the next movie and the popular genres among the movies that the user has watched. To explain next-item recommendation from the CLF perspective, we ask LLMs to determine if the user's next item can be adopted by another user through considering their common movies and common genres, and next item choices. To evaluate the explanation abilities of LLMs, we create two benchmark datasets for both CTF and CLF explanations using MovieLens. The preliminary results on our proposed benchmark dataset show that LLMs yield promising explanation performance when they are guided to perform CTF and CLF reasoning in FLEX prompting.

## 6.2 Existing Explanation Approaches

Among of the explainable recommendation works, most focus on improving the recommendation accuracy by adding explanation component to the known recommendation models. The methods include pre-defined templates [176, 19], natural language generation [176, 19], knowledge graph paths [143, 155], item features or attributes [21, 142, 77, 83], and reasoning rules [16, 187].

The Explicit Factor Model [176] extracts explicit product features (i.e., aspects) and user opinions through phrase-level sentiment analysis on user reviews. It then fills these features into a fixed template such as, "*You might be interested in [feature], on which this product performs well*", to generate an explanation. Li et al. [72] presents a Personalized Transformer, which is trained to predict the natural language explanation for the given user and item IDs. Cafe [155] generates user profiles as coarse sketches of user behaviors and then subsequently guides a path-finding process to derive reasoning paths for recommendations. Liu et al. [83] factorize input data into several factors (e.g., attributes) and use a Graph Convolutional Network to derive representation vectors. The model uses its weights to identify important factors for providing explanations. NCR [16] utilizes a modularized reasoning architecture to learn logical operations like AND, OR, and NOT as neural modules. These modules assist with implication reasoning, enabling the recommendation to make informed decisions.

In this chapter, our work focuses on generating explanations and evaluating them using LLMs.

Post-hoc explanation offers insights into predictions after a model has been trained. The existing post-hoc techniques are not LLM-based and they include methods that assess feature importance [115, 89], model visualizations [168, 8], and interpretable distillation [128, 42]. For post-hoc explanations for sequential recommendations, we can use post-hoc explanation methods such as 1) proxy models like LIME[114], 2) attention weights [15], 3) template-based textual explanation generation [71], or 4) cosine similarity over item representations drawn from the model itself [98]. These methods can help identifying items in the user sequence and provide insight into why the recommendation models recommend a particular item. In this thesis, we mainly focus on developing prompt-based post-hoc explanation using LLMs. LLMs provide an opportunity to treat post-hoc explanation as a text generation task, as LLMs can generate human readable, knowledge-enriched, and reasoning-based explanations [122]. Recently, chain-of-thought prompting method is proposed to ask LLMs to generate its reasoning step-by-step before arriving at an answer [145]. Research has shown that LLMs can generate post-hoc attributions of important features by prompting [64, 63]. When combining in-context learning with post-hoc explanations, LLMs are able to improve its performance in tasks that require reasoning and language understanding [63]. Nevertheless, research on developing LLM-based methods for post-hoc explanation of recommendation results is still unexplored. In this work, we will focus on generating and evaluating explanations using LLMs for next-item recommendation.

## 6.3 Proposed Framework

The FLEX framework consists of three steps, namely, (1) knowledge augmentation, (2) explanation generation, and (3) evaluation. Each step can be perform by prompting the LLMs. As content filtering (CTF) and collaborative filtering (CLF) are the two well-established recommendation principles, we focus on how FLEX can guide LLMs to generate post-hoc explanations based on CTF and CLF principles. CTF and

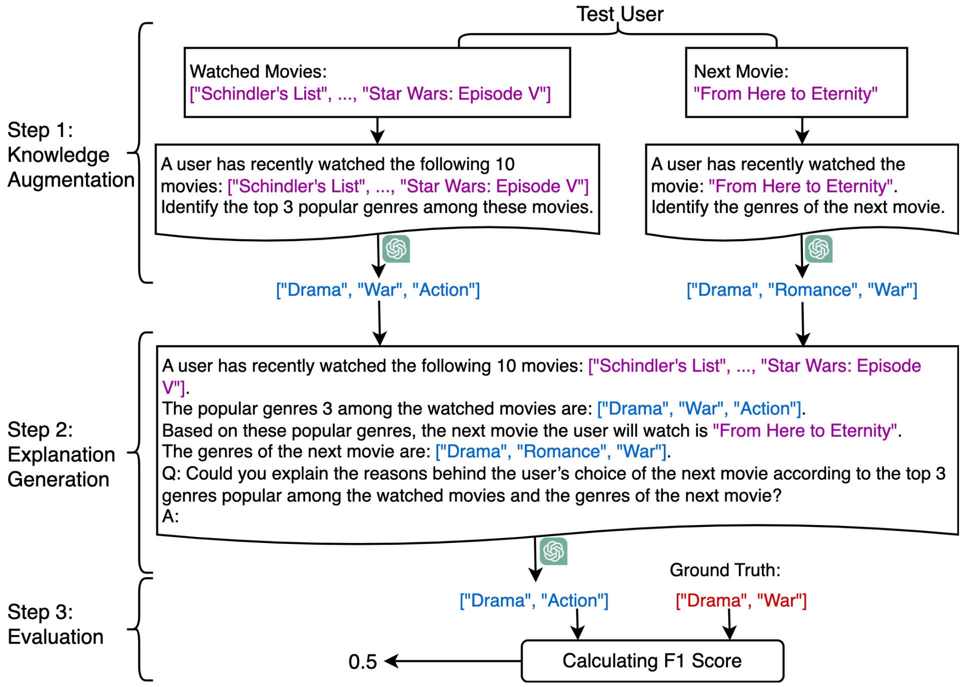


Figure 6.1: CTF in the proposed FLEX framework.

CLF in the FLEX framework are shown as Figure 6.1 ans Figure 6.2 respectively. FLEX also evaluates the generated explanations against the ground truths that are determined based on their plausibility. FLEX is also general enough to be extended for other plausible post-hoc explanations.

For the rest of this chapter, we denote a test user $u_t$ by the $(s_{u_t}, \hat{v}^{u_t})$ tuple where $s_{u_t}$ denotes the sequence of past interacted items and $\hat{v}^{u_t}$ denotes the recommended next-item. The objective of FLEX is to get a LLM to generate two explanations $e_i^{ctf}$ and $e_i^{clf}$ to state if $\hat{v}^{u_t}$ can be explained by CTF or CLF respectively. The overall FLEX's approach of post-hoc explanation by CLF and CTF is as follows.

