IMPACT OF TEMPORAL CONTEXT ON RECOMMENDER SYSTEMS ALONG GLOBAL TIMELINE

JI YITONG

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

18/02/2024

Date

JI YITONG

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

18/02/2024

.....

Date

.....

A/Prof. Sun Aixin

Authorship Attribution Statement

This thesis contains material from 3 papers published in the following peer-reviewed journal / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Yitong Ji, Aixin Sun, Jie Zhang and Chenliang Li. A Critical Study on Data Leakage in Recommender System Offline Evaluation. ACM Trans. Inf. Syst. 41, 3, Article 75 (July 2023), 27 pages.

The contributions of the co-authors are as follows:

- Prof. Sun suggested the research direction and the research problem.
- I conducted the experiment, analyzed the experimental results, and wrote the manuscript.
- Prof. Sun revised the manuscript and suggested areas of improvement.
- Prof. Sun, Prof. Zhang and I had been in continuous discussion to improve the ideas.
- Prof.Sun, Prof. Zhang and Prof. Li proofread the manuscript and provided feedback.

Chapter 4 is published as:

- Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20). Association for Computing Machinery, New York, NY, USA, 1749–1752.
- Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2022. Do Loyal Users Enjoy Better Recommendations? Understanding Recommender Accuracy from a Time Perspective. In Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '22). Association for Computing Machinery, New York, NY, USA, 92–97.

The contributions of the co-authors are as follows:

- Prof. Sun suggested the research direction and the research problem.
- I designed the experiment, conducted the experiment, analyzed the experimental results and wrote the manuscripts.
- Prof. Sun revised the manuscripts and suggested areas of improvement.

- Prof. Sun, Prof. Zhang and I had been in continuous discussion to improve the ideas.
- Prof.Sun, Prof. Zhang and Prof. Li proofread the manuscripts and provided feedback.

Chapter 5 is submitted as Yitong Ji, Aixin Sun, and Jie Zhang. Blending Old and New: Disentangled Incremental Learning for Graph-based Recommenders. Under review.

The contributions of the co-authors are as follows:

- Prof. Sun suggested the research direction.
- I camp up with the idea, designed and conducted the experiment, analyzed the experimental results and wrote the manuscripts.
- Prof. Sun revised the manuscripts and suggested areas of improvement.
- Prof. Sun, Prof. Zhang and I had been in continuous discussion to improve the ideas.

18/02/2024

Date

JI YITONG

Acknowledgements

I would like to extend my sincere gratitude and appreciation to my supervisor Prof. Sun Aixin, for his continuous guidance and support throughout my research journey. Prof. Sun always provides valuable insights and inspiring ideas during our discussions. He inspires me to perceive recommender system from a new perspective, and motivates critical thinking in research problems. Beyond his expertise, Prof. Sun is also a friendly and kind person. My PhD study has been interesting and fruitful because of him.

Furthermore, I would like to express my appreciation to Prof. Zhang Jie, for his continuous help during my research. Prof. Zhang provides valuable feedback and constructive comments during our discussions. Moreover, he provided opportunities for me to work in real-life recommendation project, which further broadens my knowledge in my research area.

Lastly, I would like to thank my parents, my husband Zhao Yue and my friends, Zhang Yixiang, Shi Jiachen and Xia Fan, my pet friends Pepper, Bun and Button for accompanying me throughout the years. Their encouragement and belief in my abilities have been a source of strength throughout my research journey.

Contents

List of Figures

List of Tables

Abstract

Recommender systems filter through the vast pool of information and provide personalized recommendations. However, with the dynamic nature of user preference, it is essential to design recommender systems that can adapt to the continuous changes in user preferences and the evolving environment.

In this thesis, we conduct a review of existing studies in recommendation models. More importantly, we perform a systematic analysis of training and evaluation protocols in recommender system research. Our analysis reveal that current setups often overlook the global timeline, leading to data leakage issues. To assess the impact of data leakage, we conduct carefully designed experiments where we gradually introduce increasing leaked data in training. The results show that data leakage results in unpredictable and inconsistent recommendation accuracy, which poses challenges in estimating a recommendation model's actual performance. Hence, we underscore the importance of following the global timeline in both training and evaluation stages of recommendation models.

Furthermore, we demonstrate that the ignorance of the global timeline and data leakage hinders recommender systems from accurately modeling the temporal context. We specifically investigate this point using "popularity" of items. It is showed that failure to capture the accurate temporal context results in less accurate recommendations. While the importance of adhering to the global timeline is emphasized, the methods for incorporating temporal context into recommender systems remain unclear. To address this gap, we conduct experiments to explore the relationship between user interactions and temporal context. The experimental results provide insights into which interactions and how many interactions should be included in the training set to improve recommendation performance within specific temporal contexts. Through extensive analysis, we find that recent interactions, which are more relevant to the target time context, should be prioritized. These findings offer guidance on effectively integrating temporal context into recommender systems to enhance recommendation accuracy in specific time contexts.

From the recent interactions, we learn a user's latest preference. However, relying solely on recent interactions for training may lead to overfitting, causing the recommendation model to overlook long-term preferences that are essential for accurate recommendations. To address this, we propose the use of incremental learning techniques to retain both the short-term and long-term preferences of users. Specifically, we introduce an incremental learning framework that retrains the GCN-based model with the disentanglement of the two types of preferences. This approach ensures effective recommendations.

In summary, despite existing research efforts in the field of recommender systems, we contend that there is a lack of study from the perspective of the global timeline. In this thesis, we emphasize the importance of following the global timeline to avoid data leakage issues and effectively model temporal dynamics over time. Furthermore, we propose a retraining framework that not only rigorously considers the global timeline during training and learns user preferences from the most relevant interactions but also retains the long-term characteristics of users to enhance recommendation performance. Finally, we discuss potential areas for future research in the concluding chapter.

Chapter 1

Introduction

With the continuous growth of the internet, users are exposed to an overwhelming amount of information when using web services. The constant production of vast quantities of data poses a significant challenge for users in their decision-making process, as they struggle to access the items or content that align with their preferences. This phenomenon is commonly referred to as "information overload" [1, 2]. In response to this challenge, many web service platforms have implemented recommender systems to expediate the search and selection process for users. For example, the popular e-commerce platform Amazon [3] utilizes a recommender system to suggest items to users. As illustrated in Figure 1.1, Amazon's "Recommended for you" feature presents users with a curated selection of eight items out of the millions available on their platform. This personalized recommendation aims to assist users in narrowing down their choices and finding items that align with their preferences. Similarly, video-sharing platforms YouTube [4] and the Google Play store [5] leverage recommendation systems to assist users in discovering videos and apps that align with their preferences.

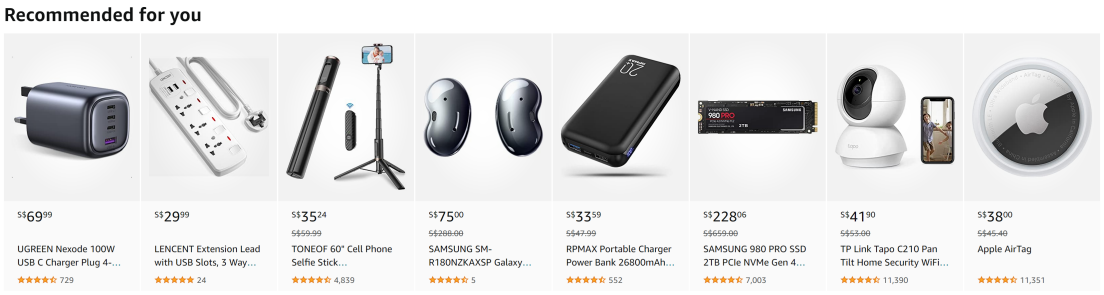


FIGURE 1.1: A snapshot of the "Recommended for you" section on Amazon.com.

1.1 Motivation and Objectives

Recent years have witnessed increasing efforts in the research of recommender system [6–9]. Various algorithms have been proposed. For instance, sequential recommendation [10], session-based recommendation [11] and graph-based recommendation [12] have been widely studied to solve different research problems in recommender system. With the large number of research paper publishes every year, one interesting question to ask is: *"what is the progress in the development of recommender system? To what extent have the research problems been solved?"* Dacrema et al. [6] have the same inquiry on the progress of research. Hence, they conduct experiments on various datasets to investigate the accuracy of various recommendation algorithms, ranging from the simplest popularity-based recommender to the more complex neural recommender like NeuMF [13] and Multi-VAE [14]. Through thorough experiments, the authors make two observations. Firstly, only less than half of the sampled research papers' results can be reproduced. Secondly, it is not necessary that more complex neural recommendation models outperform simpler algorithms. The experimental results answer the question on the degree of progress to some extent. In particular, whether recommendation models have advanced in terms of recommendation accuracy is questionable. A similar observation is made in a recent study [7].

The authors of these papers suggest that the observations could be attributed to the inconsistent evaluation protocols adopted in research papers. In this thesis, we aim to explain the observations from the global timeline perspective, which is an important aspect but has not been well explored.

Research Objective 1: Understand global timeline in evaluation stage

Conduct empirical study to study the impact of data leakage in offline evaluation of recommender system, arising from the ignorance of global timeline in evaluation.

To achieve this objective, we analyse the data splitting strategies in offline evaluation, which have garnered considerable interests from other researchers [15–22]. What is different, we show that data splitting strategies that do not follow the global timeline leads to a data leakage issue. That is, a recommendation model is trained with future data to predict historical data instances. We then design experiments to investigate the impact of data leakage, by varying the severity of data leakage in a controlled experiment setting.

Our experiments provide the following findings:

- The evaluation results demonstrate a consistent pattern of recommending "future items" to past test instances. "Future items" refer to items that have not yet been available in the system at the time of the test instance. It is an invalid recommendation, because it will never lead to a hit, making the recommendation results biased.
- Incorporating "future training instances" in the training set causes a shift in the distribution of user-item interactions. This alteration results in significant differences between the recommendation lists obtained from experiments that include future data in training and those without data leakage in training.
- Data leakage has a notable impact on the accuracy of recommender models, and its effects are unpredictable.
- The incorporation of different quantities of "future data" in our experiments leads to inconsistent ranking orders of the baseline models. The effectiveness of baseline models cannot be easily interpreted.

These findings show that data leakage, arising from ignorance of the global timeline, makes the experiment results meaningless. The interpretation of results would not lead to a concrete conclusion in whether a recommendation model is performing effectively. To some extent, we explain the irreproducibility of recommendation results due to the inconsistent evaluation protocols.

In light of these findings, we propose to conduct offline evaluation using "time point separation scheme" which is to ensure all training instances happen before the test instances.

Research Objective 2: Understand global timeline in training stage

Conduct empirical study to study the temporal context along the global timeline.

The findings in the experiments of research objective 1 suggest that ignorance of the global timeline in the evaluation stage leads to invalid recommendation results. We further extend the analysis to the training stage, and it is found that temporal context can be better captured if a global timeline is observed when training a recommendation model.

In many existing research papers, the entire training set is fed to the recommendation model without considering the temporal context along the global timeline. In this thesis, we conduct empirical study to understand the user behaviours with respect to the temporal context along the global timeline. We show that knowing all interactions of a user does not lead to a better understanding of his/her interests in the current temporal context. What matters more in recommendation is the recent interactions by a user.

Research Objective 3: Modelling dynamic temporal context

Design a retraining framework to dynamically update a graph-based recommendation model with the recent interactions.

In the real life scenario, a recommender system is dynamic with new interactions being observed continuously. Moreover, the experiments in achieving research objective 2 have shown that recent interactions carry the most updated information about a user's preference. Hence, retraining of a recommendation model with the newly observed interactions is essential to keep the recommendation model updated with the most recent user preference. In this thesis, we design a retraining framework for graph-based recommendation model. In particular, the retraining framework enables the learning from the new interactions without forgetting the long-term preference of users. It is achieved with two modules: an *Information Extraction Module* to extract long-term information and an *Disentanglement Module* to distinguish new and old information for effective recommendation. Notably, the

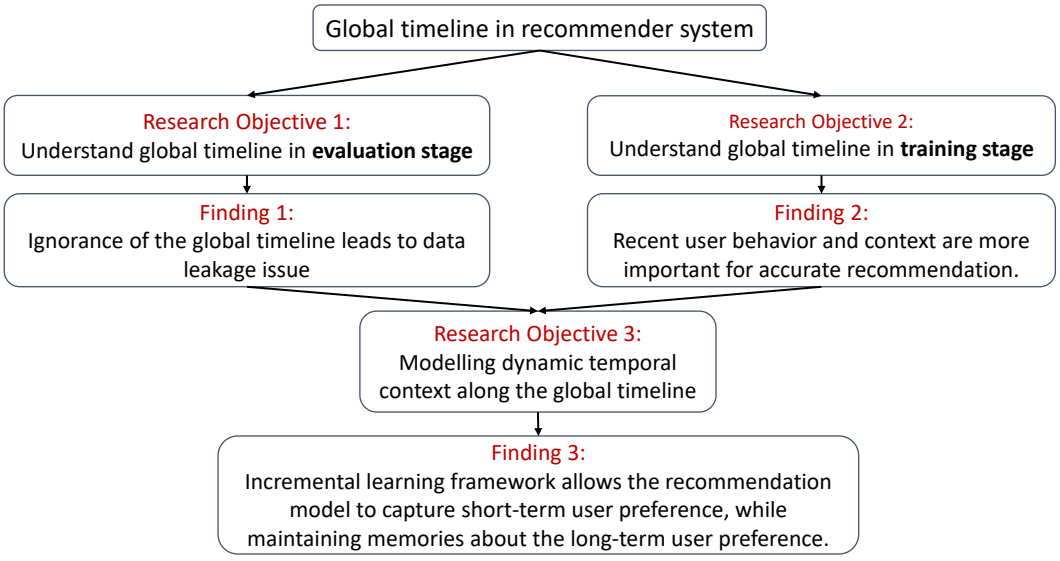


FIGURE 1.2: Thesis Overview

retraining framework strictly follows the global timeline, thus avoiding the data leakage issue mentioned in research objective 1.

To summarize, we use Figure 1.2 to illustrate the overview of this thesis. This research highlights the importance of following a global timeline in recommender system. The fundamental concept involves capturing the evolving context across the global timeline. This approach parallels the notion of "continual learning" [23, 24], where the system learns from ongoing streams of data in addition to existing knowledge.

1.2 Major Contributions

Overall, the research contributions in this dissertation can be summarized into three points:

- We examine the commonly adopted offline evaluation protocols in recommender system from a global timeline perspective. We show that a global timeline is not followed, when certain train-test splitting strategies (*e.g.*,,

leave-one-out split) are used in offline evaluation of recommender system. As a consequence, data leakage can be observed in offline evaluation, making the performance of recommendation models not accurately reflecting their actual performance in an online platform.

- Along the global timeline, temporal context has been continuously changing due to new trends in the market. In this thesis, we illustrate the temporal dynamics with popularity of items and show the necessity in modelling the correct temporal context. We further conduct experiment to investigate how user interactions shall be considered to model the correct temporal context. The findings suggest that recent user behavior and context may be more influential in generating accurate recommendations than the user's overall activity history.

- We propose an incremental learning framework, Disentangled Incremental Learning (DIL), for GCN-based recommender systems. It is designed to ensure that the new information reflected in the new user-item interactions can be well captured. Meanwhile, certain old but relevant information in the historical user-item interactions will not be forgotten. These can be achieved by an extraction module which extracts relevant old information, and a disentanglement module which ensures that both old information and new information can be well learned. DIL contributes to retrain a recommendation system to adapt to changes in user behaviors over time.

1.3 Outline of the Thesis

In the remaining chapters, we describe our research contributions in details. Before that, Chapter 2 gives a review of the relevant works that consider temporal factors in recommender system. Specifically, we review time-aware recommendation model, sequence-aware recommendation and other models that handle dynamics in recommender system, *i.e.,*, dynamic representation learning model and incremental learning model. Moreover, we conduct survey on works that study the evaluation process of recommender system. Following that, Chapter 3 demonstrates that ignorance of the global timeline and temporal context leads to data leakage problem. The impact of this issue is then studied via experiment. Chapter 4 analyzes how

user interactions shall be considered to model the temporal contexts. Chapter 5 then proposes an incremental learning framework to update a recommendation model with the most recent interactions. Moreover, relevant historical information is retained to ensure that persistent user preference is not forgotten. Lastly, we conclude the thesis and present some potential future directions of research in Chapter 6.

Chapter 2

Literature Review

In this chapter, we review existing recommendation methods. We start this chapter by presenting background information of recommender system in Section 2.1. We show that recommender system can be divided into content-based recommendation, collaborative filtering based recommendation and the hybrid of the two. In this thesis, we focus on collaborative filtering based recommendation. Furthermore, we review context-aware recommender system which takes into account different contextual information, *e.g.*,, spatial and temporal information. In Section 2.3, we focus on the temporal information and discuss existing methods that consider temporal factors. That is, we provide literature survey on:

- Time-aware recommendation model which considers timestamp at which user-item interactions take place.
- Sequential recommendation model which considers the temporal order of user-item interactions.
- Dynamic representation learning model that updates user representations and item representations when new interaction is observed.
- Incremental learning recommendation model which adaptively refreshes understanding of user preference reflected in the new batch of user-item interactions.

2.1 Background

According to [8, 25], a recommender system is defined to be a tool/software which provides suggestions of items to users, to improve user experience. In particular, a recommender system learns user preference from the data instances it records and recommends a user with the items that may be of his/her interests. The items recommended constitute a significantly smaller set compared to the entire set of the available items in the system. The data instances that a system can record and learn from are in the form of ⟨**user, user attributes, item, item attributes, timestamp of interaction**⟩ tuples. Recent years, recommender systems are designed with different techniques to learn from data instances. Specifically, recommender systems can be divided into content-based recommender system, collaborative filtering recommender system, and the hybrid of the two [26, 27]. We describe the three types of recommender system in the coming sections. However, the focus of this thesis is on collaborative filtering recommender system.

2.1.1 Content-based Recommender System

Content-based recommender system is widely adopted in the recommendation of news articles [28–30]. It generally considers user, user attributes, item and item attributes as inputs [25, 31]. Specifically, a user profile is described by attributes of his/her historically interacted items. Then, recommendation is made by matching the user profile with the contents of items in the candidate set. Hence, content-based recommender system involves three steps: (i) *item content learning*; (ii) *user profile learning* and (iii) user profile and item content matching.

For item content learning, one widely developed model is keyword-based vector space model. The key idea is to leverage a d-dimensional vector to represent the textual content of an item. Each dimension of the vector is linked to a keyword. The d-dimensional vector denotes the weights of keywords in an item's content. One commonly used method to learn weights of keywords is *TF-IDF* (Term Frequency-Inverse Document Frequency) [32]. Given N documents $D = \{d_1, d_2, ..., d_N\}$ and a list of keywords/terms $T = \{t_1, t_2, ..., t_d\}$, the term frequency (TF) quantifies the frequency of keywords in one document. Hence it can be defined as:

$$TF\left(t_k, d_j\right) = \frac{f_{k,j}}{max_z f_{z,j}} \tag{2.1}$$

where $f_{k,j}$ denotes the frequency of a term t_k in document d_j. $max_z f_{z,j}$ is the maximum frequency among all the terms. Then, we have inverse document frequency (IDF) which quantifies the frequency of a keyword across all documents:

$$IDF\left(t_k\right) = log\frac{N}{n_k} \tag{2.2}$$

where N is the number of documents and n_k is the number of documents with the keyword t_k in them. TF-IDF can be derived using equation 2.3 to represent the weights of a keyword in a document.

$$TF - IDF\left(t_k, d_j\right) = TF\left(t_k, d_j\right) \cdot IDF\left(t_k\right) \tag{2.3}$$

From there, the d-dimensional vector of a document can be obtained, with each dimension representing the weight of each keyword in the document. Using TF-IDF, the bias which frequent terms dominate the modelling process is mitigated. Given the vectors of documents (items), a user profile can be constructed based on the items he/she interacted with before. Recommendation of items can be made by matching the user profile with the candidate items via similarity measure, *e.g.,* cosine similarity.

2.1.2 Collaborative Filtering Recommender System

Collaborative filtering (CF) recommender system works under the assumption that users who rate/buy/click similar items in the past will rate/buy/click similar items in the future [25, 33, 34].