**Content filtering (CTF).** The CTF principle requires the recommended next-item $\hat{v}^{u_t}$ to be explained by its features matching the user preferred features observed from the user history, i.e., $s_{u_t}$. We denote the explanation for content filtering by $e_{u_t}^{ctf} = (A_{u_t}^{ctf}, r_{u_t}^{ctf})$ where $A_{u_t}^{ctf}$ denotes the set of item features relevant to the explanation, followed by a natural language rationale $r_{u_t}^{ctf}$. $A_{u_t}^{ctf}$ consists of features of $\hat{y}_{u_t}$ and the preferred features of user $u_t$, i.e., features that are common among the past interacted items of $u_t$. In Step 1 (Knowledge Augmentation), the set of features

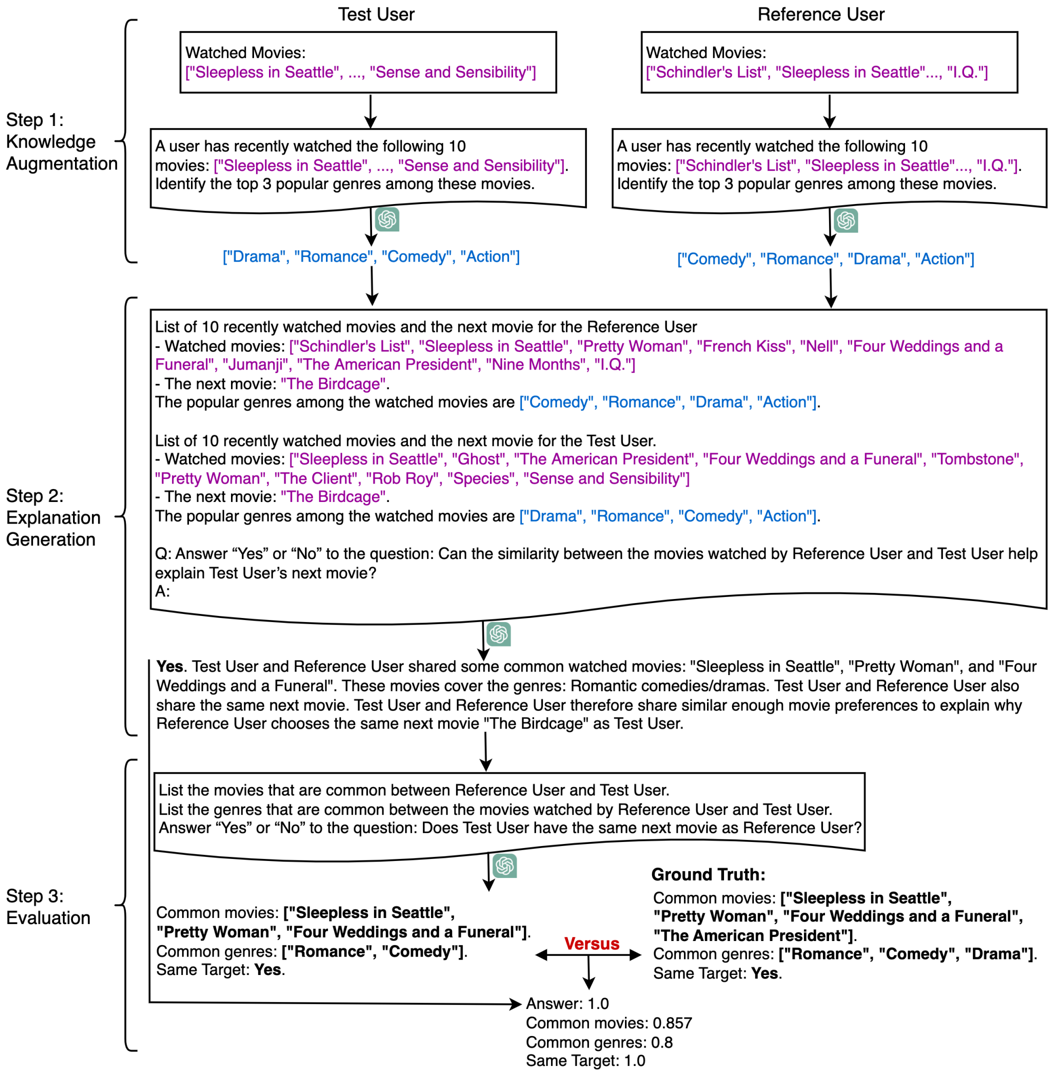


Figure 6.2: CLF in the proposed FLEX framework.

are extracted for the historical user-item interactions and the recommended next-item. In Step 2 (Explanation Generation), FLEX requires the two above sets of features to be analysed for generating plausible explanation as the final list of relevant features that supports the content filtering principle. In Step 3 (Evaluation), FLEX evaluates the accuracy of the generated explanation, i.e., list of relevant features, against the ground truth relevant features.

**Collaborative filtering (CLF).** The CLF principle assumes that the recommended next-item of the test user $u_t$ should be among the ones that have been adopted by other similar users. While there may be multiple users who are similar to $u_t$ and their existence is clearly beyond the knowledge of LLMs, we introduce a *reference user*

$u_j$ for LLMs to determine if $u_j$ can serve as a similar user and if CLF can explain why $\hat{v}^{u_t}$ is recommended to $u_t$. We denote $u_j$ by the $(s_{u_j}, v_*^{u_j})$ tuple. A post-hoc explanation $e_{u_t}^{clf}$ is to be generated to answer the question: Is user $u_j$ similar to $u_t$ so as to apply CLF to explain why $\hat{v}^{u_t}$ is recommended to $u_t$? To answer this question, we need to measure the similarity between $u_t$ and $u_j$, and to determine if $\hat{v}^{u_t}$ is also the next-item of $u_j$ (See Sect. 6.3.3). In Step 1 (Knowledge Augmentation), FLEX extracts features of historical items of $u_t$ and $u_j$. In Step 2 (Explanation Generation), FLEX determines the similarity between $u_t$ and $u_j$ by the common item(s) between their historical items or features of their historical items. FLEX also seeks to determine if the test user's recommended item is the same as that of the reference user because of the similarity. In Step 3 (Evaluation), we evaluate the accuracy of the generated explanation, i.e., common items, common features between historical items, and common next-item.

In the following, we elaborate the prompt designs for these three steps in FLEX.

### 6.3.1 Step 1: Knowledge Augmentation

**Content filtering (CTF).** We devise the knowledge augmentation step to extract features (e.g., genres) of test user's historical item interactions $s_{u_t}$ and that of the user's recommended item $\hat{v}^{u_t}$. We design the corresponding the following prompt to perform the knowledge augmentation step for $s_{u_t}$ as follows[1]:

```
A user has recently watched the following 10 movies:  ...
Identify the top 3 popular genres among these movies.
```

We design another similar prompt to perform the knowledge augmentation step for $\hat{v}^{u_t}$ as follows:

[1]The "..." denotes a list of movie titles recently watched by the test user.

```
A user has recently watched the movie: ...

Identify the genres of the next movie.
```

**Collaborative filtering (CLF).** In CLF, the prompt for knowledge augmentation extracts features from the test or reference user's history. It is shown as follows:

```
A user has recently watched the following 10 movies: ...

Identify the top 3 popular genres among these movies.
```

### 6.3.2 Step 2: Explanation Generation

**Content filtering (CTF).** We design the explanation prompt to feed the test user, the recommended item, and the augmented knowledge to LLM, and ask the LLM to provide its explanation as shown below:

```
A user has recently watched the following 10 movies: ...

The popular genres 3 among the watched movies are: ...

Based on these popular genres, the next movie the user will

watch is ...

The genres of the next movie are: ...

Could you explain the reasons behind the user's choice of the

next movie according to the top 3 genres popular among the

watched movies and the genres of the next movie?
```

**Collaborative filtering (CLF).** The prompt to obtain explanation for CLF is:

```
List of 10 recently watched movies and the next movie for the
Reference User.
- Watched movies: ...
- The next movie: ...
The popular genres among the watched movies are ...
List of 10 recently watched movies and the next movie for the
Test User.
- Watched movies: ...
- The next movie: ...
The popular genres among the watched movies are ...
Can the similarity between Reference User's watched movies
and the Test User's watched movies help to explain Test
User's next movie?
```

### 6.3.3 Step 3: Evaluation

In the evaluation step, we extract labels from the explanation text and compare them against the ground truths in the computation of explanation effectiveness measures.