The simplest CF model is neighborhood-based recommendation model, *i.e.,* user-based recommender system and item-based recommender system. For the former, a user is recommended with items rated by his/her corresponding "similar users". For the latter, a user is recommended with items similar to the items rated by him/her in the past. Neighborhood-based recommender system is easy to implement. The

key idea pertains to the computation of the pairwise similarity between two users or the computation of the pairwise similarity between two items [3, 35]. Taking user-based recommender system as an example, similarity between user i and user j can be computed from their rating vectors $\mathbf{u_i}$ and $\mathbf{u_j}$ by cosine similarity:

$$sim\left(\mathbf{u_i}, \mathbf{u_j}\right) = \frac{\mathbf{u_i} \cdot \mathbf{u_j}}{\|\mathbf{u_i}\|\|\mathbf{u_j}\|} \tag{2.4}$$

Or, similarity can be measured by Pearson Correlation [36].

$$p\left(i, j\right) = \frac{\sum_{\forall v \in V_{i,j}} \left(u_{i,v} - \bar{u}_i\right) \cdot \left(u_{j,v} - \bar{u}_j\right)}{\sqrt{\sum_{\forall v \in V_{i,j}} \left(u_{i,v} - \bar{u}_i\right)^2} \sqrt{\sum_{\forall v \in V_{i,j}} \left(u_{j,v} - \bar{u}_j\right)^2}} \tag{2.5}$$

where $V_{i,j}$ denotes the items rated by both user i and user j, namely co-rated items in this thesis. $\bar{u}_i$ and $\bar{u}_j$ are the average ratings on the cor-rated items by user i and the average ratings on the co-rated items by user j respectively.

Another type of CF recommender system is model-based recommender. It is to build a predictive model which anticipates the preference score of a user on an item. In general, a CF model can be denoted as $f(p_u, q_i) \rightarrow R_{ui}$, from which p_u denotes the user-related parameters for user u while q_i denotes the item-related parameters for item i. $f(\cdot)$ is the model which maps the user parameter p_u and the item parameter q_i to predict user u's preference score R_{ui} on item i. User parameters, item parameters and model parameters in $f(\cdot)$ are to be learned from historical user-item interactions.

One conventional model-based CF is Matrix Factorization (MF) model [37–41]. The general idea of an MF model is to learn low-dimensional latent vectors for both users and items. These vectors can be deemed as compressed representations that encode essential attributes of users and items. The widely adopted formulation of MF model is:

$$f_{MF}(\mathbf{p_u}, \mathbf{q_i}) = \mathbf{p_u}^T \mathbf{q_i} \rightarrow r_{ui} \tag{2.6}$$

where $\mathbf{p_u} \in \mathbb{R}^k$ is the k-dimensional latent vector for user u while $\mathbf{q_i} \in \mathbb{R}^k$ is the latent vector for item i. Unlike MF models, Factorization Machine (FM) [42] not

only learns relationships between users and items, it can also learn from other variables like time. Moreover, FM models second-order relationship between variables. That is, a feature vector $\mathbf{x} \in \mathbb{R}^n$ can be constructed from variables, *e.g.,* user, item and time. Then, FM models both first-order and second-order interactions by:

$$f_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\mathbf{v_i} \cdot \mathbf{v_j})\, x_i x_j \tag{2.7}$$

where $w_0 \in \mathbb{R}$ is a bias term, $\mathbf{w} \in \mathbb{R}^n$ reflects the strength of variables, $\mathbf{v_i}$ is a k-dimensional vector that describes the i^{th} variable. $(\mathbf{v_i} \cdot \mathbf{v_j})$ models the relationship between the i^{th} and the j^{th} variables. $(\cdot)$ denotes the dot product operator. The aforementioned general recommender systems lack the ability to effectively capture sequential behaviors exhibited by users. To address this limitation, sequential recommender systems have been extensively developed to model a user's behavior based on his/her historical behaviors. FPMC [43] introduces a combination of Matrix Factorization and Markov chains to capture both sequential patterns in user behaviors and long-term user preferences. Specifically, FPMC constructs personalized transition matrices using individual user behaviors. These transition matrices are then organized into a transition cube, which can be further factorized to derive latent vectors representing users and items for the purpose of recommendation.

In recent years, deep learning model based recommender system has become prevalent [6, 8, 44]. NeuMF [13], as one of the earliest deep learning recommender systems, combines Matrix Factorization (MF) and Multi-Layer Perceptron (MLP) to model more complex user-item relationships. DeepFM [45] then integrates FM model and deep neural network model to model the higher-order complex interactions between features. Other recommendation models do not fuse conventional models (*e.g.,* MF and FM) with deep neural network, but purely using deep neural model architecture to learn user interests. For example, Graph Neural Network (GNN) has been widely adopted in recommender system [12, 46, 47]. That is, a user-item interaction graph can be constructed based on observed user-item interactions. A graph can be represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denotes the set of nodes and set of edges respectively. Nodes can be either user or item nodes. If there is interaction by a user on an item, an edge would exist between the corresponding user node and item node. GNN generally adopts multiple information propagation layers to explicitly model indirect relationships between users and

items. Information propagates through edges in the user-item interaction graph. Existing GNN-based recommendation models include PinSage [48], LightGCN [49], NGCF [50] and etc. Other variants of GNN could extend beyond using user-item interaction graph. Instead, user-user and item-item relations are modelled to provide additional information for recommendation [51–53].

2.1.3 Hybrid recommender system

The collaborative filtering recommender system is highly effective in providing recommendations because it captures the intricate relationships between users and items. Its effectiveness stems from its ability to leverage historical user-item interaction data to learn user preferences accurately. However, the collaborative filtering approach encounters difficulties in accurately capturing the characteristics and attributes of new items and new users which do not have any interaction records. This challenge is commonly known as the "cold-start problem" [54, 55]. In contrast, content-based recommender systems can mitigate the cold-start problem by incorporating item descriptions into the learning process. Hence, the combination of both types of recommenders holds promise in improving the quality of recommendations [56]. Through combination, the models obtained are named "hybrid recommender system".

A commonly used technique for combining collaborative filtering and content-based recommender systems is weighted combination. In this approach, weights are assigned to the output scores generated by both systems [57]. These weights determine the relative importance of each system's score. The final score of an item is then computed as the weighted sum of the scores. This combined score is used to determine the item's suitability for recommendation. The authors of [58] further extend the weighted techniques to user-to-user, user-to-tag and user-to-item relations in social media. The aggregated scores are the predicted preference score of a user on an item. Apart from the weighting techniques, techniques that use textual content as side information in the collaborative filtering recommender system have garnered research interests. For example, the authors in [59] propose to incorporate side information as input in each layer of an autoencoder network to alleviate the cold-start issue. In [60], the authors propose to jointly learn user latent vectors and item latent vectors from their attributes and the user-item rating matrix at

the same time. Recommendation performance can be boosted with the additional information provided by the side information.

2.2 Context-aware recommender system

A context-aware recommender system is designed to consider contextual information such as spatial information, social information and temporal information [61].

Contextual information, such as spatial details, is extensively integrated into location-based recommender systems [61, 62]. This information plays a pivotal role in accurately determining a user's specific physical location at the time of making recommendations. In the paper [63], user-generated location-based rating data is combined with location features like categories and tags within the recommender system framework for effective recommendation. This integration addresses challenges associated with data sparsity. Additionally, the GeoMF model, outlined in [64], incorporates users' frequently visited places and items' related regions into a matrix factorization-based recommender system.

Social relationships among users also contribute valuable contextual information to recommender systems. In [63], the authors introduce social regularization terms for matrix factorization recommender systems, aiming to enhance the similarity between latent factors of users who are closely connected as friends. As deep-learning neural networks evolve, social relationships can be effectively represented using graphs. The authors of [65, 66] model the user-user social graph using graph neural networks. By considering both the user-item interaction graph and the user-user social graph, recommender systems can leverage the advantages of contextual social relationships.

Another crucial contextual element is the temporal factor. In the construction of a preference prediction model, a recommender system that is attuned to time takes into account the timing of a user's past interactions with the system. This temporal context information empowers a time-aware recommendation model to provide unique suggestions at various moments. The temporal factors to be considered may involve the absolute timestamp in a global time format or repeatable temporal details, such as the day and week. In the upcoming section, we will delve into the recommendation models that take temporal factors into account.

2.3 Recommendation Model considering Temporal Factors

Recommendation models that consider temporal factors can be divided into two categories: time-aware recommendation model and sequential recommendation model. In particular, a time-aware recommendation model takes into account the time at which a user interacts with a system when modeling their preferences. A sequential recommendation model considers the order in which a user interacts with items to model long-term dependencies in user behaviours.

2.3.1 Time-aware Recommender System

When constructing a preference prediction model, a time-aware recommender system factors in the timing of a user's historical interactions with the system. By leveraging this temporal context information, a time-aware recommendation model may provide differentiated recommendations at different timings.

One of the earliest studies that proposes to consider recommendation with temporal context information is [67]. The temporal context can be formulated as *morning of a day* or *day of a week*. Similar designs have been proposed in [68]. By fixing the temporal context when making recommendation, the authors can reduce the problem to a 2-dimensional problem of "what items should be recommended to a user at a specific time?" In [69], the authors further highlight that time is associated to concept drift in recommender system. For example, item popularity and user tastes could change constantly over time. To model the changes, the authors propose a model called timeSVD++ which is built on top of Matrix Factorization model. The key idea is to model users and items with different representations depending on the temporal context. While modelling time-dependent representations, seasonality and continuity are considered. Similarly, in [70], the authors propose to consider time-dependent user bias on ratings and also time-dependent item popularity bias when making recommendation using Matrix Factorization model. When it comes to deep learning model, TASER [71] models both absolute time of interactions and relative time between interactions in the encoder. Absolute time and relative time are also considered in [72]. In [73], the authors design

a temporal kernel to differentiate the impact by past events that happened in different time. Multiple kernel functions have been proposed to capture the various temporal dynamics. For example, an exponential decay kernel assumes that impact of past events decay exponentially as the time interval to the recommendation time increases. Meanwhile, a constant kernel function assumes that the impact is stable throughout the timeline. Unlike [72], [73] and [71] which focus on timestamp of an event, [74] and [75] consider time of an event as contextual information. That is, the authors are more concerned with contextual features like *year*, *month* and *week*. In [74], these contextual features are leveraged to construct time-dependent relations between user nodes and item nodes in a knowledge graph. Here, a relation exists when a user rates/clicks/buys an item, it is time-dependent because it characterises the temporal context at which the interaction takes place. As the relation defined in [74] reflects contextual information of an interaction, effectiveness and explainability of the recommender system is enhanced. Similarly in [75], contextual features are leveraged for recommendation, but in a music-listening scenario.

2.3.2 Sequence-aware Recommender System

Given a sequence of historical user-item interactions, a sequential recommender system aims to predict the next item that the user will likely interact with [10, 76, 77]. The key idea behind sequence-aware recommender system is to capture the sequential patterns and long-term dependencies between items in user interaction sequences.

One line of work is to retrieve frequent sequential patterns in user behaviours for next-item recommendation. In [78, 79], the authors propose recommendation engines which recommend the next item by matching users' recent n interactions with the frequent item sets mined using Association Rule, Sequential Pattern or Contiguous Sequential Pattern techniques. The aforementioned methods focus on explicit frequent patterns that can be obtained by pattern mining algorithms. The authors in [80], instead, propose IRRMiner to detect implicit/hidden patterns which contain items that are indirectly but closely related, for efficient and effective recommendation. To further enhance the recommendation performance, the authors propose a personalized sequential pattern mining algorithm in [81]. Specifically, the authors introduce a competence score method to compute the relevance

scores of the mined frequent sequences to a target user. The relevance score of a frequent sequence varies for different users. Using the relevance score, one can further calculate the personalized support value of each frequent pattern for each user and then make recommendations.

Dependencies over a user interaction sequence can be modelled by many other model-based techniques. FPMC [43] combines Matrix Factorization (MF) and Markov Chains (MC) models to model sequential behaviours. In FPMC, factorization is conducted on a transition cube which is obtained by stacking every user's transition matrix. A user's transition matrix is obtained from MC over actions of him/her. The authors of [82] point out that FPMC could suffer from sparsity issue. To alleviate the issue, they propose FOSSIL - a recommendation model which combines similarity-based methods and MC to model sequential behaviours. Furthermore, unlike FPMC which models the first-order Markov Chains, FOSSIL works with the higher-order Markov Chains.

Recent years have witnessed the increasing researches in deep-learning based sequence modelling techniques [77]. One popular line of work to model sequential data is on using Recurrent Neural network (RNN). In [83], GRU4Rec is proposed to adopt Gated Recurrent Unit (GRU) to model sequential data in user-item interaction sessions. To alleviate the vanishing gradient issue in GRU4Rec, the authors of [84] propose GRU4Rec+ which incorporates a novel ranking loss in training. That is, instead of having the comparison of the target score with random samples in the ranking loss, GRU4Rec+ compares the target score with the sample possessing the highest score. GRU network is also leveraged in NARM [85] to encode a user's sequential behaviours in a session. On top of the GRU encoder, NARM further incorporates an attentive mechanism to capture the purpose of a user, based on his/her latest behaviour in a session. Different from NARM and GRU4Rec, HRNN [86] not only considers a user's behaviours in a session, it also models cross-session behaviours of a user using GRU to capture the user's evolving interests over time. Furthermore, GRU network is also used in DIEN [87] as the interest extractor layer to capture dependencies between behaviours. The GRU network in DIEN is designed with an attentional update gate to assign different weights to items based on their relevance to the target item. In [88], RNN is used to capture the short-term interest of a user. The authors then differentiate long-term interest and short-term interest of a user.

Sequential dependencies could also be modelled using attention-based network. One most representative study is SASRec [89] which is built on top of a self-attention network to capture the long-term dependencies in a user interaction sequence. The authors of [90] further propose TiSASRec which improves SASRec by incorporating time interval between interactions of a user. In [91], the authors pinpoint two issues in self-attention based recommender systems like SASRec and TiSASRec. The first issue is that embeddings in SASRec and TiSASRec fail to model uncertainty in user dynamic interests. The second issue is that collaborative transitivity for indirectly related items (collaboratively similar items) cannot be well modelled. To handle these issues, the authors modify SASRec into STOSA which is designed with stochastic embeddings, Wasserstein self-attention layer and a new regularization term on ranking loss. Other variants of SASRec include consideration of mutli-behaviours of users [92] and denoising of interaction sequences [93]. Most of the self-attention based recommenders are unidirectional models, because they are designed with causality operation to ensure that only interactions before time t will be considered when predicting the interaction happened at time t. The authors of [94] highlights that user interactions order may not be rigid. Hence, they propose a bidirectional model, BERT4Rec, to model an interaction's context from its neighbors in both the past and the future.

Graph neural network (GNN) methods for sequential recommendation tasks have received significant research attention in recent years. The rationale is that these methods enable the model to capture implicit collaborative signals beyond what traditional approaches can capture. The SR-GNN [95] method utilizes a GNN model to capture intricate item transitions in a user's interaction session. The user's behaviors during a session within a short time frame is represented as a directed graph consisting of item-item interactions. GNN is then employed to generate item embeddings for recommendation by learning from the user's session graphs. MA-GNN [96] follows a similar approach to SR-GNN, using a graph neural network to capture a user's short-term interests based on their recent interaction sequence. The GNN is utilized to learn from the user's interaction sequence and generate a representation of his/her interests. Unlike SR-GNN and MA-GNN which concern item-item relationship in a session, the authors in [97] proposes to consider item-item relationship in a global graph. Additionally, in [97], a semantic item graph is introduced to represent item relationships based on their attributes. The inclusion

of a semantic item graph enhances the recommendation process by providing additional knowledge about items. The methods mentioned above adopt one of the two approaches for learning item embeddings: either by relying on session graphs or by using global graphs. In contrast, GCE-GNN [98] takes into account both types of graphs and generates two levels of item embeddings for recommendations. Similarly, in [99], the inter-session graph is utilized to model explicit item relationships, while the global graph is utilized to model implicit item relationships.

2.4 Handling Temporal Dynamics in Recommender System

As user-item interactions happen continuously in an online platform, new trends including new user preference and new item market positions may exhibit in new stream of interaction data. To capture the changes reflected in the new user-item interactions, research efforts have been devoted. One line of work is to **retrain a recommendation model** with incremental learning techniques. The other line of work is on **dynamic representation learning** which is to update user representation and item representation upon seeing a new interaction. The latter is completely different from the former. The key differences are two-folded. Firstly, the updates of model parameters are conducted with different frequency. Dynamic representation learning refreshes user representation and item representation upon seeing one new interaction, while incremental learning is only conducted when a good amount of interactions is accumulated. Second, dynamic representation learning only updates user representation and item representation with new interaction, incremental learning often involves updating of an entire recommendation model using the new stream of interactions observed. There could be other model parameters on top of the user representations and the item representations.

As "continual learning", also termed as incremental learning, is relevant to this thesis, we will first study the foundational techniques associated with continual learning in the realm of deep learning. Following this, we will expound upon the mechanisms of incremental learning within recommender systems. Subsequently, our discourse will pivot to a detailed examination of dynamic representation learning in the context of recommender systems.

2.4.1 Continual Learning

Continual learning, alternatively referred to as lifelong learning or incremental learning, concerns the ability of a machine learning model to acquire new information over time while maintaining previously learned knowledge. Within the realm of deep learning, continual learning specifically tackles the challenge of retaining past experiences in the face of evolving data distributions or shifting tasks [23, 100, 101].

Continual leaarning setting is categorized into three types based on the nature of the newly encountered data. When the observed new data pertains entirely to a novel task, the application of task incremental learning techniques becomes pertinent. These techniques facilitate the assimilation of essential knowledge for each task, both old and new. Moreover, when the newly observed data relates to the same task as a previous one but within disparate domains, it is classified as a domain incremental learning task. Lastly, when the new data introduces a distinct class not present among the existing classes, the utilization of class incremental learning techniques becomes imperative for the recognition of the additional class.

In the context of retaining knowledge in a task incremental setting, various techniques have been introduced. One such approach, presented in [102], is Elastic Weight Consolidation (EWC), which incorporates a regularization term into the loss function. This regularization term penalizes alterations to weights that are deemed important for prior tasks. Differing from EWC, which is a regularization-based method, Hyper-CL employs the parameter isolation technique, as outlined in [103]. Specifically, the authors introduce a metamodel to function as weight generators, producing weights for task models corresponding to different tasks.

In the context of domain incremental learning tasks, incremental data often presents itself as a shifted data distribution. An illustrative example includes variations in weather conditions, considered a form of distribution shift [104]. To ensure robust object detection under diverse weather conditions, the authors propose statistical correction method to retain weather-specific first and second-order statistics. This approach aims to prevent the forgetting of previously learned information.

Regarding class incremental learning, the work [105] classifies learning algorithms

into three categories: data-centric, model-centric, and algorithm-centric class incremental learning algorithms. Data-centric algorithms commonly involve replaying old data instances during training with new instances to mitigate forgetting [106, 107]. Model-centric algorithms seek to expand the model's capacity to handle new classes or apply regularization to parameters crucial for memorizing old classes [108, 109]. In algorithm-centric incremental learning, knowledge distillation is a prevalent technique that penalizes changes in the model when transferring knowledge from a teacher model to a student model

In the realm of recommender systems, incremental learning shares similarities with continual learning, wherein new data, including new user-item interactions, is observed over time. Various techniques have been proposed to reduce forgetting of long-term user preferences reflected in old interactions when training a recommender system with new user-item interactions. We detail the techniques in the following section.

2.4.2 Incremental Learning-based Recommender System

Incremental learning, in recommender system, learns from streams of interactions. In other words, the update of a recommendation model is often activated after receiving a good number of user-item interactions. One straightaway to capture the changes reflected in these new interactions is to finetune the recommendation model with the recently observed data instances. By "fine-tune", it means updating the existing model parameters with the new interactions. Although fine-tune ensures that the model is trained with the most recent interactions, it could potentially be problematic due to the potential "catastrophic forgetting" issue [110]. That means, the memory about long-term user preference and persistent item characteristics may be forgotten, when they are not reflected in the new interactions. To well memorize both long-term information and short-term information, a recommendation model could also be retrained by using all historical and newly observed interactions [111]. At a first glance, conducting "full-retrain" is effective in modelling both historical information and new information. However, "full-retrain" is a non-scalable solution as the number of interactions is continuously increasing. Moreover, "full-retrain" could incorporate old interactions that are outdated for modelling a user's interests.

Based on the discussions, efforts have been delved into the research of the incremental learning techniques which reap the benefits of fine-tune and full-retrain [112, 113]. In particular, incremental learning allows a recommendation model to update to adapt new information reflected in the new user-item interactions. At the same time, incremental learning is usually equipped with scalable techniques to avoid forgetting historical knowledge learned from old user-item interactions. Generally, incremental learning for recommender system can be categorized into three types: experience replay, model-based incremental learning, and knowledge distillation.