**Content Filtering (CTF).** The predicted explanation text in CTF consists of a list of features, which is to be matched with the ground truth feature labels. Therefore, we can directly compute the F1 score over the predicted list of genres and the ground truth list of genres, as illustrated in Step 3 of Figure 6.1.

**Collaborative Filtering (CLF).** To evaluate the ability of LLMs to perform CLF correctly, we include four types of reference user $u_j$. There are three conditions to consider: whether the test user or reference user have the same target item, whether there are overlapping items, and whether there are overlapping features. Each of these conditions presents two options, creating eight possible types. However, we only consider four of these types. The others, such as the type that includes different target items, no overlapping items, and no overlapping features, are either simple to predict or can be subsumed within the four types we are considering.

1. $u_j$ has the next-item identical to the recommended next-item of $u_t$ ($v^{u_j} = \hat{v}^{u_t}$), and $s_{u_j}$ is similar to $s_{u_t}$ due to overlapping interacted items.

2. $u_j$ has the next-item identical to the recommended next-item of $u_t$ ($v^{u_j} = \hat{v}^{u_t}$), and $s_{u_j}$ is similar to $u_{u_t}$ due to overlapping features between their interacted items but not overlapping interacted items.

3. $u_j$ has the next-item identical to the recommended next-item of $u_t$ ($v^{u_j} = \hat{v}^{u_t}$), and $s_{u_j}$ is not similar to $s_{u_t}$ as they do not share common interacted items nor common features between their interacted items.

4. $u_j$ does not have the next-item same as the recommended next-item of $u_t$ ($v^{u_j} \neq \hat{v}^{u_t}$) and $s_{u_j}$ do not share common interacted items as $s_{u_t}$, but $s_{u_t}$ and $s_{u_j}$ share common features between their interacted items.

We also design the following evaluation prompts to extract CLF-relevant answer labels from LLMs:

```
List the movies that are common between Reference User and
Test User.
List the genres that are common between the movies watched by
Reference User and Test User.
Answer ``Yes'' or ``No'' to the question: Does Test User
have the same next movie as Reference User?
```

The example of answer extraction is shown in Step 3 of Figure 6.2.

## 6.4 Dataset Construction

To evaluate the post-hoc explanation ability of LLMs, we create two new benchmark datasets with ground truth labels derived from the MovieLens Latest dataset[2]. To evaluate the LLM-based explanation from the CTF perspective, we first construct a CTF explanation test dataset. Specifically, we randomly select 500 test users such that each user has a sequence of 10 interacted items as his/her history and the item after these 10 items as the ground truth next-item. Using the genre features provided by MovieLens, we determine each user's preferred genres by determining the top K popular movie genres in the user's movie history or popular movie genres using some threshold. In this work, we use the top 3 popular genres by default. By comparing the popular genres with the genres of the recommended next movie of the user, we derive the overlapping genres as the ground truth labels for this CTF explanation dataset. The number of ground truth labels can be 0, 1, 2, and 3. Their proportions are 15%, 46%, 27.8%, and 11.2% respectively.

To evaluate CLF explanations, we create another dataset using selected CTF test users and another set of users from the original data as CTF training users. From the latter, we select the reference users. For each test user, we construct four reference users based on whether they have the same next-movie and have overlapping genres or movies with the test user in their movie histories:

(1) A reference user with next-movie identical to the recommended next-movie of the test user, and movie history having high genre similarity ($> 0.6$) and acceptable movie similarity (>0.2) with that of the test user;

(2) A reference user with next-movie identical to the recommended next-movie of the test user, and movie history having high genre similarity (>0.6) with that of the test user;

(3) A reference user with next-movie identical to the recommended next-movie

[2]https://grouplens.org/datasets/movielens/latest/

of the test user, and movie history having low genre similarity (<0.3) and no overlapping movies; and

(4) A reference user with next-movie different from the recommended next-movie of the test user.

Based on the above conditions, we collect reference users for each test user. To balance the data, we exclude test users who do not have all 4 reference users. The final CLF dataset consists of 80 test users and $80 \times 4 = 320$ (test user,reference user) pairs.

We evaluate the following aspects of CLF explanation generated by the LLM:

- CLF applicability (CLF_App): This refers to the ability of the generated explanation to determine if the reference user is applicable (i.e., type (1) or (2)), or not-applicable (i.e., type (3) or (4)) with respect to the test user. To report the accuracy of this aspect of explanation, we introduce the F1 metric which is defined by the accuracy of determining if the reference user is applicable or not-applicable (Yes or NO).
- Overlapping genres between movie histories of reference and test users (Ov_Movies): We report the F1 score that measures the accuracy of this aspect of explanation.
- Overlapping watched movies between test and reference users (Ov_Genres)
- Identical next movie between the test and reference users (Same_Next)

For each of the above aspects, we report the corresponding F1 score.

## 6.5 Experiment

In this section, we evaluate the effectiveness of LLM-based post-hoc explanation for next-item recommendation.

| Method | Movie | |
|---|---|---|
| | Recall | F1 |
| Random | 0.0867 | 0.0636 |
| POP | 0.4846 | 0.3173 |
| FLEX (Single-Step) | 0.7182 | 0.5400 |
| FLEX (Multi-Step) | **0.7868** | **0.5707** |

Table 6.1: Recall and F1 results on the CTF test set (MovieLens)(Best results are **boldfaced**.)

### 6.5.1 Experimental Setup

**Baselines.** For CTF, we employ two baseline models for comparison. The "Random" model randomly selects three genres as the predicted common genres between popular genres of the test user's watched movies and genres of the next movie. The "POP" model considers genres popular among ALL users' watched movies, identifying the top three for explanations. Our framework, FLEX, introduces a multi-step prompting to generating explanations. We also offer a variant version called "FLEX (Single-Step)" which uses single-step prompt to generate genre lists and explanation text from LLMs, based on a user's watched movies and the next movie. The experiment is conducted on the CTF dataset.

For CLF, we evaluate CLF explanations on the CLF dataset using both single and multi-step prompting as well.

**Implementation Details.** In the FLEX framework, we design in-context demonstrations to guide LLMs in achieving our goals. Specifically, we manually create three demonstration examples for the CTF explanation process and four demonstration examples for each step of the CLF explanation. For the LLM, our primary experiments use ChatGPT (gpt-3.5-turbo-1106) with a default temperature setting of T=0.