Experience Replay. To preserve historical information, a subset of historical data instances are selected for replay. These instances are considered representative instances of historical data. Instances selected for replay and newly received instances are then used to retrain the model. Note that, the number of instances selected is fixed to ensure scalable training of a recommendation model. Historical instances can be selected by random sampling [114] for keeping a "sketch" of data distribution. In [115], more recent data instances are selected to the reservoir. Contrary to [115], studies like [116–118] tend to overlook more recent user-item interactions but keep older data instances in the reservoir of historical interactions. The key idea of keeping older data instances is to maintain a long-term memory. During the retraining of a recommendation model, the more informative interactions in the reservoir are then sampled from the reservoir as training instances. For example, SPMF [116] and SSRM [117] consider the ground truth interactions with less predicted preference score the informative interactions. GAG [118] measures the *Wasserstein distance* between the predicted recommendation and the real interaction. The larger the distance, the less accurately the recommendation is, thus the more training is required on the interactions.

Model-based Incremental Learning. This line of work assumes that model parameters (*e.g.,* user embeddings and item embeddings) learned from the past interactions store necessary information about the historical information. As a result, retraining the model using historical interactions is deemed unnecessary. Instead, a transfer module is designed to transfer knowledge stored in old models to new models. It can be a convolutional neural network (CNN), as in [110] and [119]. The authors first stack the model parameters learned from historical data instances and the model parameters learned from new interactions. Then CNN layers are applied on the stacked parameters, to obtain the final model parameters that retain both

historical information and new information. ASMG [120] applies Gated Recurrent Unit (GRU) on models trained using each period's data, to model long-term dependencies among information learned in previous time periods. For graph-based recommender model, the authors [121] adopt the temporal convolutional network (TCN) to fuse the user/item embeddings learned from previous periods with the new embeddings learned from the new interactions.

Knowledge Distillation. It avoids "catastrophic forgetting" by using distillation techniques. The general idea is to penalize the changes in the terms that distillation is applied on. GraphSAIL [111] adopts knowledge distillation to preserve both local and global structure of an old graph. The local structure refers to the interaction graphs in the previous period, while global structure is defined by the communities that users and items belong to. In addition, GraphSAIL applies knowledge distillation on node representations learned in the previous period. Ader [122] combines experience replay and knowledge distillation techniques. That is, distillation loss is applied on the exemplars obtained from experience replay. Similar technique can be seen in [123, 124]. DCMR [124] proposes a dual-constrained task sampler to construct replay tasks memory. Then distillation-based losses are designed to reproduce historical knowledge stored in the meta-model. The authors in [125] propose a layer-wise contrastive distillation loss on a graph neural network to retain information learned in each layer.

2.4.3 Dynamic Representation Learning

In [126], the authors highlight the dynamics in recommender system. Hence, they stress the needs of constructing a dynamic recommendation model to capture the evolving user and item features. The techniques proposed in [126] is point-process modeling. In other words, a user embedding is affected by the embeddings of the items he/she interacted with in the past, while the influence by different items are determined by the time at which the corresponding interactions took place. Similarly, item embeddings are influenced by the users who were associated with them in the past. Although Coevolve proposed in [126] models the changes in user embeddings and item embeddings over time, it considers the linear relationship between users and items. Contrary to Coevolve, DeepCoevolve [127] is proposed to conduct recurrent neural network (RNN) modelling which captures more complex

dynamics in user embedding and item embedding. As both Coevolve and DeepCoevolve predict interaction probability, it is naturally assumed that user embeddings and item embeddings at a future recommendation time are the same as those that happen at any time before the recommendation time and after the last observed interaction. In [128], the authors propose not to predict interaction probability to avoid fixed embeddings between two consecutive interactions. Instead, they design an algorithm, namely JODIE, to predict embedding trajectory. To achieve this objective, a new projection operator is introduced, which is capable of learning to estimate the user's embedding at any future time. In JODIE, the projection operator is constructed using two recurrent neural networks: one for predicting user embeddings and the other for predicting item embeddings. JODIE is designed to model user-item interactions that capture direct relationships between users and items. To account for indirect relationships, DGCF is proposed in [129] to use graph neural networks to model the dynamic graph structure in recommender system. In details, three embedding update operators have been proposed to inherit information from previous state by self-loop, 1-hop neighbor and 2-hop neighbor in previous graph.

2.5 Evaluation of Recommender System

In the previous sections, we detail collaborative filtering recommender system, recommendation model that considers temporal factors, and techniques used in recommender systems to model dynamics in user preference. Evaluations are needed to verify the effectiveness of these algorithms and models. In this section, we detail the different evaluation protocols adopted in recommender system. Moreover, we discuss the various challenges faced in offline evaluation of a recommender system.

There are three ways to evaluate a recommender system: user study, online evaluation and offline evaluation [130–135].

User study. A user study [131, 134, 136, 137] invites a number of participants to use the recommender system. While using the recommender system, user behaviours will be recorded for analysis. From that, we can conduct quantitative analysis in factors such as the time that users spend on tasks and click-through rate. Additional indicative behaviors, such as the movement of the eyes and gestures, could

also be examined and interpreted. Participants of the user study may also be given questionnaires which include closed or open-ended questions for users to provide qualitative feedback. User study could provide comprehensive results on user response to a recommender system. However, on one hand, a user study involves recruitment of participants, design of questionnaire, analysis of results and other related operations. The preparation and execution of user study are expensive in terms of time costs and financial costs. On the other hand, a user study may produce biased results because users may exhibit different behavior, knowing that they are being observed in a study [138].

Online evaluation. In online evaluation, a recommender system is deployed on a platform, then users' response to the recommender system will be tracked [133, 134]. Unlike user study, online evaluation involves users who are self-motivated to buy/click/rate items on the platform. As online evaluation records real users' reactions to a real recommender system deployed on a real-world platform, we deem online evaluation as a setting that produces the most reliable outcomes. A widely adopted online evaluation setting is A/B testing [139, 140]. That is, two groups of users are assigned to two recommender systems respectively. One recommender system is the existing system, while the other one is the target system under testing. The users do not have information about which group they belong to. Through analysis on measures like click-through rate and watching time on video, we can understand which recommender system is performing better and answer the question of "Should we replace the old recommender system with the new one?"

Offline evaluation. A recommender system can be evaluated using benchmark datasets in offline setting. The datasets contain pre-collected user feedback [17, 141]. The feedback can be explicit feedback (*e.g.,* ratings on items) or implicit feedback (*e.g.,* clicks/purchases on items). We name user feedback on items to be user-item interactions. In offline evaluation, a portion of user-item interactions is masked as the test set. The remaining unmasked interactions are training set which is used to train the recommendation model. The well-trained recommendation model is then used to predict the interactions in the test set. The prediction accuracy is reported as the recommendation accuracy of the recommender system.

Among the three aforementioned evaluation methods, offline evaluation is the most widely adopted setting in research papers of recommender system [135, 142]. In [142], the authors show that out of the 117 papers published in conferences

AAAI [1] and IJCAI [2] in years 2018 and 2019, more than 92% of the papers follows an offline evaluation setting. In [135], the authors reviewed 70 recommendation approaches that are evaluated using either user study, online evaluation or offline evaluation, they find that 69% of the approaches are evaluated in an offline setting. The key reason that offline evaluation is more popular is that offline evaluation is cheaper compared to user study and online evaluation. As it is the most widely adopted setting, we would focus on offline evaluation in this thesis. For the remaining parts of this thesis, when we say evaluation, it is referring to offline evaluation, unless otherwise stated.

2.5.1 Evaluation Protocol

In offline evaluation, the pre-collected user-item interactions are split into training and test sets. The splitting strategies can be categorized as random-split, leave-one-out split and temporal split [15, 17, 21, 143–145]. Random-split is to randomly select a subset of interactions as training instances, while the remaining are treated as test set. Leave-one-out split means taking users' last interactions or last few interactions as test instances, while treating the remaining interactions as training instances. For temporal split, we fix a time point, interactions that happen before the time point form a training set, while interactions after the time point are test instances.

Other than the general evaluation setting mentioned, prequential evaluation is used to evaluate recommender system in a streaming environment [112, 113, 146]. This methodology adheres to a *test-then-learn* procedure [112]. Specifically, upon the observation of a new interaction, the recommender system's capacity to predict that interaction is documented. Subsequently, the recommender system undergoes a refresh, incorporating the details of the specific new interaction. However, relying solely on the evaluation of a single new interaction may introduce bias, thereby complicating fair comparisons between different recommendation algorithms. As a result, a relaxed variant of prequential evaluation is adopted. It involves utilizing a batch of new interactions instead of a single interaction in the *test-then-learn* process [110, 111, 117]. The collection of new interactions in the batch may encompass all interactions occurring within a designated time window, such as one year and

[1] https://aaai.org/
[2] https://www.ijcai.org/

one week. Alternatively, it could involve a fixed quantity of interactions, providing flexibility in the evaluation process.

2.5.2 Evaluation Metrics

To quantify the performance of a recommender system, researchers adopt various evaluation metrics [130, 133, 147].

To measure the prediction accuracy of a rating prediction task in recommendation, metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are leverage [148, 149]. These measures quantify the difference between the predicted score and the actual score. Hence, the higher the values of metrics, the worse the prediction accuracy. Adoption of MAE, MASE and RMSE requires the benchmark dataset to contain user ratings on items. However, there are datasets whereby only user behaviours (*e.g.,* clicks, purchase and watch) are recorded. The datasets only record whether a user interacts with an item, and there is no explicit score. Hence, we name user behaviours, in this case, as implicit feedback.

For datasets with implicit feedback, a top-n recommendation task is often formulated. That is, instead of predicting users' ratings on items, a recommendation system generates a list of n items that the model predicts to be of interest to the user. For top-n recommendation, performance is often measured by metrics like Recall, Precision, Hit Rate (HR), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG) [150, 151]. Among the five metrics mentioned, recall, precision and HR measure the magnitude of getting the correct recommendations among the n items recommended. The higher the value of recall, precision and HR, the more accurate the recommendations are. As for MRR and NDCG, they measure the ranking position of the correct recommendation among the n items recommended. Ideally, the ground-truth item should be ranked at the top position. A higher value of MRR or NDCG indicates that the ground-truth item is ranked nearer to the top position.

Other than recommendation accuracy and ranking accuracy, other measures that are related to user satisfaction are also discussed in some research papers [25, 133,

152–154]. Common beyond-accuracy metrics include diversity, novelty, serendipity and coverage. In summary, diversity evaluates the dissimilarity among the n recommended items, while novelty assesses the quantity of new items suggested to users [152, 154]. Serendipity, which is linked to novelty, gauges the degree of surprise in the recommended items [155]. Finally, coverage quantifies the proportion of all the available items that are suggested to users [156].

Although diversity, novelty, serendipity and coverage are related to user satisfaction, the key metric is still on recommendation accuracy. Only if prediction accuracy of user preference is ensured, then the recommender system can consider other factors like diversity to further enhance user satisfaction.

2.5.3 Challenges in Evaluating a Recommender System

Despite the proposals of the various evaluation protocols, evaluating recommender systems is still a challenging task. In [157], the authors identify four challenges in evaluating recommender systems: (i) potential bias towards highly reachable items; (ii) differences in interaction distribution between log data from an installed recommender system and interactions from users with no prior exposure to any recommender; (iii) the lack of a direct correlation between high click-through rate and revenue; and (iv) the difficulty of evaluating a system's impact on revenue, as it is unclear whether users would have purchased the items without the recommender. In a recent study [158], the authors further investigate the discrepancy between false-positive metrics and true-positive metrics in evaluating recommender systems. They find that false-positive metrics are influenced by popularity biases, but in the opposite direction compared to true-positive metrics. Another study [159] highlights that datasets used for offline evaluation of recommender systems are often biased due to the deployed recommender system. As a result, new recommendation models are evaluated based on their ability to reproduce interactions obtained from the deployed system, rather than how well they predict user preferences.

Furthermore, a common practice in offline evaluation of a recommender system is to evaluate with sampled metrics [160]. That is, instead of ranking the ground-truth item over all items, researchers rank the ground-truth item over a sampled set of items. The key reason of sampled metrics is that ranking over the entire set of items is computationally expensive, especially when the number of items

is large. The authors in [160] show that full-ranking and sampled-ranking could potentially lead to different results. A sampled metric is not a good indicator of the actual performance of the recommender system. Hence, the authors call for avoiding using sampling in evaluation.

Other than the challenges identified in above-mentioned papers, it is also unclear how to design an offline evaluation for a recommender system. There is currently no consensus on the most appropriate evaluation approach. Decisions on the offline evaluation settings, such as dataset splitting, heavily depend on the researcher's choice [17, 143, 161]. This can lead to non-comparable results, as shown by various studies. For instance, reported in [15], the performance of the published models vary significantly depending on the data partition scheme used. Said and Bellogín [162] have observed similar non-comparable recommendation results when using different data splitting strategies in offline experiments. The authors of [143] then conduct controlled experiments using *random-split-by-ratio* and *temporal-split-by-ratio* on various datasets, *i.e.,* MovieLens-1M, Yelp, Epinions, and Amazon electronic dataset. They find that random-split generally leads to better performance than temporal-split. In this thesis, we argue that this comparison is not meaningful due to the problem of data leakage associated with the random split approach. Consequently, the obtained recommendation results from random-split are unrealistic. Similarly, in [16], the authors conclude that the discrepancy between recommendation results obtained from *random-split-by-user* and *temporal-split* is due to the failure to consider time. Hence, results obtained from *random-split-by-user* are not reflective of the actual performance of a recommender in a practical situation.

Some studies have stressed the necessity of considering "time" factor in offline evaluation of recommender system [18–20, 163]. The authors of [163] argue that a setting which strictly follow a global timeline is a more realistic setting for evaluation. They show that different recommendation results can be observed from the experiment that follows the global timeline and the experiment that uses less strict setting. In [19], the authors emphasize that recommender systems are subject to constant evolution as new items and users enter the system. To handle the dynamic nature of recommender systems, they propose a strategy called "temporal leave-one-out", which treats each user independently. Using this strategy, the interactions of each user are sorted chronologically, and recommendations are made

based on historical interactions that occurred before the timestamp of the new interaction. Additionally, Verachtert et al. [164] show that the optimal time window of training data could vary for different recommendation algorithms. They suggest the tuning of time window for more reflective recommendation performance for each recommendation model. In [20] and [18] also consider the impact of temporal factors on recommendation performance, demonstrating that performance varies over time as more interactions are observed. They argue that this variability suggests that context changes over time in recommendations. Hence, it is essential to consider the global timeline in evaluation of recommender system.

Failure to consider the global timeline may lead to data leakage issue. In this thesis, the issue of data leakage in offline evaluation of recommender system is identified. Other studies also note that time-independent data partition strategies may result in the leakage of "future information" in training [9, 15, 163, 165].

2.6 Summary

This chapter provides an overview of three types of recommender systems: collaborative filtering, content-based, and hybrid recommender systems. As we establish the importance of considering the temporal factors in Chapter 1, we further describe the existing research papers that consider the temporal factors in this chapter. Additionally, we explore algorithms and techniques that effectively handle the dynamic nature of recommender systems in Section 2.4.3. Lastly, we delve into the evaluation options for recommender system and list various challenges faced during the evaluation.

Chapter 3

Revisit Offline Evaluation of Recommender System from A Global Timeline Perspective

3.1 Introduction

Over the past decade, recommender systems have gained significant attention in both academia [25, 26] and industry [139, 166]. Numerous solutions have been proposed, ranging from popularity-based model to nearest neighbor-based methods, and more recently, to model-based recommendation algorithms [25, 34, 143, 167]. The field has also witnessed rapid advancements in deep learning-based recommender models [8]. However, there are concerns about the actual progress made in this research area [6, 7] and reports of inconsistencies in model performance due to different data splitting strategies [15, 21, 144]. We have detailed the issues in Chapter 1. These issues underscore the need to re-evaluate the evaluation protocols of recommender systems to ensure that recommendation models are being assessed in a realistic manner.[1] Before we delve into recommendation performance evaluation concerns, it is important to revisit the fundamental problem definition of recommender systems. To achieve this, we draw on established sources such as

[1]This chapter is published as Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20). Association for Computing Machinery, New York, NY, USA, 1749–1752. [168]

TABLE 3.1: Statistics of the four datasets used in this study

Dataset	Time span selected	Data filtering	#User	#Item	#Interaction
MovieLens-25M	21/11/2009 to 20/11/2019	No filtering	62,202	56,774	9,808,925
Yelp	13/12/2009 to 12/12/2019	10-core	116,655	61,027	3,127,215
Amazon-music	02/10/2008 to 01/10/2018	5-core	11,651	9,243	114,833
Amazon-electronic	05/10/2008 to 04/10/2018	10-core	109,990	39,552	1,752,238

the recommender system handbook [25] and five survey papers [8, 133, 169, 170], which provide a generic definition of the problem. According to this definition, the task of a recommender system involves learning the preferences of users based on their past interactions with items, with the goal of predicting which item a user will interact with in the near future. This definition is predicated on the notion of a **global timeline**, where "history" and "future" do not intersect along the timeline for any given user at any particular point in time. What is important, "history" happens before "future".

The definition of recommender system provided above details a top-N recommendation task. This task involves recommending N items that a user is most likely to be interested in. Although rating prediction is another popular task in recommender systems, it is distinct from top-N recommendation. Rating prediction focuses on predicting the score a user would give to an item, rather than predicting whether a user would rate an item. In this research, we concentrate on the top-N recommendation task since it has been the subject of more extensive research in recent years [171]. Furthermore, the ultimate aim of rating prediction is to identify the items that a user would give a high score and recommend those items to the user. In other words, rating prediction is aimed at improving the accuracy of top-N recommendations. Therefore, this thesis focuses on the top-N recommendation task.

We argue that while the *global timeline* is included in the problem definition for recommender systems, it is often disregarded in offline evaluation scenarios. This disregard for the global timeline refers to cases where events are not evaluated in their chronological order along the timeline. Our argument is that neglecting the timeline creates two significant issues.

One of the major problems with ignorance of the global timeline is that it may

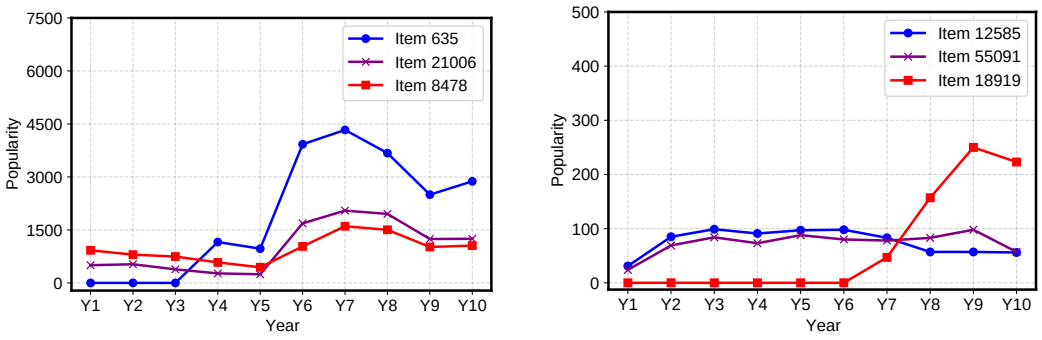
result in failure to capture the overall temporal context of the data. While previous research [172, 173] has examined dynamic trends in datasets like Douban and Amazon, this thesis seeks to further illustrate the importance of the temporal context in three additional datasets: MovieLens-25M, Yelp, and Amazon-electronic. To demonstrate this, we present three examples of how temporal context impacts recommender systems.

Firstly, in Figure 3.1, we plot the yearly popularity of three sampled items from two datasets (MovieLens-25M and Yelp) over a ten-year period, showing how the popularity of items changes over time. The details of the two datasets can be found in Table 3.1. From Figure 3.1, We note that not all items are available for ratings from the beginning of the timeline. For instance, item 18919 in Yelp received its first rating only after Year 7, becoming popular from that point onwards. However, most recommender models, including popularity-based ones, ignore this temporal context in their evaluations.

Another example of temporal context is the release of different iPad models by Apple. In Figure 3.2b, we list nine iPad models and their corresponding release dates. We then match these models with products in the Amazon-electronic dataset and plot their activeness (i.e., number of reviews) over time. From this plot, we find that the activeness of iPads in Amazon-electronic can be used to infer their release dates. That is due to the fact that iPads start receiving reviews only from their release years. Moreover, we find that iPads experience a surge in popularity in the year of their release but gradually lose consumer attention over the following years.

If the global timeline is not observed, trends such as newly released products and the dynamic popularity of products cannot be captured effectively. This creates a fundamental problem for recommender systems, as they are designed to predict and recommend items to users based on their previous interactions. Therefore, ignoring the temporal context of user-item interactions undermines the accuracy and reliability of the recommendation algorithms.

The second issue pertains to data leakage problem which can occur during evaluation due to the inappropriate train-test data splitting. Inappropriate train-test split can lead to ignorance of the global timeline. In this research, we define "data leakage" as predicting a test instance with a model learned from training instances

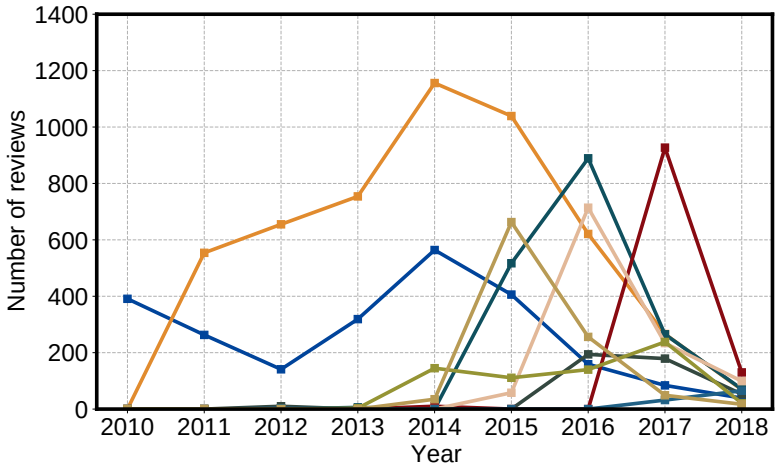


(A) MovieLens-25M (B) Yelp

FIGURE 3.1: Yearly popularity of three sampled items from MovieLens-25M and Yelp respectively, over ten-year period ($Y1$ to $Y10$).

that happen after the timestamp of the test instance. Table 3.2 lists the commonly used data partitioning techniques in recommender system, and shows whether ignorance of the global timeline exists. With ignorance of the global timeline, the training set contains user-item interactions that have occurred after the test instances in the test set. Consequently, the model learns from future interactions to predict past user preferences, which is a problem because a model can never have access to data instances from the future in real-world settings. For example, a model shall not learn from user interactions with iPad 3 that is released in 2012, to predict user's preference on iPad 1 any time in the year of 2010. This violates the problem definition of a recommender system, where predictions of test instances should only be based on model learned from interactions that happened before test instances. As a result, evaluations conducted under such circumstances may not accurately reflect a model's performance in real-world settings. In this thesis, we argue that data instances that happen after the test instances should be kept private to the model.

In this thesis, we provide a thorough analysis of the impact of data leakage on the offline evaluation of recommender systems. Although previous research [9, 163, 165] has briefed the concept of data leakage, we present the first comprehensive explanation for the phenomenon, which is caused by collaborative filtering not following the global timeline in offline evaluation. This explanation is presented in Section 3.2 of our paper. Moreover, we also quantitatively assess the effect of data leakage by conducting experiments with four widely used datasets: MovieLens-25M, Yelp, Amazon-music, and Amazon-electronic, using leave-one-out data split. We evaluate four popular models, namely Bayesian Personalized Ranking (BPR) [174], Neural



(A) Activeness of Ipad models

Line	Product Name	Product ID in Amazon-electronic Dataset	Official Release Date
	Ipad 1	B002C7481G	03 Apr 2010
	Ipad 2	B0013FRNKG	11 Mar 2011
	Ipad 3	B00746NEJW	16 Mar 2012
	Ipad Mini	B00746WCEA	02 Nov 2012
	Ipad Air	B00G2X1VIY	01 Nov 2013
	Ipad Mini 2	B00GQDAFKA	12 Nov 2013
	Ipad Mini 3	B00OTXGLW0	22 Oct 2014
	Ipad Air 2	B00OSKREVG	22 Oct 2014
	Ipad Pro	B0155KDJWA	11 Nov 2015

(B) iPad models' official release dates and their IDs in Amazon-electronic dataset

FIGURE 3.2: The figure presents the activeness of different iPad models in the Amazon-electronic dataset from 2010 to 2018. In addition, a table is provided listing the release dates of each iPad model and their corresponding product ID in the Amazon-electronic dataset.

Matrix Factorization (NeuMF) [13], SASRec [89], and LightGCN [49]. These models are chosen for their representation of different modeling techniques, and they all often serve as baseline models in the literature. Our findings from experiments are summarized as follows:

- The evaluation results from all four baseline models indicate that they recommend "future items" to past test instances. Future items refer to items that are only available in the system after the time point of the test instance. For example, a model recommends iPad Pro, which was released two years

TABLE 3.2: Commonly adopted data split strategies in offline evaluation of recommender systems. The table further shows whether local timeline (*e.g.,* time specific to a user) or global timeline is followed, and whether data leakage exits.

Data split strategy	Definition of training and test instances	Local timeline	Global timeline	Data leakage
Random-split-by-ratio	Randomly sample a percentage of user-item interactions as test instances; the remaining are training instances.	No	No	Yes
Random-split-by-user	Randomly sample a percentage of users, and take all their interactions as test instances; the remaining instances from other users are training instances.	No	No	Yes
Leave-one-out-split	Consider each user's last interaction as a test instance; all remaining interactions are training instances.	Yes	No	Yes
Split-by-timepoint	All interactions after a time point are test instances; interactions before this time point are training instances.	No	Yes	No

later in 2015, to a test instance that occurred in 2013. This observation provides strong evidence that the mainstream offline evaluation setting, which disregards the global timeline, is flawed.

- The inclusion of "future training instances" in the training set alters the distribution of user-item interactions. As a result, the training set that contains such data is not an accurate representation of the temporal context at the time of a test instance. Consequently, the top-N recommendation lists obtained from experiments that incorporate future data exhibit notable discrepancies compared to those without data leakage

- The accuracy of a recommender model is affected by data leakage. The effect of data leakage on recommender models *cannot be predicted*. Specifically, it

is not necessarily true that the inclusion of more future data in training will lead to increased accuracy. The addition of future data may either improve or diminish the accuracy of recommendations.

- The inclusion of different amounts of "'future data" in our experiments results in varying ranking orders of the baseline models. Therefore, it is impractical and meaningless to compare the performance of recommendation models in the presence of data leakage during offline evaluation. We believe that the failure to observe the global timeline during evaluation is a major obstacle to reproducibility in recommender systems.

3.2 Data Leakage in Offline Evaluation

When a model is deployed in a production environment, it is inherently subjected to a global timeline. In other words, the recommender system can only learn from historical user-item interactions and generate recommendations based on real-time requests. Nevertheless, due to the constraints in accessing online platforms, academic research on recommender systems is often limited to offline evaluations that utilize static datasets [133, 135].

Handling datasets in a static environment demands meticulous consideration to avoid potential data leakage issues. In particular, we defined "data leakage" as a situation where a recommendation model assesses a test instance that occurred chronologically before the corresponding training data, introducing inaccuracies in the evaluation process. In remaining of this section, we elaborate on the mechanisms through which data leakage may arise in the offline evaluation of recommender systems.

In a conventional offline evaluation scenario, a user-item interaction matrix of size $m \times n$ is created, where m represents the number of users and n represents the number of items in the dataset. Each entry of this matrix denotes a user's interaction with an item, which can either be an explicit score between 1 to 5 or an implicit indicator of purchase, rating, or click behavior (1 for positive and 0 for negative). In this thesis, we consider only implicit user-item interactions, i.e., the act of interaction is important, not the rating value itself. Therefore, our focus is

on the top-N recommendation task, which involves predicting the positive ("1") entries in the user-item interaction matrix.

To evaluate recommendation accuracy, a subset of user-item interactions in the user-item interaction matrix is masked as a test set, and the remaining known entries are used as the training set. The recommendation model is trained using the training set and evaluated using the test instances. Various strategies have been used to sample the user-item interactions to be masked to form a test set, as summarized in Table 3.2. However, we note that only the split-by-timepoint strategy observes the global timeline.

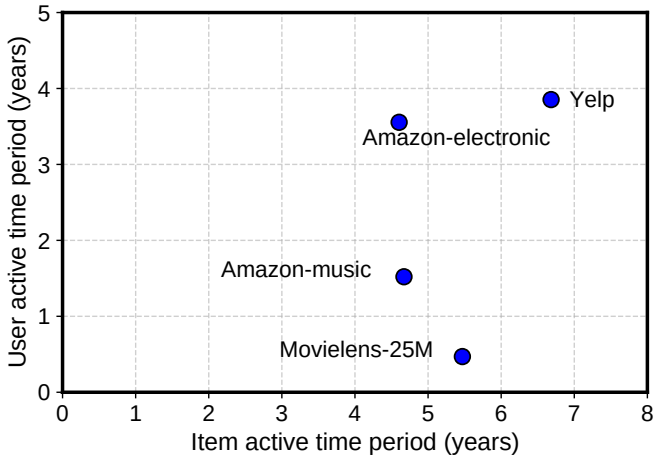
In most scenarios, the entire training set is utilized to train a recommender model as a whole, as seen in many studies [17, 143]. This implies that the training set is treated as a static dataset, without considering any time-based changes. Although some time-aware recommendation models [9, 85, 89, 165] do incorporate timestamp information of training instances as an attribute, they still consider the entire training set as a *static dataset*. They are not intended to observe the global timeline. Consequently, these models are inevitably evaluated with data leakage.

3.2.1 Global Timeline and Local Timeline

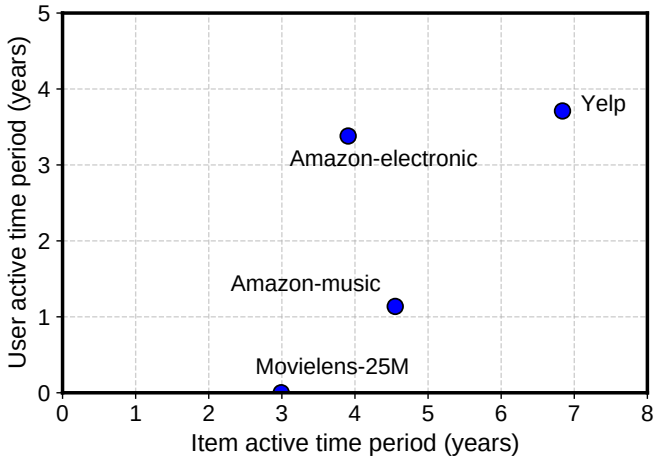
Our discussion highlights the significance of accounting for the global timeline, and to demonstrate this, we employ four benchmark datasets to showcase the user-item interaction distributions over the global timeline. The four datasets we use are MovieLens-25M[2], Yelp[3], Amazon-music[4], and Amazon-electronic[5]. We extract the interactions within a ten-year period from each dataset and provide the dataset statistics in Table 3.1. Further information on data filtering techniques such as the k-core approach and dataset preparation is presented in Section 3.3.

Not all users or items are active throughout the entire 10-year period covered by the datasets, and some items may not have been available for rating starting the time period. For instance, a Yelp user who registered in 2018 may have only started reviewing businesses on Yelp from that year onwards, and a movie that was released

[2]https://grouplens.org/datasets/movielens/25m/
[3]https://www.kaggle.com/yelp-dataset/yelp-dataset
[4]https://jmcauley.ucsd.edu/data/amazon/
[5]https://jmcauley.ucsd.edu/data/amazon/



(A) Mean active time period in years.

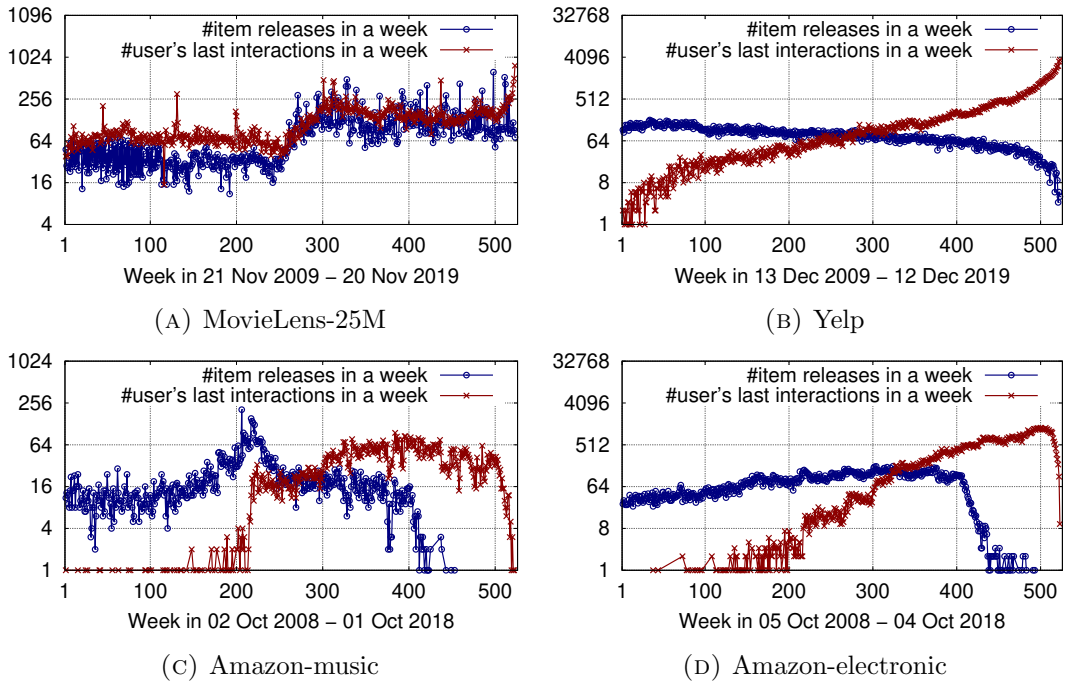


(B) Median active time period in years

FIGURE 3.3: Mean and median active time period of users and items in the four datasets.

in 2018 can only receive ratings from that year onwards. To capture this variation, we define the *active time period* of a user or item as the time between their first and last interactions recorded in the dataset. This *local timeline* is specific to each user or item, and covers the period between their first and last interactions.

The mean and median *active time periods* of users and items in the four datasets are presented in Figure 3.3 to provide a comprehensive overview. Due to the different nature of items such as movies, music, and restaurants, the distributions of user and item activeness vary among the four datasets. Nevertheless, all four datasets show that the mean and median active time of items are significantly longer than those of users. More importantly, the mean and median user active time periods are less than 4 years in all datasets, which is less than half of the 10-year period



(A) MovieLens-25M (B) Yelp

(C) Amazon-music (D) Amazon-electronic

FIGURE 3.4: Users' last interactions and item releases in each week, in the 10-year period. (Best viewed in color)

covered by the datasets. In the MovieLens-25M dataset, we observe users tend to be active for a very short period of time, often less than 1 year. Even more, a considerable number of users in the Movielens dataset rate all their movies within a single day.[6]

To better visualize the distribution of user and item activeness throughout the global timeline, we plot the number of item releases and the number of users' last interactions that occurred each week in Figure 3.4. For an item's release time, we consider the time at which it receives its first rating from any user. The blue line represents the number of items released each week within the 10-year period, totaling 520 weeks, while the other line represents the number of users whose last interactions happened in each week throughout the 10-year span. It is apparent that in all four datasets, there is a continual introduction of new items. Moreover, users' last interactions happen at any given time.

[6]Many users are only active for one day. What is worse, many interactions of a user are completed within one single timestamp. Hence, we note that MovieLens does not contain reliable timestamp information. In [175], the authors make the same observations. Despite that, we include MovieLens in this thesis for its extreme popularity in the research of recommender system [143, 176].

To summarize, it should be noted that not all users are active for the entire time period covered by a dataset, and not all items are accessible to be interacted with since the start of the dataset in the context of the global timeline. Furthermore, as depicted in Figure 3.1, temporal dynamics are evident in user-item interactions over the global timeline. Temporal dynamics can also be deduced from Figure 3.2.

3.2.2 Data Leakage in Recommender System

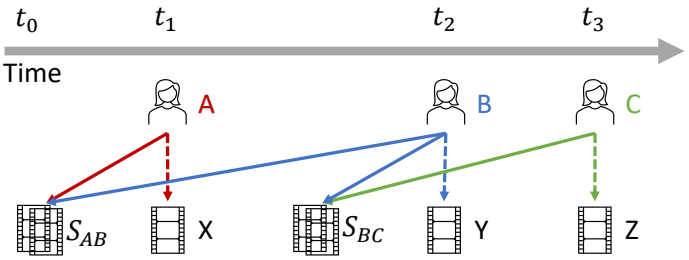
According to Wikipedia[7], data leakage refers to "the use of information in the model training process that would not be expected to be available at prediction time". It is considered one of the top ten data mining mistakes. Kaufman *et al.* also provide a critical discussion on data leakage and recommend "time separation" of train and test data as a methodology for avoiding it, similar to the split-by-timepoint strategy mentioned in Table 3.2. It is essential to distinguish between data leakage and privacy leaks. Data leakage refers to the incorrect inclusion of information in the model training process that would not typically be available at prediction time, while privacy leaks refer to the unauthorized disclosure of personal information to third parties.

To better understand the data leakage problem, we present an illustration in Figure 3.5. Considering a system with a global timeline spanning from time t_0 to time t_3, where three users A, B, and C interact with items in the system. In Figure 3.5a, we plot the activities of the users and items along the global timeline, with the items placed in the plot at the time points they are introduced to the system (*i.e.,*, their release time), and the users placed in the plot at the time points of their last interactions.[8] For instance, the last rating by user A is on item X at time t_1, and there are no more interactions by user A after time t_1 in the dataset. We use S_{AB} to denote the set of items rated by both users A and B, and S_{BC} to denote the items rated by both users B and C.

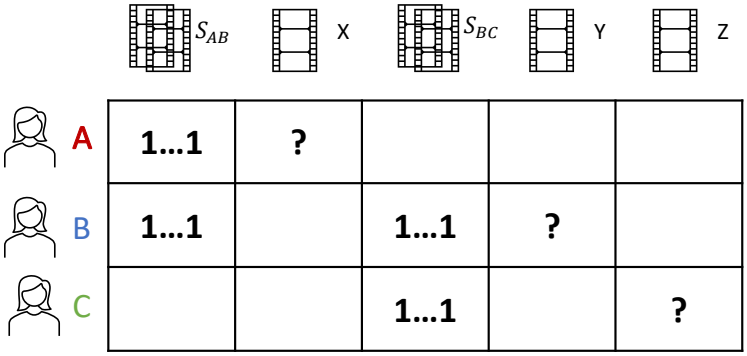
Then, let us assume that we are using the *leave-one-out-split* strategy in this illustration, where the last interaction of each user is used as a test instance. Figure 3.5a shows the test interactions of the three users, indicated by dotted lines. In Figure 3.5b, we present a matrix view of the same example. The test instances are

[7] `https://en.wikipedia.org/wiki/Leakage_(machine_learning)`

[8] To simplify the presentation, we assume that once an item is released for rating, it remains in the system and is not removed.



(A) User-item interaction along global timeline.



(B) User-item interaction matrix

FIGURE 3.5: Interactions along global timeline, and in matrix form.

masked with a question mark '?', and we use '1 . . . 1' to indicate the ratings given by users to multiple items in S_{AB} and S_{BC}.

As mentioned earlier, when the entire training set is provided as a static set to a recommender system without considering the global timeline, users with common ratings to items in S_{AB} or S_{BC} may appear to have a high level of similarity. For instance, users A and B may be considered as similar users due to their common ratings to items in S_{AB}, and similarly, users B and C may appear to be similar because of their common ratings to items in S_{BC}. Through learning with collaborative filtering recommendation model, users A and C may be inferred as having a high level of similarity.

However, this approach leads to a data leakage problem because all items are plotted in the figure at their release time, and all items rated by user C are only available after user A's last interaction (i.e., time point t_1). In reality, these items and their interactions are not expected to be available at time point t_1. Thus, when predicting user A's preference for item X, we are using "future data" that is not expected to be available at the prediction time, resulting in data leakage. The model may even recommend items that are rated by user C, which are "future items", to

user A because of the learned similarity between users A and C. In Table 3.2, it is indicated that when using a *random split* for training and testing data, neither the local nor the global timeline is taken into account. Consequently, this type of split can lead to data leakage, similar to the leave-one-out-split method. By not considering the global timeline, the model may use a user's future interactions to predict their current interactions.

A recent study [143] reviewed 85 recommender system related papers that were published in the highly ranked conferences (such as SIGIR, WWW, and KDD) between 2017 and 2019. The study found that among the 85 papers, 53.7% of them employed a random split method for evaluation, 28% of the papers used the leave-one-out method, and only 12.3% of the papers utilized the split-by-timepoint method. Therefore, a very small number of offline evaluations are based on separating the training and testing data by a fixed time point to avoid data leakage. As a result, a large number of offline evaluations for recommender systems are conducted without considering the issue of data leakage. In a more recent study [177], it has been shown that 59.1% of the papers published in ACM RecSys conference in 2020 to 2022 do not follow the global timeline. Consequently, most reported results do not accurately reflect a model's true performance in a production setting.