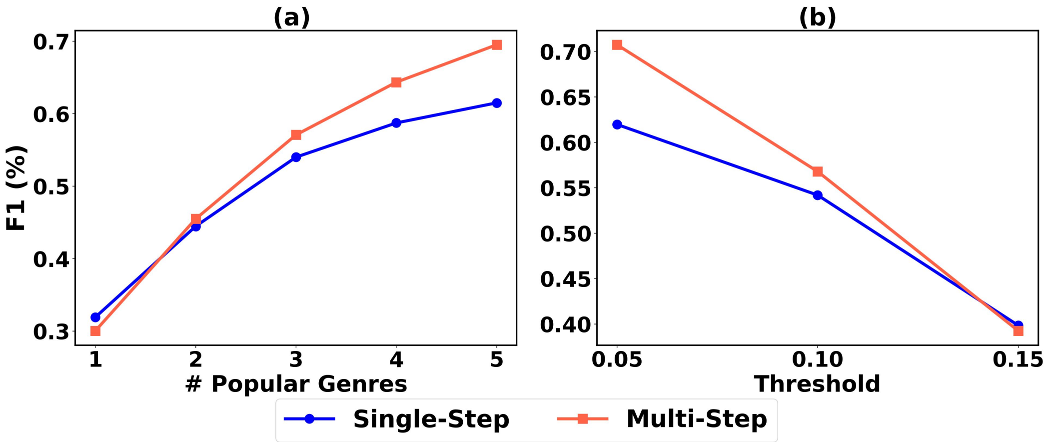


Figure 6.3: F1 results of different prompting methods on the CTF test set (MovieLens) with different overlap between the recommended movie's genres and the popular genres of watched movies controlled by: (a) number of overlapping genres; or (b) relative popularity threshold that determines popular genres among watched movies.

### 6.5.2 Results and Analysis

**Results for CTF explanation.** The results in Table 6.1 show that the FLEX (Multi-Step) method significantly outperforms the other methods. Although the FLEX (Single-Step) method is not as effective as the Multi-Step counterpart, it still significantly outperforms the POP and Random baselines. These results highlight two main insights: (a) LLMs generate more accurate explanations than the simple baselines, and (b) a multi-step prompting approach, which incorporates open-world knowledge into LLM inputs, improves their explanation capabilities beyond relying on their built-in knowledge alone.

**Impact of popular genres on constructing ground truth labels for CTF explanation evaluation.** In this study, we explore how varying the set size of popular genres impacts the creation of ground truth labels for evaluating the CTF explanations. As mentioned in Section 6.3, we demonstrate that ground truth labels can be adjusted by either modifying the number of top popular genres or by setting a threshold to determine genre popularity. We adjusted the number of popular genres $K$ from 1 to 5 (Figure 6.3(a)) and varied the popularity threshold between 0.5, 0.1, and 0.15 (Figure 6.3(b)). Our findings show that changes in ground truth labels impact

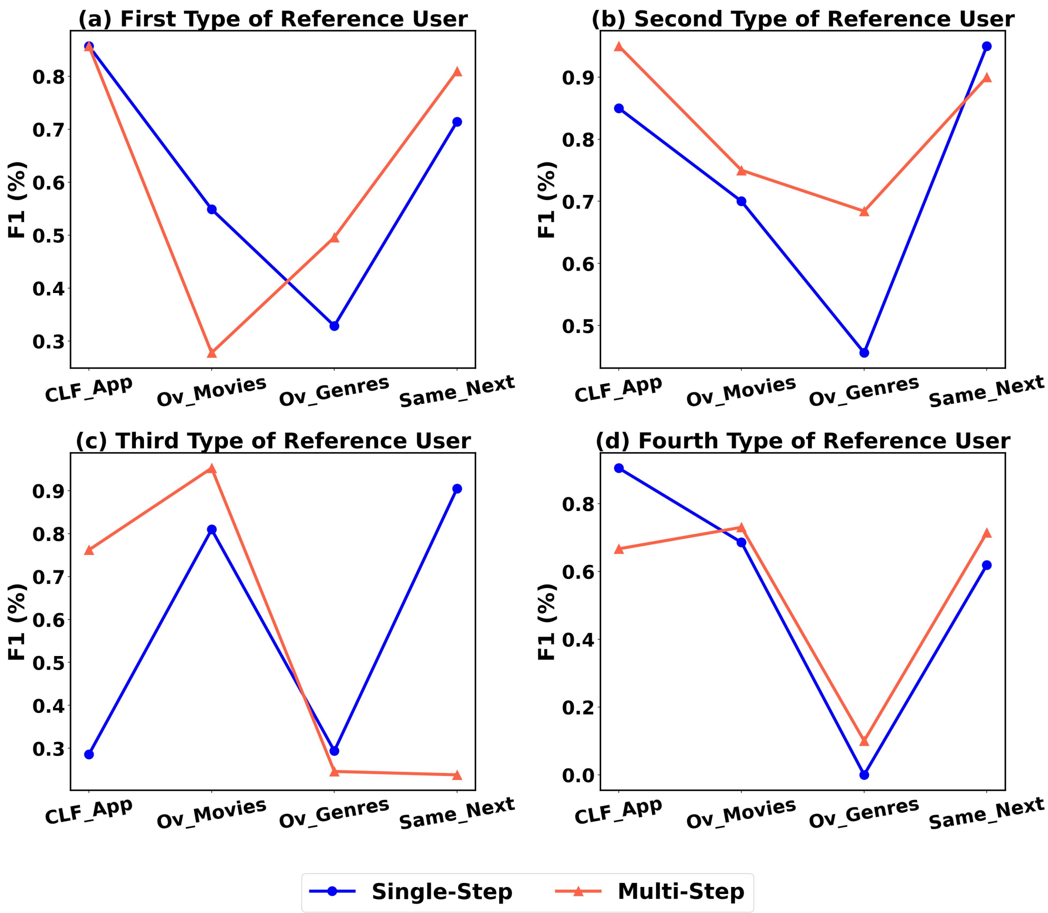


Figure 6.4: F1 results for CLF-applicability prediction for four types of reference users, common movies between reference and test users, common genres between movies watched by reference and test users, and same next-movie between reference and test users on the CLF test set (MovieLens).

explanation evaluations. Specifically, increasing the number of labels simplifies the evaluation process and enhances explanation performance, while decreasing the popularity threshold broadens the range of genres considered as ground truth, affecting evaluations.