This section presents the idea that the context changes over the global timeline. This means that users may have varying levels of activity or inactivity at different time points, and items may become available at any given time. It is, therefore, crucial to model the temporal context along the global timeline when evaluating recommender systems offline to avoid data leakage. If the global timeline is not considered, it may lead to data leakage issue. It is important to consider the impact of data leakage on recommendation accuracy, as this determines the extent to which reported performance in literature truly reflects a model's real-world performance. If the impact is insignificant, the evaluation process does not require any alterations. However, if it is substantial, a more realistic evaluation approach is required for a fair comparison of various recommendation models. The following sections (Sections 3.3 and 3.4) examine the impact of data leakage on recommendation results with varying degrees of severity.

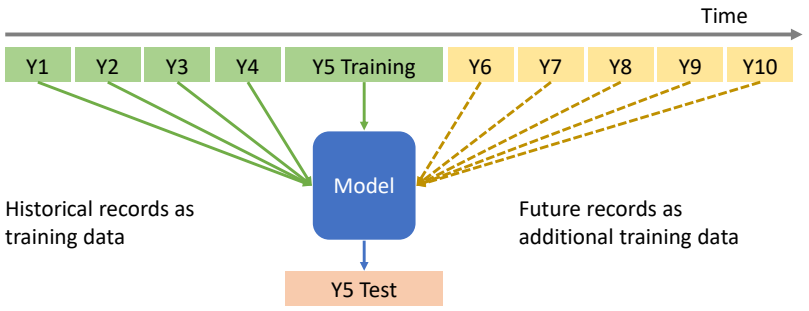


FIGURE 3.6: Train/test data split for test year $Y5$ as an example.

3.3 Experiment Design and Setup

Various offline evaluation techniques are used to split data into training and testing sets, and only the split-by-timepoint approach is immune to data leakage. To eliminate the influence of random factors, this study employs the leave-one-out split method, where a user's last interaction is masked as a test instance. As depicted in Figure 3.4, a user's last interaction can occur at any point in the global timeline, and all interactions that follow it are considered "future data" for that particular test instance. By this definition, the amount of future data available to each test instance varies, as the test instances obtained from leave-one-out split can arise at any point along the global timeline. Therefore, studying the impact of data leakage on individual user/test instance is computationally expensive. Hence, this study does not examine the issue at a granular level.

To investigate the impact of data leakage, we conduct experiments that simulate varying degrees of severity at a coarse-grained level. In this approach, we divide the entire dataset into multiple time windows, each spanning a fixed time interval of one year. We then add user-item interactions from different numbers of time windows in the training set to include various amounts of "future data" for training. This approach is considered coarse-grained because we do not factor in the differences in future data for individual test instance that occurs within the test year.

In the experiment, we follow a series of steps. Firstly, we arrange the user-item interactions chronologically and divide the dataset into fixed time windows. Suppose the global timeline of the dataset is 10 years and the time window is one year, we denote the first year as $Y1$, and the last year as $Y10$. Assuming that the test year selected is $Y5$, we then select all test instances in $Y5$, where a user's last interaction falls within year 5, as the evaluation subset (based on *leave-one-out split*).

Interactions that take place before $Y5$, *i.e.*,, $Y1$ to $Y4$, are considered as "historical interactions", and those that occur after $Y5$, *i.e.*,, $Y6$ to $Y10$, are labeled as "future interactions". We perform multiple experiments to simulate different degrees of data leakage. In the first experiment, we use all historical interactions from $Y1$ to $Y4$, and the training instances in $Y5$ (highlighted in green in Figure 3.6) as training data. In the second run, we include all interactions in $Y6$ in addition to the data from the first experiment. The third run includes all interactions in $Y7$, and this process continues until all user-item interactions up to $Y10$ are incorporated into the training data. This process creates a gradually increasing data leakage scenario for the test instances in $Y5$. The first experiment has the least data leakage, while the last experiment suffers from the most significant data leakage. By studying the recommendation results of the models in these different scenarios, we quantify the impact of data leakage.

3.3.1 Dataset

Our experiments involve four datasets: MovieLens-25M, Yelp, Amazon-music, and Amazon-electronic. The selection of these datasets is motivated by two factors. Firstly, they are popular and commonly used in both time-aware and non-time-aware recommender systems [143]. Secondly, all these datasets include timestamps of ratings, which allow us to reconstruct the global timeline.

We approach the recommendation task in a top-N recommendation setting using implicit feedback for all four datasets. To ensure fair comparisons, we extract interaction data spanning 10 years and remove any duplicates. We perform k-core filtering to exclude inactive users and items for Yelp, Amazon-music,and Amazon-electronic datasets. For Yelp and Amazon-electronic, we apply 10-core filtering, while for Amazon-music, we use 5-core filtering to retain a reasonable number of users and items. No filtering is applied to MovieLens-25M, because the original version of the dataset includes only users who rated at least 20 movies. We only keep ratings from users whose first rating occurred after the start of the 10-year period to ensure that we have a complete view of each user's interactions in the dataset.

The evaluation time window is set to one year, and we gradually incorporate "future interactions" year by year for evaluation. Our evaluation is performed on two

TABLE 3.3: Number of test and training instances for test years $Y5$ and $Y7$. From the test year till $Y10$, more future data are made available to training.

Dataset	Test year	#Test instances	#Training instances accumulated till adding Yx's data					
			$Y5$	$Y6$	$Y7$	$Y8$	$Y9$	$Y10$
MovieLens -25M	$Y5$	3,171	2,489,066	3,876,800	5,602,278	7,243,348	8,474,179	9,805,754
	$Y7$	9,232	-	-	5,596,217	7,237,287	8,468,118	9,799,693
Yelp	$Y5$	3,093	878,494	1,280,070	1,723,554	2,203,266	2,702,445	3,124,122
	$Y7$	7,241	-	-	1,719,406	2,199,118	2,698,297	3,119,974
Amazon -music	$Y5$	829	18,283	38,873	71,227	95,571	108,496	114,004
	$Y7$	2,686	-	-	69,370	93,714	106,639	112,147
Amazon -electronic	$Y5$	652	234,398	479,507	898,947	1,317,418	1,607,543	1,751,586
	$Y7$	8,747	-	-	890,852	1,309,323	1,599,448	1,743,491

different test years, namely $Y5$ and $Y7$. The reason for selecting two test years is to have a more comprehensive understanding of the problem, to avoid any potential biases that may arise in a single year. As we increase the number of "future training instances" in our experiments to simulate a more severe data leakage problem, we summarize the number of training and test instances for each dataset in Table 3.3.

3.3.2 Models

The field of recommender systems is highly dynamic and researchers have proposed a plethora of models [7, 8, 25]. For our study, we have chosen four commonly used baseline models, each belonging to a distinct recommendation model family.

- **BPR** [174] is a machine learning-based approach that learns a matrix factorization model using pairwise ranking loss. To optimize its performance, we conduct hyperparameter tuning using continuous random search. Specifically, we explore the latent dimension from 8 to 128, learning rates from 1e−6 to 1e−1, and regularization coefficients from 1e−4 to 1e−1.
- **NeuMF** [13] combines matrix factorization and multi-layer perceptron layers to learn the user-item interaction function. To find the best hyperparameters, we perform continuous random search and test various values for the latent dimension ranging from 8 to 128, the learning rate ranging from 1e−5 to 1e−1, the number of negative instances ranging from 1 to 4, and the regularization coefficient ranging from 0 to 1e−4.

- **SASRec** [89] is a recommendation model that considers the sequence of user interactions. It employs a self-attentive network to learn dependencies between items in a user interaction sequence. We perform hyperparameter tuning using continuous random search with a range of values for learning rate (1e−5 to 1e−2), maximum historical interactions (1 to 50), latent dimension (8 to 256), and number of blocks (1 to 4).

- **LightGCN** [49] is a recommendation model based on graph neural networks, which uses a graph convolutional network model to make recommendations. To optimize the hyperparameters, we perform continuous random search over a range of values. Specifically, we test the latent dimension from 8 to 128, the number of layers from 2 to 4, the regularization coefficient from 1e−5 to 1e−1, and the learning rate from 1e−5 to 1e−1.

The validation set, comprising the second last interactions of the testing users, is used to tune the hyperparameters of each model, following the approach taken in many previous studies [178–180]. We conduct 50 hyperparameter search trials for each baseline model and select the optimal combination of hyperparameters.

3.3.3 Evaluation with Non-sampled Metrics

During the offline evaluation process, a recommender system suggests items to users from a group of potential choices. According to [181], various strategies can be employed for selecting these potential items, including *TestRatings*, *TestItems*, *TrainingItems*, *AllItems*, and *One-Plus-Random methodologies*. Among these strategies, only *AllItems* involves recommendation from all available items in the system, while the remaining strategies use a subset of item as the candidates.

In our evaluation, instead of generating recommendations from a subset of available items, we generate a list of the top-N recommendations by considering a comprehensive ranking of all the items present in the training data. This means we do not employ sampled metrics. Sampled metrics involve randomly selecting a small number of irrelevant items (*e.g.*,, 1000) and then ranking the relevant items solely within this limited sample. Recent research [160] has indicated that sampled metrics can be inconsistent with non-sampled metrics, failing to accurately reflect the performance of a model. Consequently, our results, based on non-sampled metrics,

yield considerably lower values compared to those reported using sampled metrics [13, 182]. Naturally, ranking a relevant item among all the irrelevant items is much more challenging than ranking within a smaller sampled set.

In this study, we formulate the recommendation task as top-20 recommendation. To assess the accuracy of the recommendation models, we employ two metrics: Hit Rate (**HR**) and Normalized Discounted Cumulative Gain (**NDCG**). HR measures the percentage of users for whom at least one relevant item is included in the top-20 recommended items. NDCG considers both the relevance of recommended items and their position in the list. It accounts for the diminishing returns of relevance for items lower in the ranked list. HR and NDCG replicate a real-world scenario in which the recommender system displays only a limited set of items. By evaluating the accuracy of recommendations through HR and considering the positional importance of accurate recommendations using NDCG, these metrics effectively mirror users' responses to the recommended items in the real-life setting.

Given this setup, we examine the impact of data leakage from two perspectives. First, we meticulously analyze the top-20 recommendation list for each test instance and investigate how data leakage affects these lists. Second, we investigate how HR@20 and NDCG@20 change when varying amounts of "future data" are included in the training process.

3.4 Experiment Results

In our experiments, our objective is to measure and quantify the effects of data leakage from two distinct angles: (1) the influence it has on the lists of top-N recommended items, and (2) its impact on the accuracy of the recommendations. In the following section, we will present the results of our study and provide a comprehensive explanation of our findings.

3.4.1 Impact on Top-N Recommendation List

Finding 1. *The models being evaluated eventually recommend items that are considered "future items". These items are exclusively available within the system only after the specific time point of a given test instance.*

In Section 3.2, we discuss the underlying cause of data leakage and provide an explanation supported by Figure 3.5. The collaborative filtering approach employed by the recommender system allows it to learn similarities between users, such as users A and C, as illustrated in Figure 3.5. Consequently, the recommender may recommend items that have been rated by user C to user A. However, it is important to note that all the items rated by user C are only available within the system after user A's last interaction. Therefore, from the perspective of user A's last interaction, these items are considered "future items", as they become accessible only at a later time point.

Through our experiments, we have observed that all four models in our evaluation do recommend future items. For each test instance in our experiments, we generate a top-N recommendation list using a recommendation model. Subsequently, we analyze the number of "future items" present among these N recommended items. Here, "future items" refer to items that become available in the system after the timestamp of the specific test instance. In our study, we set N to be 20 and present the count of future items in Table 3.4. Each entry in the table represents the total number of "future items" recommended across all the recommendation lists obtained for the test set. We report these numbers for both test years $Y5$ and $Y7$. As indicated in Table 3.4, with an increasing inclusion of "future data" in the training phase from $Y6$ to $Y10$ for test year $Y5$ and from $Y8$ to $Y10$ for test year $Y7$, we observe a consistent trend of an increasing number of "future items" being recommended by all models across all datasets.

The fact that the recommender systems in our evaluation are recommending "future items" for test instances provides strong evidence of data leakage, indicating that the current offline evaluation setting is flawed. This data leakage compromises the reliability of the evaluation process since the recommendations made are based on information that would not be available at the time of the test. Inspired by our study, the authors of [21] conduct a similar experiment. Their findings further validate that both random-split-by-ratio and temporal-split methodologies, when applied to a user's local timeline, lead to the generation of recommendations that include "future items". These recommendations, being based on information from a future time point, are considered invalid and further underscore the issue of data leakage in the evaluation process.

TABLE 3.4: Among the top-20 recommendations, the total number of future items recommended for test instances in $Y5$ and $Y7$, respectively, by the four models.

Model	Dataset	MovieLens-25M		Yelp		Amazon-music		Amazon-electronic	
	Test year	$Y5$	$Y7$	$Y5$	$Y7$	$Y5$	$Y7$	$Y5$	$Y7$
BPR	$Y5$	0	–	0	–	0	–	0	–
	$Y6$	0	–	421	–	615	–	79	–
	$Y7$	22	0	829	0	970	0	363	0
	$Y8$	7	11	2,365	504	1,101	651	263	200
	$Y9$	6	88	5,048	287	1,304	1,103	499	1,224
	$Y10$	4	81	1,851	1,598	1,197	1,155	200	583
NeuMF	$Y5$	0	–	0	–	0	–	0	–
	$Y6$	3	–	602	–	910	–	28	–
	$Y7$	7	0	1,631	0	1,501	0	1,303	0
	$Y8$	27	31	3,260	130	1,733	878	549	0
	$Y9$	22	6	3,542	1,177	1,491	1,276	729	216
	$Y10$	15	1	5,205	1,791	1,577	1,573	2,655	326
LightGCN	$Y5$	0	–	0	–	0	–	0	–
	$Y6$	11	–	369	–	626	–	37	–
	$Y7$	32	0	739	0	1,050	0	148	0
	$Y8$	116	189	1,070	569	998	632	367	220
	$Y9$	22	26	1,257	979	1,036	893	262	430
	$Y10$	15	58	1,103	1,360	1,152	1,029	260	470
SASRec	$Y5$	0	–	0	–	0	–	0	–
	$Y6$	315	–	967	–	906	–	216	–
	$Y7$	442	0	3,074	0	1,548	0	625	0
	$Y8$	144	489	2,228	2,666	1,814	1,341	487	1388
	$Y9$	342	403	3,162	2,893	1,982	1,376	20	3,209
	$Y10$	993	386	1,741	3,014	1,980	1,662	12	2,479

In a valid evaluation setting, a recommendation model should not have any knowledge of "future items" or any interactions associated with them. Apart from the issue of data leakage, recommending "future items" has a detrimental impact on recommendation accuracy. This is because recommending an item that is not yet available at the time of the test instance will never result in a successful hit. Consequently, the offline evaluation tends to underestimate the true performance of the model.

Merely filtering out the "future items" from the top-N recommendation list before calculating recommendation accuracy is not a viable solution. The presence of "future items" highlights the fundamental problem that the current offline evaluation setting is invalid. It is essential to acknowledge that a model should not learn from future data that are not accessible at the time of the test instances. By recognizing this, we emphasize the need for designing evaluation frameworks that prevent data leakage and provide an accurate assessment of recommendation model performance.

Finding 2. *With increasing number of "future interactions" included in training, the top-N recommendation lists observe variations.*

In the previous set of experiments, we conducted tuning for each run and utilized the optimal hyperparameters specific to each recommendation. However, we recognized that hyperparameters can introduce confounding effect that affects recommendation performance. Therefore, in this new set of experiments, we aim to eliminate this confounding factor by employing fixed hyperparameters throughout.[9]

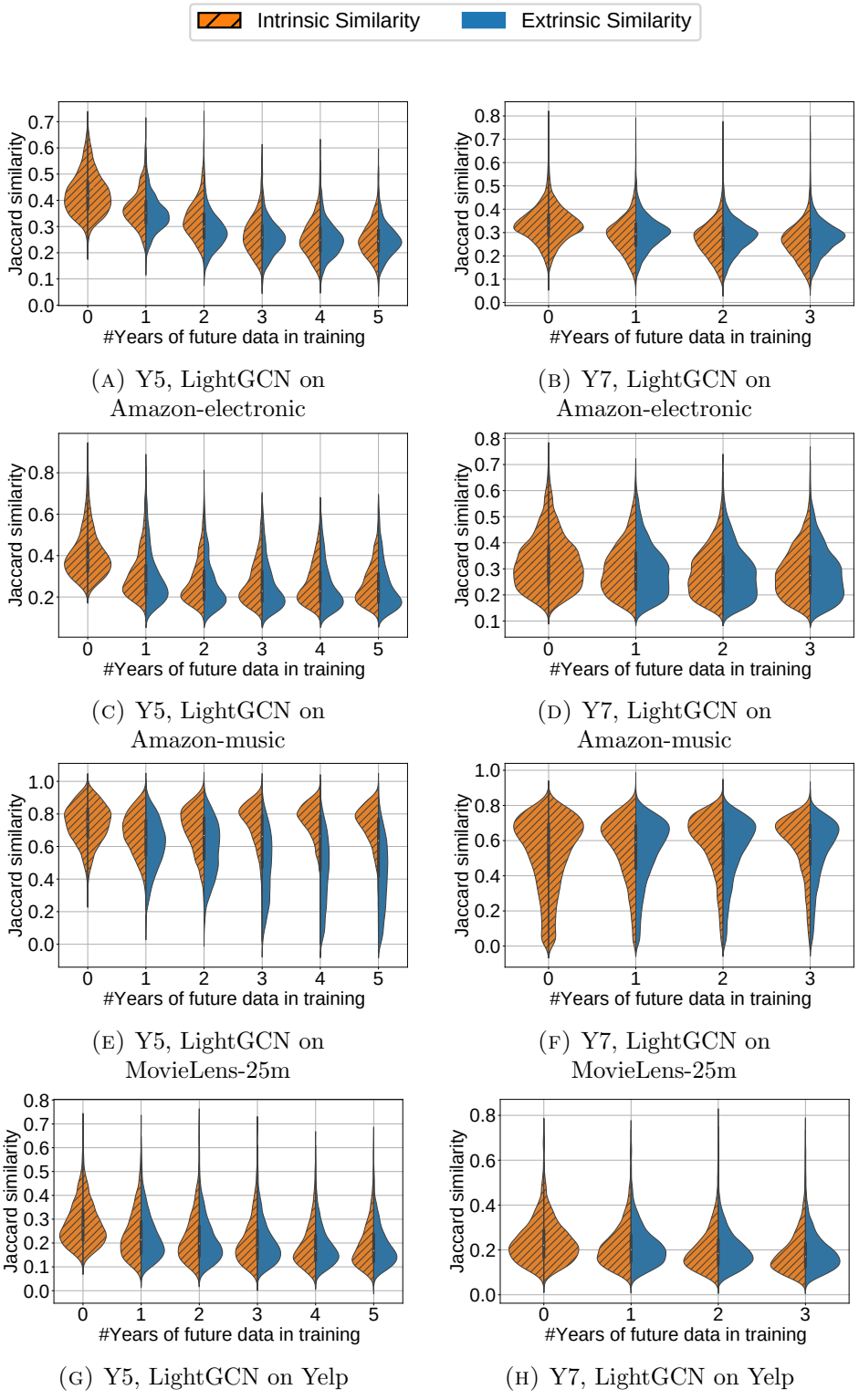
For this purpose, we select a dataset, a baseline model, and a test year (such as $Y5$ or $Y7$) and perform multiple runs of experiments with varying amounts of "future data" as specified in Table 3.3. Rather than tuning the hyperparameters for each individual run, we now keep the hyperparameters fixed at the optimal values obtained from the experiment that has no data leakage. By utilizing fixed hyperparameters, we can compare the top-N recommendation sets obtained from different experiments and assess the impact of data leakage. This approach allows us to isolate the effects of data leakage without the interference of varying hyperparameters. The comparison between top-N recommendation sets can be done by:

$$J(L_A, L_B) = \frac{|L_A \cap L_B|}{|L_A \cup L_B|}$$

where L_A and L_B are two recommendation lists obtained from two experiments, and $J(L_A, L_B)$ refers to the Jaccard similarity between L_A and L_B. Note that, a lower similarity score indicates that data leakage has a greater impact on the training data distribution.

In every experiment, we perform seven random iterations using different seeds. Thus, for each test instance within each experiment, we obtain seven sets of the top-N recommendations. Then, we compare pairwise the seven recommendation sets generated for a test instance when there is no data leakage, along with the seven sets obtained using x years of "future training data". As a result, we obtain $(7 \times 7) = 49$ similarity values for each test instance. These similarity values are then averaged to mitigate random errors. Figures 3.7 and 3.8 illustrates the distributions of similarity scores for all test instances in each experiment.