**Results for CLF explanation across four types of reference users.** The results in Figure 6.4 indicate that the performance of FLEX (Multi-Step) and FLEX (Single-Step) varies across the four types of reference users. Specifically, ChatGPT accurately predicts the applicability of explanations for the first two user types but struggles with the third and fourth. The Multi-Step approach excels in identifying common genres, aided by the knowledge augmentation prompting step. Although we expect a strong correlation among the four prediction types, the results suggest a weak

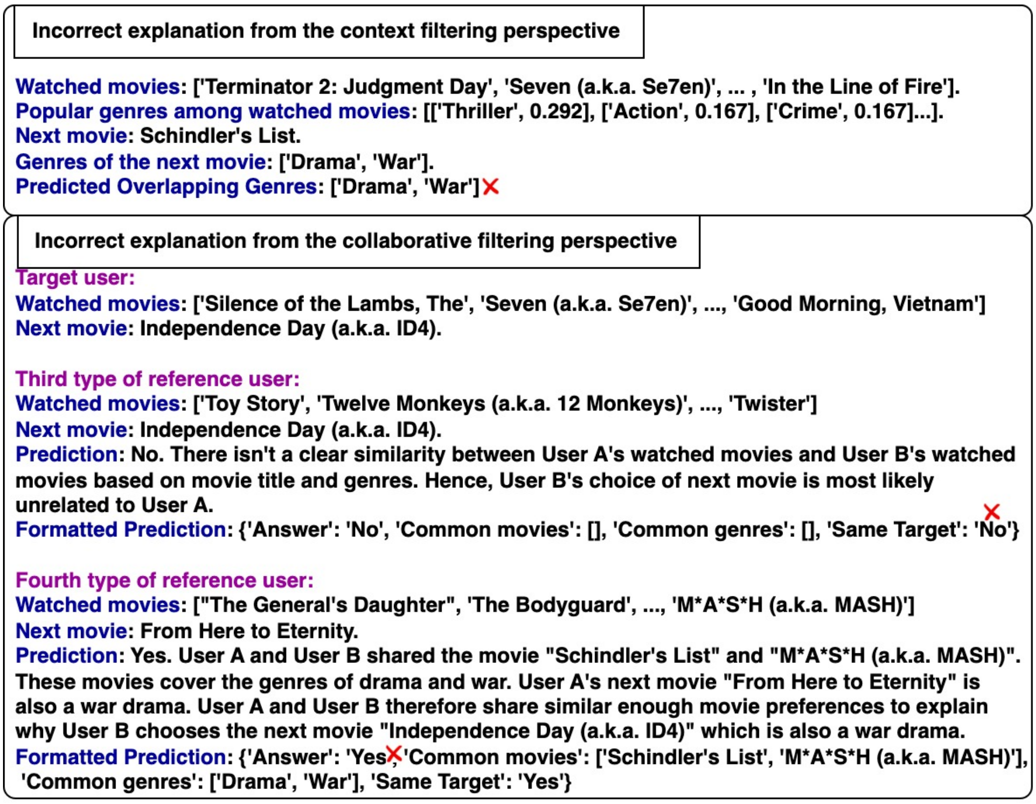


Figure 6.5: Error Analysis.

connection between explanation applicability, common movies, common genres, and same targets. The results of FLEX (Multi-Step) and FLEX (Single-Step) are inconsistent across different types of reference users. Further exploration will be carried out in future work.

**Error Analysis.** Figure 6.5 illustrates error cases in CTF and CLF explanations. In the CTF example, ChatGPT appears to replicate genres of the next movie rather than identifying common ones. The first CLF example reveals that the response inaccurately indicates a wrong prediction for the same target due to a lack of information about the next movie in the generated explanation text.

In the second CLF case example, ChatGPT incorrectly predicts the same target, influenced by numerous shared movies and genres, leading to an incorrect prediction for CLF applicability.

## 6.6 Summary

In this paper, we propose the novel FLEX framework that evaluates the use of LLMs to explain post-hoc next-item recommendation results using the CTF and CLF principles. FLEX includes a 3-step prompting strategy to guide LLMs to generate features associated with items, explanation text as well as the answer labels extracted from explanation text so as to evaluate them against the plausible ground truths. The preliminary results on our proposed benchmark dataset show that ChatGPT yields promising explanation performance when it is guided to perform CTF and CLF reasoning in multi-step FLEX prompting. This findings points to the possibility to develop effective LLM-based explainers for a wide range of recommendation models developed for different domains. The extension of FLEX to incorporate other recommendation principles is also another interesting direction for future research.

# Chapter 7

# Conclusion and Future Work

## 7.1 Summary of Contributions

In this dissertation, we provide an overview of sequential recommendation, covering topics from representation learning, reasoning, to explanation, as well as discuss our contributions to the field, from before the era of large language models to the present.

In representation learning, we explore the use of explanation methods to generate high-quality views for contrastive learning in sequential recommendation tasks. Initially, we find that existing contrastive learning-based methods suffer from false positive and false negative issues, potentially leading to decreased recommendation performance. Therefore, we introduce several explanation-guided augmentations to generate positive and negative views, mitigating these problems. Additionally, we propose an explanation-guided contrastive learning model for sequential recommendations, based on these augmentations. Our experiments on four real-world benchmark datasets demonstrate the effectiveness, generality, and flexibility of our proposed explanation-guided approach.

For LLM-based reasoning, we investigate how LLM-based reasoners can effectively operate in zero-shot and few-shot settings for sequential recommendations. In the zero-shot setting, we concentrate on designing prompts that enhance recommendations using LLMs. However, applying LLMs to sequential recommendation presents challenges such as task-specific recommendation space and user preference modeling. To address these, we introduce a method that begins with an external module for candidate item generation, followed by a 3-step prompting strategy to capture user preferences and make ranked recommendations. Evaluations on MovieLens 100K and LastFM datasets using LLMs reveal that the proposed method competes well with powerful sequential recommendation models, opening up new intriguing research opportunities to leverage LLMs as recommender systems. In the few-shot setting, we investigate in-context learning for LLM-based sequential recommendation. Our study identifies key factors, such as instruction format and demonstration selection, that influence the effectiveness of in-context learning. We further introduce the LLMSRec-Syn method, which utilizes our proposed aggregated demonstration to efficiently incorporate relevant information from multiple training users. Tested on three datasets, LLMSRec-Syn consistently outperforms existing LLM-based sequential recommendation methods.

For LLM-based explanation, we propose the novel FLEX framework that evaluates the use of LLMs to post-hoc explain next-item recommendation results using the content filtering and collaborative filtering principles. FLEX includes a 3-step prompting strategy to guide LLMs in generating features associated with items, explanation text, and answer labels extracted from the explanation text to evaluate them against plausible ground truths. Preliminary results on our proposed benchmark dataset show that ChatGPT yields promising explanation performance when guided to perform content filtering and collaborative filtering reasoning in multi-step FLEX prompting. These findings point to the possibility of developing effective LLM-based explainers for a wide range of recommendation models developed for different domains.

## 7.2 Future Work

Below, we also list four potential directions worth exploring in the future.

The first potential area for future research is the integration of representation learning and LLM-based reasoning. On the one hand, with representation learning, we don't have to worry about the size of the recommendation space or the candidate item pool. We can calculate the similarity matrix between user representation vectors and candidate item vectors and store them for future use. This allows us to quickly generate a recommendation list based on the recalculated similarity matrix. Furthermore, representation learning-based sequential recommendations primarily focus on local knowledge for a specific domain or task. This type of information is crucial for making accurate recommendations. On the other hand, LLM can provide rich world knowledge and powerful reasoning abilities, which can compensate for the drawbacks of representation learning-based methods.