[9]Experiments are conducted with SASRec and LightGCN.



(A) Y5, LightGCN on Amazon-electronic

(B) Y7, LightGCN on Amazon-electronic

(C) Y5, LightGCN on Amazon-music

(D) Y7, LightGCN on Amazon-music

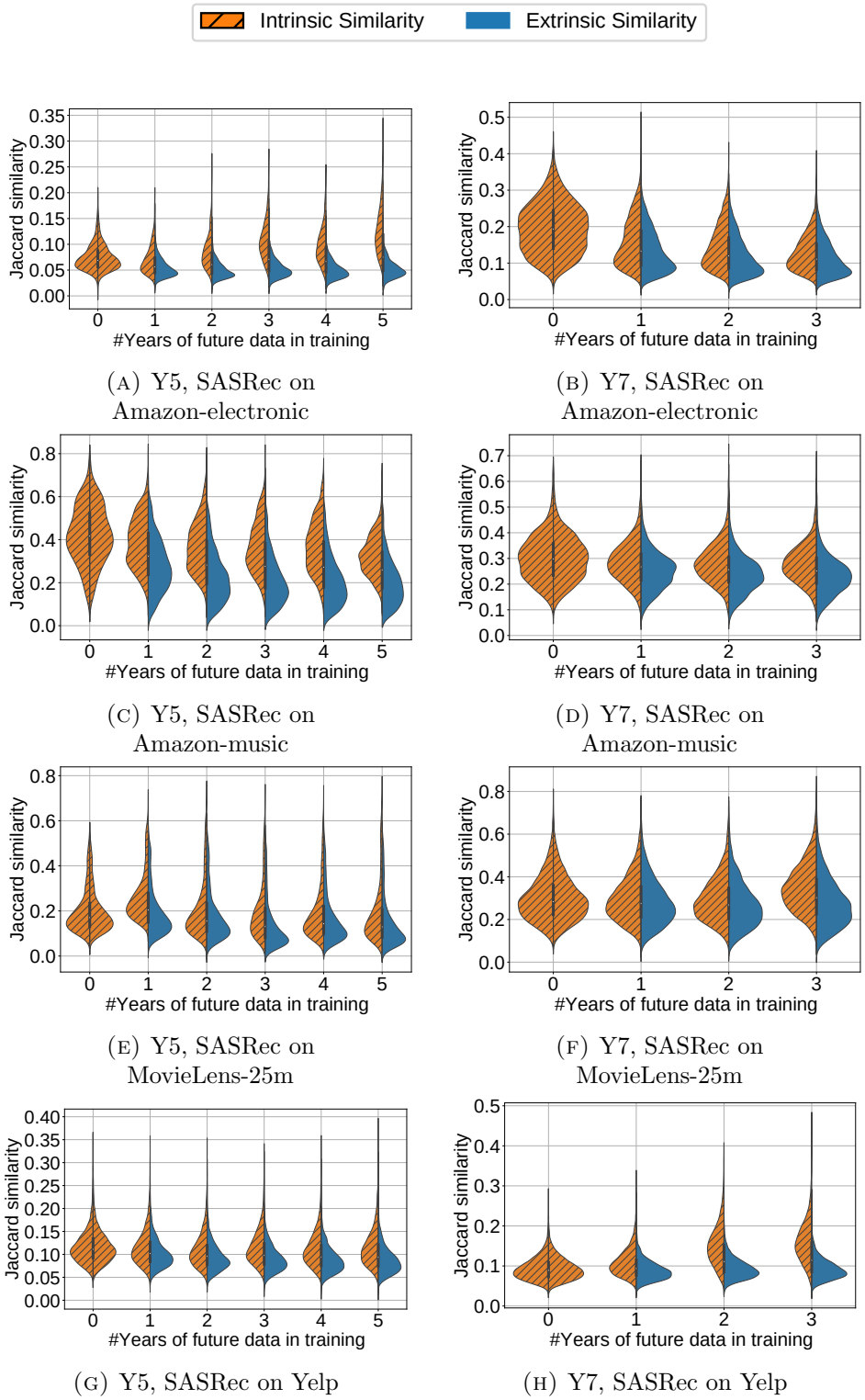
(E) Y5, LightGCN on MovieLens-25m

(F) Y7, LightGCN on MovieLens-25m

(G) Y5, LightGCN on Yelp

(H) Y7, LightGCN on Yelp

FIGURE 3.7: The intrinsic Jaccard similarity and extrinsic Jaccard similarity distributions in experiments with LightGCN.



(A) Y5, SASRec on Amazon-electronic

(B) Y7, SASRec on Amazon-electronic

(C) Y5, SASRec on Amazon-music

(D) Y7, SASRec on Amazon-music

(E) Y5, SASRec on MovieLens-25m

(F) Y7, SASRec on MovieLens-25m

(G) Y5, SASRec on Yelp

(H) Y7, SASRec on Yelp

FIGURE 3.8: The intrinsic Jaccard similarity and extrinsic Jaccard similarity distributions in experiments with SASRec.

In Figures 3.7 and 3.8, we refer to the similarity obtained from the aforementioned comparisons as "extrinsic similarity" because these comparisons involve experiments with different training data, one with data leakage and the other without. In contrast, we also calculate the "intrinsic similarity" which accounts for the randomness resulting from different random seeds used in each experiment with the same training data. Within seven sets of top-20 recommendations for each test instance in each experiment, we can now perform $(7 \times 6)/2 = 21$ pairwise comparisons, resulting in 21 intrinsic similarity scores for each test instance. These intrinsic similarity values are then averaged for each test instance, and their distributions are depicted in Figures 3.7 and 3.8 as the intrinsic similarity distribution for all test instances.

Intrinsic similarity serves as a benchmark for evaluating extrinsic similarity. The absolute value of extrinsic similarity alone lacks meaningful interpretation due to the inherent randomness in the model training process. Therefore, we assess extrinsic similarity by comparing it with the corresponding intrinsic similarity. If the extrinsic similarity distribution significantly differs from its corresponding intrinsic similarity distribution, it indicates that the top-20 recommendations are influenced by the presence of "future training data". This comparison helps us understand the impact of incorporating such data on the recommendation outcomes.

From Figures 3.7 and 3.8, we made three observations. Firstly, as more "future data" is incorporated into the training of recommendation models, the extrinsic similarity tends to decrease. This means that the top-20 recommendation lists increasingly differ from the lists obtained from the experiment without data leakage. This finding provides strong evidence that the models are influenced by the altered user-item interaction distributions caused by the inclusion of "future data" in the training set. Consequently, the training set with "future data" fails to accurately represent the historical context in which the test instances occurred. Notably, MovieLens-25M is more impacted by "future data" compared to Amazon-electronic. This disparity could be attributed to the short time period of user activity in the MovieLens dataset and the unreliable timestamps present in MovieLens. Consequently, the "future data" introduces substantially different contexts from the users' previous interactions. Additionally, variations in the recommendation models employed on the two datasets may contribute to the differential impact.

Secondly, the intrinsic similarity distributions in experiments with "future data" differ from the intrinsic similarity distribution in the 0-year future data experiment. This suggests that the presence of "future data" does indeed affect the recommended items for each test instance, further supporting the conclusion that the context of "history" is influenced by the inclusion of "future data".

Thirdly, as more "future data" is added to the training, the extrinsic similarity becomes increasingly distinct from the intrinsic similarity. This observation implies that the differences in extrinsic similarity are not solely due to random variations in the model training processes but rather due to the presence of data leakage.

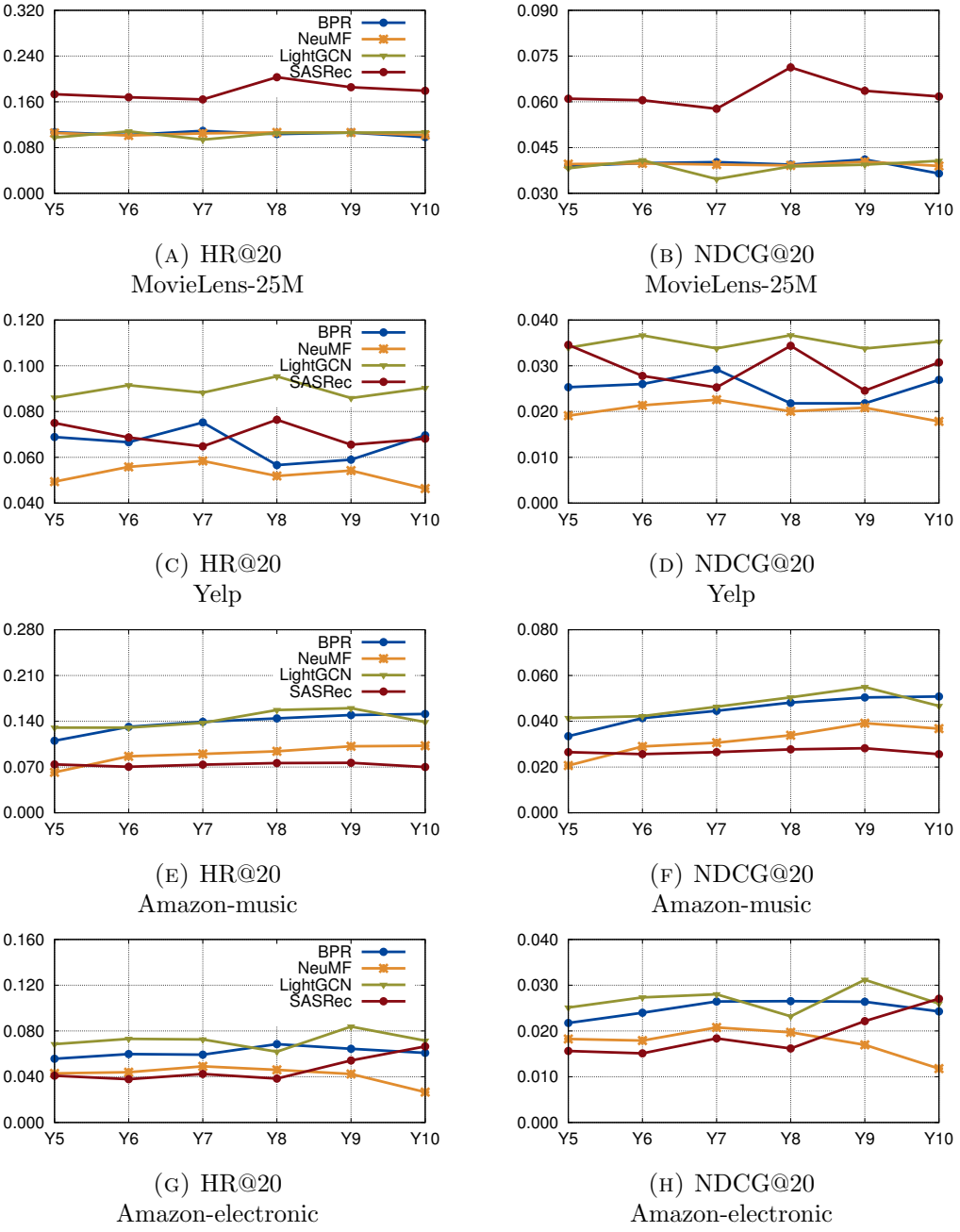
Recalling the problem definition of recommendation discussed in Section 3.1, which involves learning users' preferences from historical data to predict their future interactions, we demonstrate that the inclusion of "future data" inadequately modifies the context of the "history" and, consequently, violates the recommendation problem definition, leading to an impact on the recommendation results.

3.4.2 Impact on Recommendation Accuracy

Finding 3. *The impact of incorporating "future data" on recommendation accuracy can vary, and it is not predictable.*

Figure 3.9 depicts the HR@20 and NDCG@20 values for BPR, NeuMF, SASRec, and LightGCN on the four datasets, with test year $Y5$ selected. These values represent the average results from three random trials of the same experiments conducted with different seed numbers. By averaging the results, we aim to mitigate the potential impact of random factors and reduce noise in the measurements. It is important to note that we utilize non-sampled metrics, as explained in Section 3.3 of the paper. Consequently, the reported HR@20 and NDCG@20 values in Figure 3.9 may appear lower compared to values reported in other papers (such as [13] and [182]) that employ sampled metrics.

Analyzing Figure 3.9, we notice fluctuating patterns for all four models when additional "future data" is incorporated into the training, specifically from $Y6$ to $Y10$. The HR and NDCG values exhibit ups and downs on the MovieLens-25M and Yelp datasets. Similarly, fluctuating patterns can be observed for LightGCN on the Amazon-music dataset and for BPR and LightGCN on the Amazon-electronic



(A) HR@20
MovieLens-25M

(B) NDCG@20
MovieLens-25M

(C) HR@20
Yelp

(D) NDCG@20
Yelp

(E) HR@20
Amazon-music

(F) NDCG@20
Amazon-music

(G) HR@20
Amazon-electronic

(H) NDCG@20
Amazon-electronic

FIGURE 3.9: The HR@20 and NDCG@20 metrics are evaluated for BPR, NeuMF, SASRec, and LightGCN on the test year $Y5$, while considering the progressive addition of future data starting from $Y6$ and continuing up to $Y10$.

TABLE 3.5: Given test year $Y5$, the lowest and highest performance changes (in percentage) when having more future data from the $Y6$ till $Y10$, using the result of no data leakage as reference. The lowest changes are presented in the left column, while the highest changes are presented in the right column.

Dataset	Metric	BPR	NeuMF	LightGCN	SASRec
MovieLens -25M	HR@20	−8.0%, +2.3%	−4.1%, +0.9%	−3.8%, +11.1%	−5.3%, +17.2%
	NDCG@20	−6.3%, +5.5%	−1.5%, +2.0%	−9.3%, +6.8%	−5.4%, +16.8%
Yelp	HR@20	−17.8%, +9.2%	−6.1%, +18.3%	−0.3%, +10.8%	−13.6%, +1.9%
	NDCG@20	−13.9%, +15.4%	−6.6%, +18.3%	−0.5%, +8.0%	−29.0%, −0.6%
Amazon -music	HR@20	+19.3%, +37.2%	+39.6%, +65.6%	0%, +22.8%	−5.4%, +3.3%
	NDCG@20	+23.6%, +51.8%	+40.2%, +89.5%	+1.9%, +32.7%	−3.4%, +6.3%
Amazon -electronic	HR@20	+6.4%, +22.9%	−38.1%, +14.3%	−9.7%, +22.4%	−7.5%, +62.5%
	NDCG@20	+10.3%, +22.0%	−35.5%, +13.8%	−7.7%, +24.1%	−3.3%, +73.0%

dataset. Additionally, a decreasing trend is observed for NeuMF on Amazon-electronic, while an increasing pattern is seen for BPR, NeuMF, and SASRec on Amazon-electronic. In contrast, we observe minimal changes in HR@20 and NDCG@20 for SASRec on the Amazon-music dataset, which is the smallest dataset among the four considered. These fluctuations and trends indicate the varying impact of incorporating "future data" on the performance of different models across different datasets.

In summary, the inclusion of more "future training instances" does not consistently result in improved or degraded recommendation accuracy, but it does have an impact on the values. Consequently, the presence of "future data" in the training set relative to the test instances introduces unpredictability into the recommendation results. This observation indicates that the recommendation performance becomes unpredictable when data leakage occurs. Additionally, it is worth noting that the experiments conducted on test year $Y7$ yield similar results to those obtained on test year $Y5$, further emphasizing that data leakage leads to unpredictable recommendation performance.

Table 3.5 provides a summary of the unpredictability of recommendation accuracy. In this table, the reference results are based on the absence of any future data in the training set. We report the lowest and highest changes in performance observed when future data is added until test year $Y10$. Taking the example of test year $Y5$ and HR@20, we compute the percentage changes in HR@20 when future data is added from $Y6$ to $Y10$, and present the lowest and highest percentage changes.

As shown in Table 3.5, compared to the results without data leakage, the performance changes exhibit significant variations in both positive and negative directions. The magnitudes of these changes can reach as high as 89.5%, and no specific pattern emerges. In summary, the impact of data leakage on recommendation results is unpredictable, and the reported performance from experiments incorporating "future data" does not accurately reflect the performance in a practical setting without data leakage.

Finding 4. *The relative performance ordering of the evaluated models does not exhibit consistent patterns when additional future data is included.*

When additional "future data" is incorporated during training, the ranking order of the four baselines (BPR, NeuMF, SASRec, and LightGCN) in Figure 3.9 undergoes changes. Specifically, the rankings shift from Y6 to Y10. The summary of these ranking orders is presented in Table 3.6, where a value of 1 signifies the best performance (highest HR@20), while 4 indicates the weakest among the four baseline models.

In Table 3.6, it is evident that there are inconsistent ranking orders among the four baseline models when different amounts of "future data" are added for training. The behavior is observed across multiple datasets.

For example, on the MovieLens-25M dataset, the sequential recommender SASRec consistently performs the best. However, the ranking orders of the other recommenders (BPR, NeuMF, and LightGCN) frequently interchange, making it difficult to determine which general recommender would provide better recommendation results. Similarly, on the Yelp dataset, LightGCN consistently performs the best, while NeuMF consistently performs the worst, regardless of the amount of "future data" added. However, the relative performance ordering between BPR and SASRec is inconsistent and unpredictable. The inconsistent ranking orders of the four baseline models are also observed on both the Amazon-music and Amazon-electronic datasets.

It is important to note that all models are tested on the same set of test instances from $Y5$, and the only difference lies in the inclusion of future data (from $Y6$ to $Y10$) that are not supposed to be available in $Y5$. Due to these inconsistent patterns, it is challenging to determine which model would generally yield better recommendation results.

TABLE 3.6: The ranking order of BPR, NeuMF, SASRec, and LightGCN in terms of HR@20 for the test year $Y5$. Here, a ranking of 1 indicates the model with the highest HR@20 and the best recommendation performance, while a ranking of 4 corresponds to the model with the lowest HR@20 among the four baselines.

Dataset	Train Year	BPR	NeuMF	SASRec	LightGCN
MovieLens-25M	$Y5$	2	3	1	4
	$Y6$	3	4	1	2
	$Y7$	2	3	1	4
	$Y8$	4	2	1	3
	$Y9$	3	2	1	4
	$Y10$	4	3	1	2
Yelp	$Y5$	3	4	2	1
	$Y6$	3	4	2	1
	$Y7$	2	4	3	1
	$Y8$	3	4	2	1
	$Y9$	3	4	2	1
	$Y10$	2	4	3	1
Amazon-music	$Y5$	2	4	3	1
	$Y6$	1	3	4	2
	$Y7$	1	3	4	2
	$Y8$	2	3	4	1
	$Y9$	2	3	4	1
	$Y10$	1	3	4	2
Amazon-electronic	$Y5$	2	3	4	1
	$Y6$	2	3	4	1
	$Y7$	2	3	4	1
	$Y8$	1	3	4	2
	$Y9$	2	4	3	1
	$Y10$	3	4	2	1

3.5 Summary

This chapter presents a critical analysis of data leakage in the offline evaluation of recommender systems. The main focus is on considering the global timeline in the evaluation process. We examine the temporal dynamics of users and items by analyzing their average active time across four major datasets. We emphasize that users' last interactions can occur at any time, and items can be released at any point along the global timeline.

In collaborative filtering, if the train/test data split does not take the global timeline into account and all training instances are presented to the recommender as a

whole, the model can learn from data instances that are not available at the time of the test instance. Through meticulously designed experiments, we demonstrate that models with data leakage recommend future items that are not yet available to the system at the time of the test instance. Additionally, we show that incorporating more future data leads to different recommendation lists compared to the model without data leakage, specifically for the test instances.

By evaluating recommendation accuracy measures, we reveal that the impact of data leakage is unpredictable, meaning that the reported results in the literature may not accurately reflect the performance of recommendation models in an online setting. Data leakage also affects the relative ranking order of the evaluated models in terms of performance. Consequently, it becomes challenging to draw conclusions from existing literature regarding which models are more likely to provide better recommendation results in an online setting.

Based on our comprehension of the problem definition in recommender systems and the insights gained from this critical analysis, we propose to conduct a more realistic evaluation of recommender systems in an offline setting. The key idea is to follow the global timeline and conduct train-test split using a “time point separation scheme”.