The second future research direction is to leverage post-hoc explanations generated by LLMs to further improve recommendation accuracy. Recent research has shown that chain-of-thought prompting [145] can outperform standard prompting as it can elicit step-by-step reasoning in LLMs to derive the final answer more accurately and confidently. Compared to standard prompting, which teach LLMs to directly output an answer, chain-of-thought prompting includes demonstrations that include human-annotated rationales which teach LLMs to generate rationales followed by the final answer, thereby improving performance. For post-hoc explanations generated by LLMs, we can consider them as one type of rationale derived from LLMs instead of humans [63]. Then, we can leverage in-context learning including post-hoc explanations to further improve recommendation accuracy.

The third future research direction is to leverage LLMs to help conversational recommendations. Unlike typical recommender systems, conversational recommender systems must understand users' natural language input and historical interactions. They also need to provide natural language responses and suggested items. LLMs can offer distinct advantages for such tasks, such as superb natural language understanding, a wealth of general knowledge, excellent reasoning abilities, and outstanding natural language generation. Recently, He et al. [48] explore the use of LLMs as zero-shot conversational recommenders. However, they find that LLM recommendations are subject to popularity bias in conversational recommendation systems. Moreover, the performance of LLMs varies across different geographical regions. Thus, studying how to incorporate external knowledge and how to utilize in-context learning to mitigate these issues should be intriguing topics.

The fourth future research direction involves integrating visual reasoning abilities from large multimodal models like GPT-4V [101] into recommendation systems. GPT-4, now equipped with vision capabilities, is a powerful and versatile large multimodal model. It has garnered growing interest from the research community for its robust ability to process and interpret both text and image inputs. On the other hand, as multimodal data items increase their presence (e.g., short video, and news), these multimodal models will be required for recommendations. Moreover, multimodal features can help alleviate the issue of data sparsity in recommendation systems [85]. Given GPT-4's capabilities and the growing amount of multimodal data, exploring how to integrate sequential recommender systems with GPT-4 or other open-source large multimodal models should be an interesting direction.

There are still many open questions. We hope the above discussion about future work can inspire further research on combining the advantages of recommendation models and large language models as well as utilizing post-hoc explanations to assist with recommendations.

# Bibliography


[1] S. Abnar and W. Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

[2] S. Agrawal, C. Zhou, M. Lewis, L. Zettlemoyer, and M. Ghazvininejad. In-context examples selection for machine translation. *arXiv preprint arXiv:2212.02437*, 2022.

[3] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.263. URL https://www.aclweb.org/anthology/2020.emnlp-main.263.

[4] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] 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. *arXiv preprint arXiv:2305.00447*, 2023.

[7] J. Bastings, W. Aziz, and I. Titov. Interpretable neural predictions with differentiable binary variables. *arXiv preprint arXiv:1905.08160*, 2019.

[8] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

[9] A. Boz, W. Zorgdrager, Z. Kotti, J. Harte, P. Louridas, D. Jannach, and M. Fragkoulis. Improving sequential recommendations with llms. *arXiv preprint arXiv:2402.01339*, 2024.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish,

A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[12] I. Cantador, P. Brusilovsky, and T. Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

[13] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, and Y. Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 378–387, 2021.

[14] H. Chen, G. Zheng, and Y. Ji. Generating hierarchical explanations on text classification via feature interaction detection. *arXiv preprint arXiv:2004.02015*, 2020.

[15] H. Chen, Y. Li, X. Sun, G. Xu, and H. Yin. Temporal meta-path guided explainable recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 1056–1064, 2021.

[16] H. Chen, S. Shi, Y. Li, and Y. Zhang. Neural collaborative reasoning. In *Proceedings of the Web Conference 2021*, pages 1516–1527, 2021.

[17] J. Chen, L. Chen, C. Zhu, and T. Zhou. How many demonstrations do you need for in-context learning? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11149–11159, 2023.

[18] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[19] T. Chen, H. Yin, G. Ye, Z. Huang, Y. Wang, and M. Wang. Try this instead: Personalized and interpretable substitute recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 891–900, 2020.

[20] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 108–116, 2018.

[21] X. Chen, H. Chen, H. Xu, Y. Zhang, Y. Cao, Z. Qin, and H. Zha. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 765–774, 2019.

[22] X. Chen, Y. Zhang, and J.-R. Wen. Measuring" why" in recommender systems: A comprehensive survey on the evaluation of explainable recommendation. *arXiv preprint arXiv:2202.06466*, 2022.

[23] Y. Chen, Z. Liu, J. Li, J. McAuley, and C. Xiong. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 2172–2182, 2022.

[24] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *ArXiv preprint*, abs/2204.02311, 2022. URL https://arxiv.org/abs/2204.02311.

[25] 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. *arXiv preprint arXiv:2210.11416*, 2022.

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[27] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[28] Z. Cui, J. Ma, C. Zhou, J. Zhou, and H. Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.

[29] S. Dai, N. Shao, H. Zhao, W. Yu, Z. Si, C. Xu, Z. Sun, X. Zhang, and J. Xu. Uncovering chatgpt's capabilities in recommender systems. *arXiv preprint arXiv:2305.02182*, 2023.

[30] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296, 2010.

[31] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, and Z. Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

[32] 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.

[33] Y. Du, D. Luo, R. Yan, H. Liu, Y. Song, H. Zhu, and J. Zhang. Enhancing job recommendation through llm-based generative adversarial networks. *arXiv preprint arXiv:2307.10747*, 2023.

[34] X. Fan, Z. Liu, J. Lian, W. X. Zhao, X. Xie, and J.-R. Wen. Lighter and better: low-rank decomposed self-attention networks for next-item recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1733–1737, 2021.

[35] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*, 2020.

[36] S. Feng, E. Wallace, A. Grissom II, M. Iyyer, P. Rodriguez, and J. Boyd-Graber. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*, 2018.

[37] T. Gao, X. Yao, and D. Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[38] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*, 2023.

[39] F. Gedikli, D. Jannach, and M. Ge. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367–382, 2014.

[40] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). *arXiv preprint arXiv:2203.13366*, 2022.

[41] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[42] W. Ha, C. Singh, F. Lanusse, S. Upadhyayula, and B. Yu. Adaptive wavelet distillation from neural networks through interpretations. *Advances in Neural Information Processing Systems*, 34:20669–20682, 2021.

[43] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[44] J. Harte, W. Zorgdrager, P. Louridas, A. Katsifodimos, D. Jannach, and M. Fragkoulis. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1096–1102, 2023.

[45] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[46] R. He and J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200. IEEE, 2016.

[47] R. He, W.-C. Kang, and J. McAuley. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 161–169, 2017.

[48] Z. He, Z. Xie, R. Jha, H. Steck, D. Liang, Y. Feng, B. P. Majumder, N. Kallus, and J. McAuley. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 720–730, 2023.

[49] B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 843–852, 2018.

[50] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[51] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, and W. X. Zhao. Large language models are zero-shot rankers for recommender systems. *arXiv preprint arXiv:2305.08845*, 2023.

[52] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514, 2018.

[53] S. Jain and B. C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North*, volume 1, pages 3543–3556, Stroudsburg, PA, USA, 2 2019. Association for Computational Linguistics. ISBN 9781950737130. doi: 10.18653/v1/N19-1357. URL http://aclweb.org/anthology/N19-1357.