Chapter 4

Modelling Temporal Context Along The Global Timeline

4.1 Introduction

In the previous chapter, we have demonstrated that neglecting the global timeline leads to the issue of data leakage. This data leakage hinders the effective modeling of the users' preference along the global timeline. In this chapter, we aim to study this issue from perspective of temporal context along the global timeline.[1]

Detecting and capturing the temporal context along the global timeline is challenging for several reasons. Firstly, the temporal context is dynamic and can vary in duration based on external factors such as important events occurring along the global timeline. For instance, in the film industry, a highly popular movie like *AVENGERS* can dominate the market and attract a substantial number of consumers. Similarly, in the mobile phone industry, the release dates of new products significantly impact interaction trends. Consequently, domain knowledge is necessary to appropriately delineate the temporal context within the global timeline.[2]

[1]This chapter is published as Yitong Ji, Aixin Sun, Jie Zhang and Chenliang Li. A Critical Study on Data Leakage in Recommender System Offline Evaluation. ACM Trans. Inf. Syst. 41, 3, Article 75 (July 2023), 27 pages. [183]

[2]This chapter also includes content from the publication - Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2022. Do Loyal Users Enjoy Better Recommendations? Understanding Recommender Accuracy from a Time Perspective. In Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '22). Association for Computing Machinery, New York, NY, USA, 92–97. [184]

In this chapter, we visualize the temporal context using items' popularity. Note that, the popularity based recommender, MostPop, is a widely used baseline in academic research of recommender system.

4.1.1 Popularity Baseline in Recommender System

The popularity of items can vary over time. We demonstrate this phenomenon in Figure 4.1, where we present three example items: A, B, and C, along with their respective number of interactions over time. Suppose a user interacts with the system at times t_1, t_2, and t_3. In that case, the "most popular" items at those specific times would be A, B, and C, respectively, assuming that the system only consists of three items.

In order to gain insights into the popularity based recommender (*i.e.,* MostPop) in recommendation tasks, we conduct a study by sampling 12 papers from top-tier conferences such as KDD, WWW, SIGIR, and RecSys. Additionally, we also examined 6 open-source toolkits. Several studies define the MostPop baseline as a non-personalized method where popular items are recommended. This definition is often provided briefly without further elaboration [180, 185–189]. Other works highlight that popularity can be measured based on the number of explicit or implicit feedback on items [6, 13, 190–192]. Additionally, we observe that in several open-source recommendation tools, including LibRec [193], SequenceBasedRecommender [194], RecommenderLab [195], CaseRecommender [196], Microsoft Recommender [197], and TagRec [198], popularity is defined based on the "number of interactions in the training data".

In summary, many studies define and evaluate popularity based on the overall popularity of items in the training set. This means that the recommended items are those with the highest number of interactions in the training data. However, depending on how the offline data is partitioned into training and test sets, we argue that in many existing evaluations, the MostPop baseline does not accurately reflect the common understanding of popularity. We illustrate this point using *leave-one-out* split which treats the last interaction of each user as a test instance, and considers the remaining interactions as training instances. Referring to Figure 4.1, t_1 is taken as the last time point when user u_1 interacts with the system. t_2 and t_3 are considered as the last time points of interactions from u_2 and u_3, respectively.

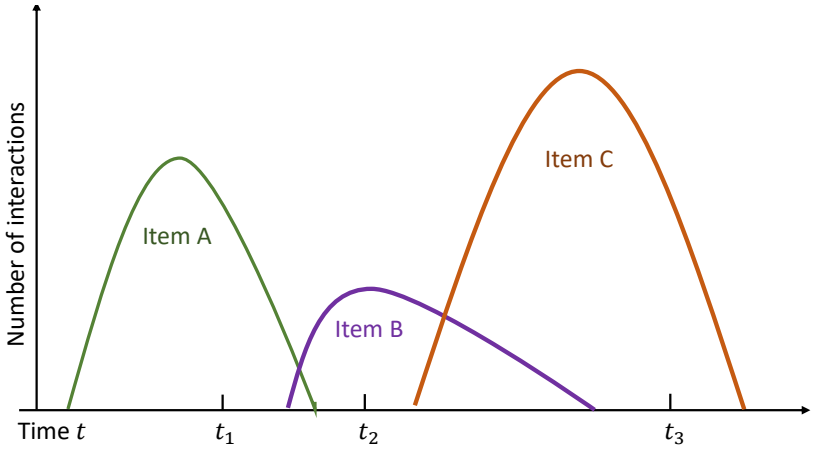


FIGURE 4.1: Popularity of items A, B, and C over time.

Regardless of the popularity of items B and C, u_1 has never had the opportunity to access these items because they were not released yet at time t_1. However, according the mainstream MostPop definition, the MostPop model simply ranks items based on the accumulated user interactions in the training set. And the popularity rank becomes C, A and B in descending order. As a result, items that are released after a user's last interaction can be recommended to that user, which is not feasible in reality. This illustration shows that temporal context along the global timeline cannot be well modelled, due to the presence of leaked future interactions in the training set.

To prove the importance of considering temporal context in recommendation, we performed experiments on the MovieLens-20m dataset, which includes interaction data spanning 20 years from January 9, 1995, to March 31, 2015. Then, we utilize a leave-one-out approach, where we set aside the last interaction of each user as the test data and used the remaining interactions for training, following the commonly used leave-one-out splitting method. To replicate a scenario where a system needs considerable time to accumulate user interactions, we evaluated the models using test instances that occurred within the last five years (from March 31, 2010, to March 31, 2015). Our evaluation consisted of 29,431 test instances. We employed the Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG) as evaluation metrics. We compared three "popularity" models in our analysis.

- **MostPop**. It is commonly used in research papers and toolkits. MostPop ranks items based on the number of interactions they receive in the entire training set.

TABLE 4.1: Results of popularity based recommendation model; best results in bold

Popularity	HR@5	HR@10	NDCG@5	NDCG@10
MostPop	0.0304	0.0462	0.0198	0.0248
RecentPop	0.0530	**0.0845**	0.0338	0.0440
DecayPop	**0.0532**	0.0843	**0.0341**	**0.0441**

- **RecentPop**. RecentPop is a recommendation approach specifically designed to suggest popular movies to a user at the time of their interaction with the system. This time point is denoted as t_0, which represents the timestamp of the user's last interaction during the testing phase. To determine the most popular movies at t_0, we take into account that a movie's popularity can change over time. Therefore, we rank the movies based on the number of ratings they received within a short time period, specifically $[t_0 - \Delta t, t_0]$. In our evaluation, we set Δt to be 1 month. This allows us to capture the movies that have been popular within that recent time frame.

- **DecayPop**. DecayPop is an extension of the RecentPop approach, designed to determine popular movies over a longer time period leading up to t_0. In DecayPop, we consider the past 6 months before t_0 and calculate a weighted sum of the number of ratings received by movies during that period. The weight assigned to each month's ratings is determined using an exponential decay function, $e^{(-t_m)}$. Here, t_m represents the number of months relative to t_0, ranging from 1 (for the most recent month) to 6 (for the oldest month). By applying this exponential decay, we assign higher weight to the more recent interactions of movies, reflecting the notion that the recent ratings are more indicative of the current popularity.

Both RecentPop and DecayPop, similar to MostPop, are non-personalized methods. The key distinction is that RecentPop and DecayPop consider the time point when a user interacts with the system and derive the most popular items specifically at that time. In Table 4.1, we can observe the recommendation accuracy of the three popularity methods. Two noteworthy observations can be made. Firstly, RecentPop demonstrates a significant improvement over MostPop across all evaluation metrics, with enhancements ranging from 70% to 83%. Secondly, DecayPop

achieves a slightly higher recommendation accuracy compared to RecentPop and performs best on three out of four measures.

By incorporating the time dimension into the evaluation, both RecentPop and DecayPop yield substantial improvements over MostPop. These new definitions of popularity are straightforward to implement and can be customized by adjusting parameters such as Δt and the weighting function. We argue that both RecentPop and DecayPop models the temporal context better and thus achieve better recommendation. More importantly, this experiment further demonstrates that data leakage could result in failure to model the correct temporal context when a user interacts with an item.

4.2 User Behaviors vs Temporal Context

Most datasets commonly used in academic research of recommender systems lack explicit information regarding the temporal context. Furthermore, extracting relevant information that could be utilized to derive the temporal context, such as special promotional events, is a challenging task. As a result, establishing a clear connection between user interactions and the temporal context becomes difficult, thereby impeding the ability to model user preferences in consideration of the temporal context effectively.

In many existing studies, a training set is fed to a recommendation model to learn user preference without considering the temporal context along the global timeline. Hence, the objective of this chapter is to investigate the relationship between user interactions and the temporal context. Specifically, we set aside a time frame that is of a short duration. By selecting a limited time interval, we assume that the temporal context remains relatively consistent during this period. Further, we analyze the influence of user interaction-related factors on predicting interactions that occur within the specified duration. By examining these associations, we aim to gain insights into how user behaviors affect the prediction of interactions that happen within a predefined time frame.

Assuming that the test instance of a user happen at time t, the three user interaction related factors of him/her can be defined according to Figure 4.2. In particular, we have (i) **number of interactions** that the user has until time t;

FIGURE 4.2: User interactions related factors considered in analysis

TABLE 4.2: The statistics provided below pertain to the four datasets, each capturing interactions that occurred within a 10-year period starting from their respective starting times.

Dataset	Starting time	Data filtering	#User	#Item	#Data instances
MovieLens-25M	21 Nov 2009	No filtering	62,202	56,774	9,808,925
Yelp	13 Dec 2009	10-core	116,655	61,027	3,127,215
Amazon-music	02 Oct 2008	5-core	11,651	9,243	114,833
Amazon-electronic	05 Oct 2008	10-core	109,990	39,552	1,752,238

(ii) **active time period (ATP)** which specifies the time duration between the user's first interaction in the system and the time of the test instance (*i.e.,* time t); and (iii) **recency** defined by the time interval between time t and the user's latest interaction just before time t. The first two factors determine the extent that the user is engaged with the system.

4.2.1 Experiment

This section provides an in-depth description of the datasets used, the recommendation models evaluated, and the experimental setup employed for our study.

4.2.1.1 Datasets and Evaluated Models

Our experimentation encompasses four publicly available datasets, namely MovieLens-25M, Yelp, Amazon-music, and Amazon-electronic. All interactions within these datasets are accompanied by timestamps. To ensure consistency and relevance, we extract interactions that occurred within a specific 10-year time period for our experiments (refer to Table 4.2 for the starting time of each dataset). Additionally, we apply a filtering process to eliminate exceedingly inactive users or items.

For Yelp, Amazon-music, and Amazon-electronic datasets, we employ k-core filtering. This technique ensures that all users and items included in the dataset possess a minimum of k interactions. To accommodate the dataset's size and sparsity, we opt for a 10-core filtering strategy for Yelp and Amazon-electronic, while selecting a 5-core filtering approach for Amazon-music. Note that, MovieLens-25M dataset does not undergo any filtering process as it already guarantees that each user has a minimum of 20 ratings. The statistics of the four datasets after the filtering procedure are presented in Table 4.2. By implementing these filtering methods, we can focus on a core subset of users and items, thus enhancing the effectiveness of our analysis.

Our findings are based on the recommendation results generated by five widely used baseline models: BPR [174], NeuMF [13], LightGCN [49], SASRec [89], and TiSASRec [90]. Each baseline model represents a distinct type of recommendation approach. BPR, NeuMF, and LightGCN are general recommender models that do not explicitly incorporate temporal information. BPR utilizes pairwise ranking loss within matrix factorization to learn latent factors of users and items. NeuMF combines matrix factorization and multi-layer perceptron to learn the user and item interaction function. LightGCN is a graph-based recommender model that leverages graph convolutional networks to capture complex relationships between users and items. On the other hand, SASRec is a time-aware model that considers the temporal order of interactions. SASRec employs a self-attentive network to model sequential patterns in user behavior. The first four baselines follow the same baselines in Chapter 3. We further introduce TiSASRec in this chapter. It is built on top of SASRec, but taking into account the time intervals between events as input, enhancing the temporal-awareness of the model. These five baseline models encompass various recommendation approaches, enabling a comprehensive evaluation of their performance and the impact of user interactions related factors in our study.

4.2.1.2 Evaluation Setting

Chapter 3 highlights the significant issue of data leakage in the offline evaluation of recommender systems. Specifically, data partition strategies such as leave-one-out and random-split-by-ratio, which do not adhere to the global timeline, can lead to the training of recommendation models using future training instances. Future

training instances refer to interactions that occur after the time point of a test instance.

While we acknowledge that using leave-one-out-split could be problematic, we adopt one for experiment in this chapter. This means that for each user, their last interactions are treated as the test instances, while the remaining interactions are utilized as training instances. The decision to employ leave-one-out split is twofold: it is a widely adopted data partitioning strategy in recommender systems, and it enables recommendation models to have a comprehensive understanding of a user's historical interactions by excluding only their last interaction from the training set. Once the training and test sets are obtained, we follow the methodology outlined in Chapter 3 to evaluate the performance of the models on test instances that occur within a specific year.

Same as in Chapter 3, each dataset consists of interactions spanning a 10-year period. We specifically select test instances that occur within a particular year for evaluation purposes. Let's consider year 10 (denoted as $Y10$) as an example. In this scenario, test instances that took place in $Y10$ are included in the test set for that specific experiment run. All instances preceding $Y10$ (*i.e.*,, $Y1$ to $Y9$), along with the training instances within $Y10$, are used for training the models. Despite the presence of a data leakage issue with the leave-one-out split, we mitigate it to a minimal extent by restricting it solely to training instances in $Y10$. Simultaneously, we leverage the advantage offered by the leave-one-out split, namely, gaining a comprehensive view of a user's interaction history.

Following this experimental setup, we perform experiments on each of the four datasets, using $Y10$ as the test year. For each model, we conduct continuous random search to tune their hyperparameters during each run of the experiment. To ensure consistency with previous research [178, 179], we perform hyperparameter tuning using a validation set that includes the second last interaction of each user.

Regarding the evaluation metrics, we employ Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG). In our research, we rank all the available items and generate top-N recommendations for evaluation purposes. It is worth noting that we do not utilize sampled metrics since they can introduce bias in measuring recommendation accuracy [160]. By considering all available items and making

TABLE 4.3: Average number of interactions for each user group.

Dataset	Baselines	#Interactions	
		Loyal Users	Non-loyal Users
MovieLens-25M	General Recommenders	505.0	61.4
	SASRec	40.0	28.1
	TiSASRec	36	26.4
Yelp	General Recommenders	48.2	11.8
	SASRec	17.9	11.8
	TiSASRec	32.2	11.8
Amazon-music	General Recommenders	17.7	4.7
	SASRec	16.1	4.7
	TiSASRec	15.6	4.7
Amazon-electronic	General Recommenders	21.8	10.2
	SASRec	21.1	10.2
	TiSASRec	18.5	10.2

recommendations based on their rankings, we aim to provide a comprehensive and unbiased assessment of the performance of the evaluated models.

4.2.2 Experiment Results

We now study the recommendation results to quantify the impact of user interactions related factors on the recommendation accuracy.

4.2.2.1 Number of Interactions

In our data partitioning strategy, known as leave-one-out-split, we divide the dataset into training and test sets. For each user, his/her last interaction is treated as the test instance, while all previous interactions are considered as training instances. The number of interactions a user has in the training set can serve as an indicator of his/her loyalty. To simplify the analysis, we rank users based on the number of interactions they have in the training set. Users who fall within the top 50% are classified as loyal users, while the remaining users are categorized as non-loyal users. The average number of interactions for both loyal and non-loyal users are provided in Table 4.3.

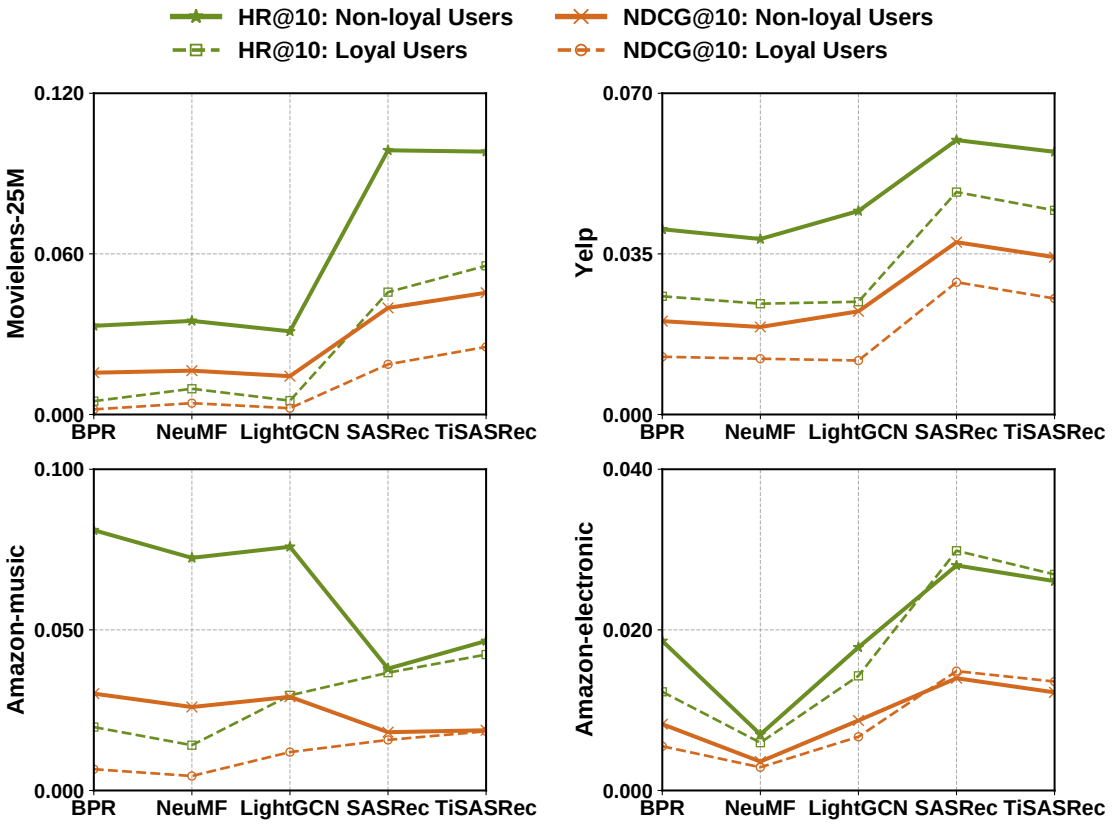


FIGURE 4.3: HR@10 and NDCG@10 of loyal and non-loyal users by **number of interactions**, in test year $Y10$

It's important to note that for the SASRec and TiSASRec models, only the most recent "n" interactions of each user are utilized during training. This is because SASRec and TiSASRec leverage self-attention networks, which require memory that is quadratic in the length of the user's interaction sequence. As mentioned in the original papers of SASRec and TiSASRec, we tune the value of "n" as a hyperparameter. Consequently, Table 4.3 reflects varying average numbers of interactions for loyal and non-loyal users in the case of SASRec and TiSASRec, compared to the general recommender models (BPR, LightGCN and NeuMF).

Figure 4.3 illustrates the recommendation accuracy in terms of HR@10 and NDCG-@10 for loyal and non-loyal users, categorized based on the number of interactions. Across the MovieLens-25M, Yelp, and Amazon-music datasets, all five models demonstrate superior recommendations for non-loyal users when considering HR@10. When it comes to the Amazon-electronic dataset, only SASRec and TiSASRec exhibit better results for loyal users, the general recommenders show

similar trends as in the other datasets. Regarding NDCG@10, non-loyal users generally receive better recommendations compared to loyal users, with the exception of TiSASRec on Amazon-music, as well as SASRec and TiSASRec on Amazon-electronics.

Interestingly, the overall trend suggests that non-loyal users tend to receive better recommendations than loyal users, particularly with the general recommender models. This finding contradicts the common intuition that a larger amount of historical data would lead to better understanding of a user's preference in recommendations. Consequently, we propose a hypothesis that not all historical interactions by a user are equally valuable when making "recent" recommendations. In other words, not all historical interactions of a user provide useful information for user preference prediction at the temporal context that the test instance is in.

4.2.2.2 Active Time Period (ATP)

Merely considering the number of interactions is deemed inadequate. This is because a user can accumulate numerous interactions within a short time frame. To address this limitation, we introduce a new factor called Active Time Period (ATP). We consider the time point of the last interaction of a user as the time point when a recommendation is about to be made for this user. Hence, we define ATP to be the number of days between a user's initial interaction and the recommendation time (the last interaction/test instance in our setting).

Similar to the "number of interactions", ATP can be considered as a loyalty indicator. The loyalty definition threshold is set as the median ATP of users in the training set. Users with ATP higher than the threshold are considered loyal users, due to their long-time engagement with the system. The remaining users are classified as non-loyal users. In table 4.4, we list the average ATP values for loyal and non-loyal users across different models.

Figure 4.4 depicts the HR@10 and NDCG@10 results for loyal and non-loyal users based on their active time period. In general, non-loyal users, characterized by a shorter active time period, tend to receive better HR@10 and NDCG@10 results compared to loyal users. A loyal user, as indicated by the active time period, is an individual who has been interacting with the system for an extended period.