[54] S. Jain and B. C. Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[55] S. Jain, S. Wiegreffe, Y. Pinter, and B. C. Wallace. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.409. URL https://aclanthology.org/2020.acl-main.409.

[56] J. Ji, Z. Li, S. Xu, W. Hua, Y. Ge, J. Tan, and Y. Zhang. Genrec: Large language model for generative recommendation. *arXiv e-prints*, pages arXiv–2307, 2023.

[57] S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 659–667, 2013.

[58] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

[59] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[60] S. Kim, S. J. Joo, D. Kim, J. Jang, S. Ye, J. Shin, and M. Seo. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. *arXiv preprint arXiv:2305.14045*, 2023.

[61] P.-J. Kindermans, K. Schütt, K.-R. Müller, and S. Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016.

[62] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[63] S. Krishna, J. Ma, D. Slack, A. Ghandeharioun, S. Singh, and H. Lakkaraju. Post hoc explanations of language models can improve language models. *arXiv preprint arXiv:2305.11426*, 2023.

[64] N. Kroeger, D. Ley, S. Krishna, C. Agarwal, and H. Lakkaraju. Are large language models post hoc explainers? *arXiv preprint arXiv:2310.05797*, 2023.

[65] T. Lei, R. Barzilay, and T. Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

[66] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[67] J. Li, Y. Wang, and J. McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*, pages 322–330, 2020.

[68] 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. *arXiv preprint arXiv:2305.13731*, 2023.

[69] J. Li, W. Zhang, T. Wang, G. Xiong, A. Lu, and G. Medioni. Gpt4rec: A generative framework for personalized recommendation and user interests interpretation. *arXiv preprint arXiv:2304.03879*, 2023.

[70] L. Li, L. Chen, and Y. Zhang. Towards controllable explanation generation for recommender systems via neural template. In *Companion proceedings of the web conference 2020*, pages 198–202, 2020.

[71] L. Li, Y. Zhang, and L. Chen. Generate neural template explanations for recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 755–764, 2020.

[72] L. Li, Y. Zhang, and L. Chen. Personalized transformer for explainable recommendation. *arXiv preprint arXiv:2105.11601*, 2021.

[73] L. Li, Y. Zhang, and L. Chen. Personalized prompt learning for explainable recommendation. *arXiv preprint arXiv:2202.07371*, 2022.

[74] S. Li, J. Chen, Y. Shen, Z. Chen, X. Zhang, Z. Li, H. Wang, J. Qian, B. Peng, Y. Mao, et al. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*, 2022.

[75] X. Li, K. Lv, H. Yan, T. Lin, W. Zhu, Y. Ni, G. Xie, X. Wang, and X. Qiu. Unified demonstration retriever for in-context learning. *arXiv preprint arXiv:2305.04320*, 2023.

[76] Y. Li, T. Chen, and Z. Huang. Attribute-aware explainable complementary clothing recommendation. *World Wide Web*, 24:1885–1901, 2021.

[77] Z. Li, W. Cheng, H. Xiao, W. Yu, H. Chen, and W. Wang. You are what and where you are: Graph enhanced attention network for explainable poi recommendation. In *Proceedings of the 30th ACM International conference on information & knowledge management*, pages 3945–3954, 2021.

[78] G. Lin, C. Gao, Y. Li, Y. Zheng, Z. Li, D. Jin, and Y. Li. Dual contrastive network for sequential recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2686–2691, 2022.

[79] S. Lin, J. Hilton, and O. Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

[80] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[81] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[82] J. Liu, C. Liu, R. Lv, K. Zhou, and Y. Zhang. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*, 2023.

[83] N. Liu, Y. Ge, L. Li, X. Hu, R. Chen, and S.-H. Choi. Explainable recommender systems via resolving learning representations. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 895–904, 2020.

[84] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[85] Q. Liu, J. Hu, Y. Xiao, J. Gao, and X. Zhao. Multimodal recommender systems: A survey. *arXiv preprint arXiv:2302.03883*, 2023.

[86] Z. Liu, Y. Chen, J. Li, M. Luo, S. Y. Philip, and C. Xiong. Self-supervised learning for sequential recommendation with model augmentation. 2021.

[87] Z. Liu, Y. Chen, J. Li, P. S. Yu, J. McAuley, and C. Xiong. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479*, 2021.

[88] Z. Liu, Z. Fan, Y. Wang, and P. S. Yu. Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer. In *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*, pages 1608–1612, 2021.

[89] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[90] H. Luo and L. Specia. From understanding to utilization: A survey on explainability for large language models. *arXiv preprint arXiv:2401.12874*, 2024.

[91] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[92] A. Madaan and A. Yazdanbakhsh. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*, 2022.

[93] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[94] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[95] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[96] A. Modarressi, M. Fayyaz, Y. Yaghoobzadeh, and M. T. Pilehvar. Globenc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. *arXiv preprint arXiv:2205.03286*, 2022.

[97] S. Mysore, A. McCallum, and H. Zamani. Large language model augmented narrative driven recommendations. *arXiv preprint arXiv:2306.02250*, 2023.

[98] J. Narwariya, P. Gupta, G. Gupta, L. Vig, and G. Shroff. X4sr: Post-hoc explanations for session-based recommendations. 2023.

[99] OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt, 2022.

[100] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[101] OpenAI. Gpt-4v(ision) system card, 2023.

[102] X. Peng, K. Wang, Z. Zhu, M. Wang, and Y. You. Crafting better contrastive views for siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16031–16040, 2022.

[103] A. Petrov and C. Macdonald. Effective and efficient training for sequential recommendation using recency sampling. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 81–91, 2022.

[104] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

[105] C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*, 2023.

[106] R. Qiu, Z. Huang, H. Yin, and Z. Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 813–823, 2022.

[107] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137, 2017.

[108] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[109] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[110] W. Ren, L. Wang, K. Liu, R. Guo, L. E. Peng, and Y. Fu. Mitigating popularity bias in recommendation with unbalanced interactions: A gradient perspective. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 438–447. IEEE, 2022.

[111] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[112] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644, 2011.

[113] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[114] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-Augu, pages 1135–1144, New York, NY, USA, 8 2016. ACM. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL https://dl.acm.org/doi/10.1145/2939672.2939778.

[115] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[116] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[117] O. Rubin, J. Herzig, and J. Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

[118] S. Serrano and N. A. Smith. Is Attention Interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Stroudsburg, PA, USA, 6 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1282. URL https://www.aclweb.org/anthology/P19-1282.

[119] P. Shi, R. Zhang, H. Bai, and J. Lin. Xricl: Cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-sql semantic parsing. *arXiv preprint arXiv:2210.13693*, 2022.

[120] D. Sileo, W. Vossen, and R. Raymaekers. Zero-shot recommendation as language modeling. In *European Conference on Information Retrieval*, pages 223–230. Springer, 2022.

[121] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[122] C. Singh, J. P. Inala, M. Galley, R. Caruana, and J. Gao. Rethinking interpretability in the era of large language models, 2024.

[123] W. Song, S. Wang, Y. Wang, and S. Wang. Next-item recommendations in short sessions. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 282–291, 2021.

[124] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[125] 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 *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[126] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[127] Q. Tan, J. Zhang, J. Yao, N. Liu, J. Zhou, H. Yang, and X. Hu. Sparse-interest network for sequential recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 598–606, 2021.

[128] S. Tan, R. Caruana, G. Hooker, and Y. Lou. Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 303–310, 2018.

[129] Y. K. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 17–22, 2016.

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

[131] N. Tintarev and J. Masthoff. Explaining recommendations: Design and evaluation. In *Recommender systems handbook*, pages 353–382. Springer, 2015.

[132] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[133] T. X. Tuan and T. M. Phuong. 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 138–146, 2017.

[134] B. Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 273–276, 2016.

[135] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[136] C. Wang, W. Ma, and C. Chen. Sequential recommendation with multiple contrast signals. *ACM Transactions on Information Systems (TOIS)*, 2022.

[137] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1101–1110, 2020.

[138] L. Wang and E.-P. Lim. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*, 2023.

[139] L. Wang, E.-P. Lim, Z. Liu, and T. Zhao. Explanation guided contrastive learning for sequential recommendation. *arXiv preprint arXiv:2209.01347*, 2022.

[140] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

[141] L. Wang, S. Zhang, Y. Wang, E.-P. Lim, and Y. Wang. Llm4vis: Explainable visualization recommendation using chatgpt. *arXiv preprint arXiv:2310.07652*, 2023.

[142] X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1543–1552, 2018.

[143] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5329–5336, 2019.

[144] Y. Wang, Z. Chu, X. Ouyang, S. Wang, H. Hao, Y. Shen, J. Gu, S. Xue, J. Y. Zhang, Q. Cui, et al. Enhancing recommender systems with large language model reasoning graphs. *arXiv preprint arXiv:2308.10835*, 2023.

[145] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[146] J. Wei, J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

[147] S. Wiegreffe and Y. Pinter. Attention is not not Explanation. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, 8 2019. doi: 10.18653/v1/D19-1002. URL https://www.aclweb.org/anthology/D19-1002.

[148] S. Wiegreffe and Y. Pinter. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

[149] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 726–735, 2021.

[150] 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. *arXiv preprint arXiv:2305.19860*, 2023.

[151] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 346–353, 2019.

[152] Y. Wu, R. Xie, Y. Zhu, F. Zhuang, X. Zhang, L. Lin, and Q. He. Personalized prompts for sequential recommendation. *arXiv preprint arXiv:2205.09666*, 2022.

[153] Z. Wu and D. C. Ong. On explaining your explanations of bert: An empirical study with sequence classification. *arXiv preprint arXiv:2101.00196*, 2021.

[154] Y. Xi, W. Liu, J. Lin, J. Zhu, B. Chen, R. Tang, W. Zhang, R. Zhang, and Y. Yu. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933*, 2023.

[155] Y. Xian, Z. Fu, H. Zhao, Y. Ge, X. Chen, Q. Huang, S. Geng, Z. Qin, G. De Melo, S. Muthukrishnan, et al. Cafe: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1645–1654, 2020.

[156] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, B. Ding, and B. Cui. Contrastive learning for sequential recommendation. *arXiv preprint arXiv:2010.14395*, 2020.

[157] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou. Graph contextualized self-attention network for session-based recommendation. In *IJCAI*, volume 19, pages 3940–3946, 2019.

[158] C. Xu, J. Feng, P. Zhao, F. Zhuang, D. Wang, Y. Liu, and V. S. Sheng. Long-and short-term self-attention network for sequential recommendation. *Neurocomputing*, 423:580–589, 2021.

[159] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[160] L. Xu, J. Zhang, B. Li, J. Wang, M. Cai, W. X. Zhao, and J.-R. Wen. Prompting large language models for recommender systems: A comprehensive framework and empirical analysis. *arXiv preprint arXiv:2401.04997*, 2024.

[161] Y. Xu, H. Chen, Z. Wang, J. Yin, Q. Shen, D. Wang, F. Huang, L. Lai, T. Zhuang, J. Ge, et al. Multi-factor sequential re-ranking with perception-aware diversification. *arXiv preprint arXiv:2305.12420*, 2023.

[162] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[163] X. Yang, X. He, X. Wang, Y. Ma, F. Feng, M. Wang, and T.-S. Chua. Interpretable fashion matching with rich attributes. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 775–784, 2019.

[164] Z. Yang, Z. Gan, J. Wang, X. Hu, Y. Lu, Z. Liu, and L. Wang. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3081–3089, 2022.

[165] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1): 1, 2023.

[166] T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang, et al. Self-supervised learning for large-scale item recommendations. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4321–4330, 2021.

[167] K. M. Yoo, J. Kim, H. J. Kim, H. Cho, H. Jo, S.-W. Lee, S.-g. Lee, and T. Kim. Ground-truth labels matter: A deeper look into input-label demonstrations. *arXiv preprint arXiv:2205.12685*, 2022.

[168] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[169] J. Yuan, Z. Song, M. Sun, X. Wang, and W. X. Zhao. Dual sparse attention network for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4635–4643, 2021.

[170] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[171] 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.

[172] S. Zhang, D. Yao, Z. Zhao, T.-S. Chua, and F. Wu. Counterfactual user sequence synthesis for sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 367–377, 2021.

[173] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. T. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022. URL https://doi.org/10.48550/arXiv.2205.01068.

[174] T. Zhang, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, D. Wang, G. Liu, and X. Zhou. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326, 2019.

[175] T. Zhang, X. Wang, D. Zhou, D. Schuurmans, and J. E. Gonzalez. Tempera: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[176] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 83–92, 2014.

[177] Y. Zhang, X. Chen, et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

[178] Y. Zhang, H. Ding, Z. Shui, Y. Ma, J. Zou, A. Deoras, and H. Wang. Language models as recommender systems: Evaluations and limitations. In *I (Still) Can't Believe It's Not Better! NeurIPS 2021 Workshop*, 2021.

[179] Y. Zhang, K. Zhou, and Z. Liu. What makes good examples for visual in-context learning? *arXiv preprint arXiv:2301.13670*, 2023.

[180] F. Zhao, T. Pang, Z. Wu, Z. Ma, S. Huang, and X. Dai. Dynamic demonstrations controller for in-context learning. *arXiv preprint arXiv:2310.00385*, 2023.

[181] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 2023.

[182] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, et al. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4653–4664, 2021.

[183] Y. Zheng, C. Gao, J. Chang, Y. Niu, Y. Song, D. Jin, and Y. Li. Disentangling long and short-term interests for recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 2256–2267, 2022.

[184] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1893–1902, 2020.

[185] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen. Filter-enhanced mlp is all you need for sequential recommendation. In *Proceedings of the ACM web conference 2022*, pages 2388–2399, 2022.

[186] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[187] Y. Zhu, Y. Xian, Z. Fu, G. de Melo, and Y. Zhang. Faithfully explainable recommendation via neural logic reasoning. *arXiv preprint arXiv:2104.07869*, 2021.