TABLE 4.4: Active time period (days) for each user group.

Dataset	Baselines	ATP (days)	
		Loyal Users	Non-loyal Users
MovieLens-25M	General Recommenders	909.2	3.3
	SASRec	385.9	1.1
	TiSASRec	366.9	1.0
Yelp	General Recommenders	2299.2	964.1
	SASRec	1899.1	622.9
	TiSASRec	2173.8	864.9
Amazon-music	General Recommenders	1592.3	445.0
	SASRec	1578.5	434.2
	TiSASRec	1572.0	429.4
Amazon-electronic	General Recommenders	2208.5	1012.0
	SASRec	2188.9	1003.1
	TiSASRec	2101.1	940.1

On one hand, the system has a greater opportunity to capture the user's long-term interests. On the other hand, interactions that occurred a long time ago may not necessarily reflect the user's current preferences in the current context. Our findings, as shown in Figure 4.4, largely support the latter scenario across multiple datasets. Additionally, we observe that loyal and non-loyal users achieve comparable recommendation accuracy with SASRec and TiSASRec on all four datasets. This is due to the fact that SASRec and TiSASRec are designed to treat recent and old interactions differently for each user.

Our results indicate that the preferences of loyal users may not be accurately predicted, particularly for general recommender models that treat all training instances equally. We hypothesize that this observation is primarily attributed to outdated interactions that occurred a long time ago, which fail to reflect a user's current interests in the latest temporal context. Therefore, this motivates us to further investigate another factor: recency.

4.2.2.3 Recency

Recency, as defined in our research, refers to the time interval in days between a user's test instance and their most recent interaction just before the test instance. To differentiate between active and inactive users based on recency, we rank all

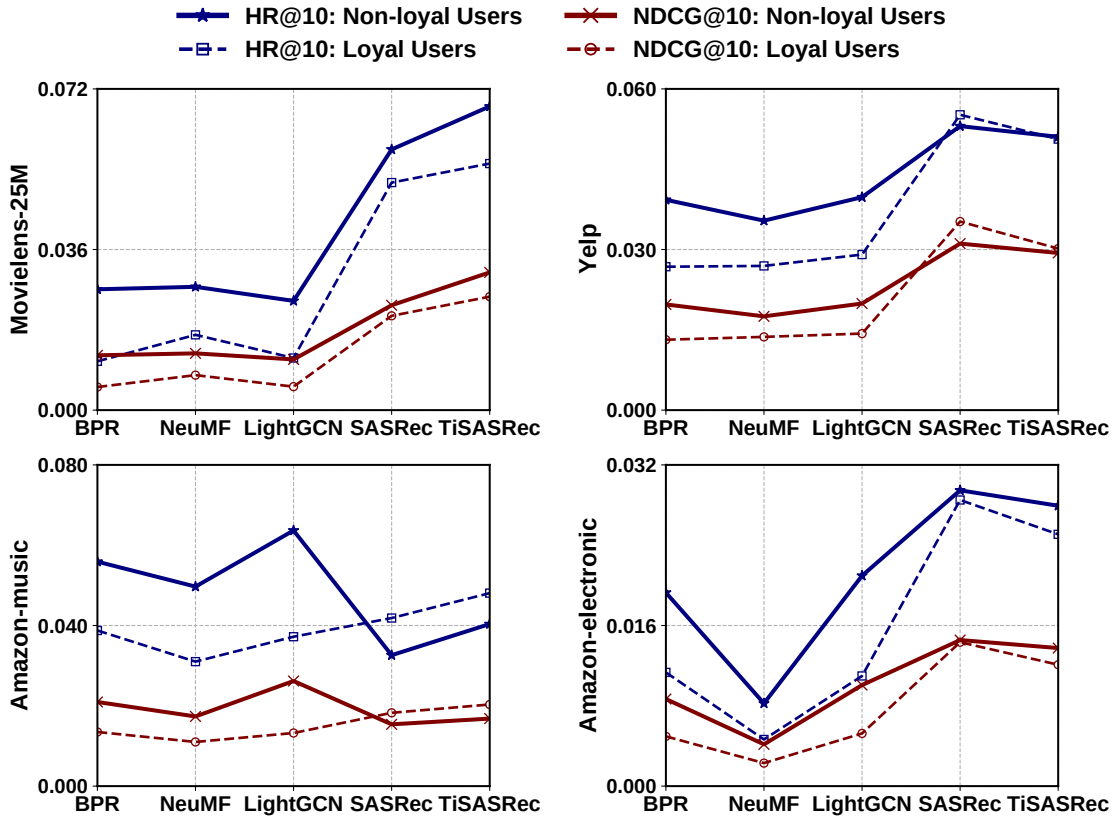


FIGURE 4.4: HR@10 and NDCG@10 of loyal and non-loyal users by **active time period**, in test year $Y10$

TABLE 4.5: Recency (days) for each user group.

Dataset	Recency (days)	
	Loyal Users	Non-loyal Users
MovieLens-25M	$1.6e-4$	32.2
Yelp	12.5	283.0
Amazon-music	1.7	447.8
Amazon-electronic	15.1	373.8

users and use the 50^{th} percentile recency as the threshold. Active users are identified as those with shorter recency values, while the remaining users are categorized as inactive. The statistics about recency factor is shown in Table 4.5

Figure 4.5 presents the results of our evaluation, displaying the HR@10 and NDCG@10 scores for both active and inactive user groups. Across all types of recommenders, encompassing both general and sequential models, we observe that active users consistently receive better recommendations in terms of HR@10 and NDCG@10. These findings strongly suggest that recent interactions hold more

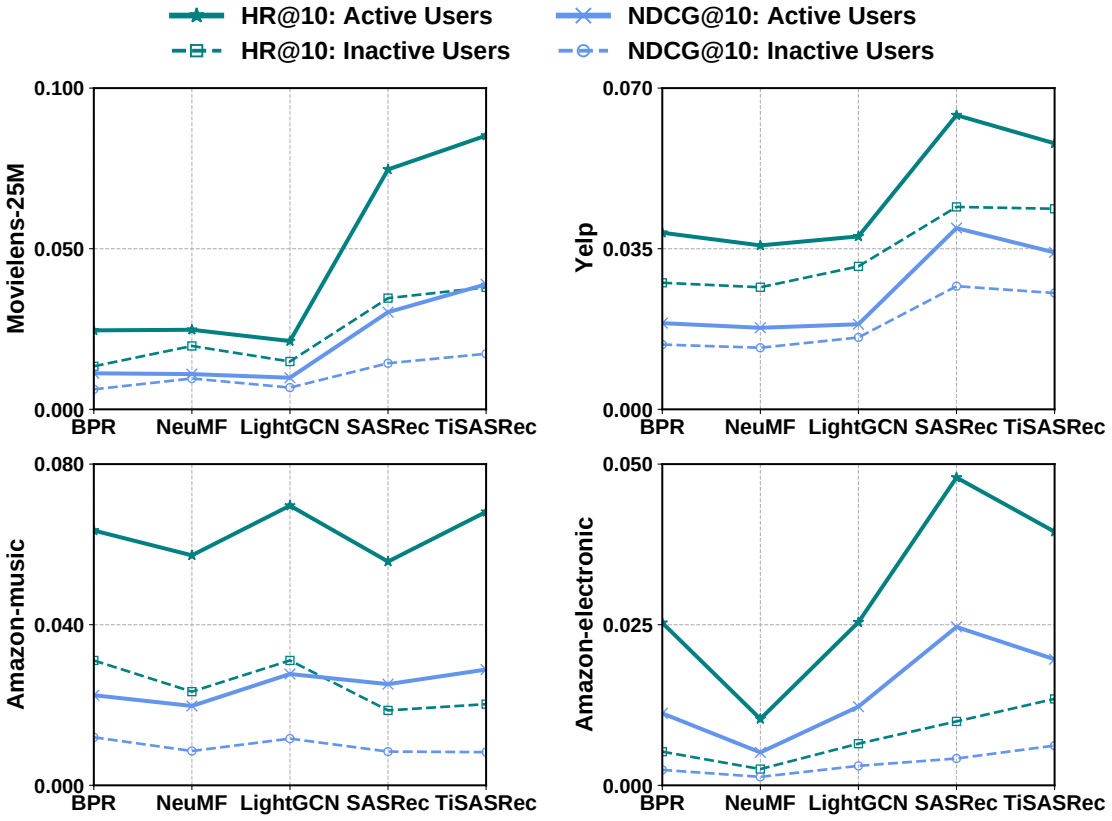


FIGURE 4.5: HR@10 and NDCG@10 for users grouped by **recency** of the latest interaction before test instance.

significance in predicting user preferences. This observation also helps explain the earlier finding that loyal users tend to receive poorer recommendations compared to non-loyal users.

Furthermore, it is interesting to note that the recommenders, regardless of their consideration of time information in their models, exhibit similar trends regarding the influence of recency. Even time-aware models like SASRec and TiSASRec do not explicitly incorporate the recency factor during the recommendation process. Instead, they focus on the user's local timeline (*i.e.*,, their interaction sequence) rather than the global timeline. Based on the insights gained from the impact of recency, we argue that recommenders should dynamically account for recent interactions when making recommendations, taking into consideration the specific time point along the global timeline, to enhance their performance in predicting a user's preference at the latest temporal context.

4.3 Summary

In this chapter, we aim to demonstrate the crucial role of considering the global timeline in capturing the temporal context for effective recommendations. Our findings reveal that neglecting the global timeline can result in the failure to capture the necessary temporal context. We conduct a simple experiment using a popularity baseline, which further confirms that without correctly modeling the temporal context, the optimization of recommendations cannot be achieved. Thus, it emphasizes the significance of adhering to a global timeline.

While it may seem straightforward to follow a global timeline, merely doing so is not always sufficient for adequately modeling the temporal context. Many existing studies overlook the importance of ensuring that the training set comprehensively represents the temporal context. Consequently, in this chapter, we investigate the capability of a training set to effectively model the temporal context. Specifically, we explore the relationship between user behaviors and the temporal context by examining the impact of three factors associated with user behaviors on recommendation performance.

The experimental results obtained illustrate that not all user interactions contribute equally to learning a user's preference. Instead, recent interactions carry more significance, highlighting their crucial role in capturing the temporal context accurately.

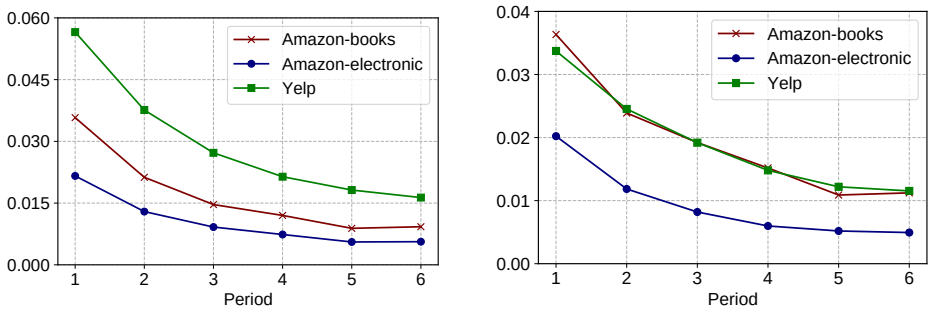
Chapter 5

Incremental Learning for Recommender System

5.1 Introduction

In previous chapters, we have shown that it is essential to evaluate a recommender system following the global timeline to ensure the correct and fair comparison between recommendation baseline models. Moreover, we conduct experiment on popularity baseline to further prove the importance of considering temporal context in recommendation. In addition, experiments that study the relationship between user behaviours and temporal context also proves that it is not necessary to learn a user's preference from all his/her historical behaviours. Instead, the more recent interactions are more important to the user's choice under the recent temporal context. Hence, the more recent interactions should be placed with more weights to model the user's recent interests.

Not only do the users change over time, items in the market undergo changes like popularity and market positioning. In a real-life scenario, as a business expands and attracts more users and items, it becomes essential to continually update the recommender system. This ensures that the recommendation model stays updated with the latest developments, including the addition of new users, items, and emerging trends observed from recent interactions. By updating the model, the business aims to better cater to its users and provide them with effective recommendations.



(A) LightGCN (B) NGCF

FIGURE 5.1: We record the recall@20 over 6 periods in offline evaluation of recommendations on Amazon-books, Amazon-electronic and Yelp, when no retraining has been conducted for LightGCN and NGCF.

The need to update or refresh a recommendation model using the most recent interactions can be supported by Figure 5.1. In the experiment conducted, LightGCN and NGCF models are trained using specific datasets from Amazon-books, Amazon-electronic, and Yelp. Once trained, these models are kept fixed without any further retraining. The fixed models are then used to make recommendations over the course of six periods, where the length of each period varied depending on the dataset: one month for Amazon-books and half a year for Amazon-electronic and Yelp.

Figure 5.1 illustrates that the performance of the recommendations gradually deteriorated over time. This decline can primarily be attributed to the fact that an outdated model, which has not undergone retraining, cannot effectively capture the latest trends in the data. As new interactions occur and user preferences evolve, the recommendation model needs to be regularly updated to adapt and provide accurate and relevant recommendations.

In this thesis, we focus on retraining of the Graph Convolutional Network (GCN)-based recommendation model. The key reason of considering graph-based recommendation is that GCN-based recommenders have been proven to be effective in recommendation, because it captures the high-order relationship between users and items. Moreover, retraining of GCN-based recommendation has been less explored compared to general recommenders, since it is non-trivial to consider the evolving graph structure.

Figure 5.2 demonstrates a typical scenario that the retraining process takes place at the end of specific time windows denoted as t_0, t_1, up to t_n. Initially, an model M_{init}

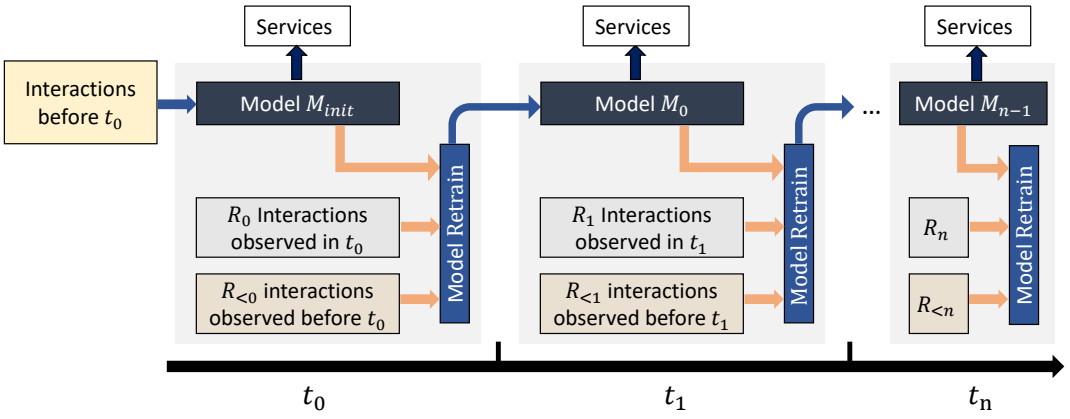


FIGURE 5.2: Model retraining with incremental Learning. The inputs for retraining can be from the latest deployed model, the new interactions and the historical interactions observed prior to the current time window.

is trained using interactions observed before the first time window, t_0. The model M_{init} is then deployed for providing recommendations during the time window t_0. At the end of t_0, the model is retrained, resulting in an updated model M_0. This updated model M_0 is subsequently deployed to provide recommendations during the next time window, t_1. At the end of each time window like t_n, the retraining process incorporates three types of information. First, R_n represents the set of new interactions observed during the time window t_n, capturing the most recent user-item interactions. Second, $R_{<n}$ encompasses all historical interactions observed before t_n. Lastly, M_{n-1} refers to the latest model updated during the previous time window t_{n-1}. The parameters of M_{n-1} reflect the most up-to-date knowledge captured by the recommender system prior to t_n, taking into account the multiple updates performed since the initial model M_{init}.

We would use an example to illustrate the two straightforward retraining strategies. Assuming it is at the end of time window t_n, R_n represents the new interactions observed within t_n and M_{n-1} refers to the latest recommendation model. The simplest retraining strategy, fine-tuning, is to update the latest model M_{n-1} using the newly observed interactions R_n to obtain M_n. However, this strategy can lead to a potential problem known as catastrophic forgetting [125, 199]. It means that the model may forget the long-term preferences of users because it is always focused on capturing the current trends in each update. If the long-term preference like brand loyalty happens to be not reflected in R_n, fine-tuning may eventually abandon this piece of information, which is not desirable. Figure 5.1 further demonstrates the

existence of long-term preference. It shows that although the recall@20 metric decreases over time without retraining the recommendation model, it does not drop to zero even after six periods of no retraining. This suggests that parts of user preferences and item characteristics may remain relatively stable over time. Thus, an old model trained from historical interactions can still make accurate predictions on new interactions. To this end, we argue that it is crucial to preserve historical information expressed in historical interactions to enhance recommendation accuracy.

To effectively utilize both historical and new information, the other strategy is full-retraining. This approach abandons the latest model and rebuilds a new recommendation model from scratch using all interactions (*i.e.*,, $R_{<n} \cup R_n$) seen so far. However, full-retraining poses scalability issues as the training cost increases with the growing number of interactions over time. Moreover, not all historical interactions carry relevant information in a recent context, as some may be outdated or less indicative of current user preferences and trends, as discussed in Chapter 4.

Researchers have explored various strategies to address these challenges and strike a balance between retaining historical information and capturing recent trends. Several existing studies (SML [110], ASMG [120], SPMF [116], GraphSAIL [111], CI [119]) have employed a scalable and effective retraining strategy known as **Incremental Learning**. These studies utilize various techniques to ensure that the model benefits from both newly observed interactions in the current time window and the retention of old information.

To accomplish this, incremental learning strategies employ different approaches to make the model "memorize" or retain old information. One common technique involves sampling a subset of historical interactions to serve as representative examples of long-term information. This approach is adopted in studies such as SPMF [110], SSRM [117], and GAG [118]. Additionally, some studies, including SML [110] and ASMG [120], consider the model parameters (*e.g.*,, user embeddings) learned in the past as a form of historical information. By incorporating these parameters into the retraining process, the model can retain valuable knowledge from previous iterations. By combining newly observed interactions with sampled historical interactions or utilizing previous model parameters, these incremental learning strategies strike a balance between incorporating recent trends and preserving long-term preferences. This enables the recommendation model to

continuously adapt to changing user behavior while leveraging the accumulated knowledge from the past.

In this chapter, we propose a novel incremental learning strategy called Disentangled Incremental Learning (DIL) specifically designed for Graph Convolutional Network (GCN) based recommenders. GCN-based recommenders are chosen for their ability to model complex relationships between users and items, as well as their proven effectiveness in previous studies [49, 50, 52, 200].

DIL addresses the two main challenges in incremental learning: extracting historical information and effectively blending historical and new information in the retrained model. While existing methods often consider the entire node representations from the previous model as historical information, DIL takes a different approach. It recognizes that not all information in the previous node representations is relevant for long-lasting attributes. Additionally, DIL treats information from neighbors at different hops separately, as different hops contribute differently to historical information. To tackle this, DIL introduces an Information Extraction Module (IEM) that dynamically adjusts what and how much historical information to extract from the previous model. Rather than extracting predefined information, the IEM learns weights to facilitate the extraction process.

To blend new and old information effectively, DIL proposes a Disentanglement Module (DM) that treats the two types of information differently during model retraining. The DM ensures that the old and new information are distinct, reflecting their respective semantics. However, both pieces of information work together through GCN learning to achieve accurate recommendations.

This chapter contributes threefold. Firstly, it introduces the IEM as an approach to extract historical information from the previous model, providing two implementation designs. Secondly, it introduces the disentanglement of historical and new information to effectively capture both types of information. Notably, this is the first work to disentangle long-term preferences and short-term preferences in recommendation model retraining. Lastly, it redesigns the embedding layer in GCN-based recommenders for model retraining, incorporating historical information from the IEM and new information to be learned from the current interaction graph. This enables information exchange among adjacent nodes and indirect

neighbors through GCN message propagation, resulting in more accurate recommendations.

DIL is evaluated on two base models, NGCF [50] and LightGCN [49], using three benchmark datasets: Amazon-books, Amazon-electronic, and Yelp. The experimental results demonstrate the effectiveness of DIL in retraining GCN models. Furthermore, DIL is shown to be a generic approach for GCN-based model retraining and performs well when integrated with various GCN-based recommendation models.

5.2 Preliminary: GCN-based Model

Our retraining framework DIL is specially designed for GCN-based recommendation model. In this section, we thus present the general architecture of the base model. Specifically, we focus on a *user-item bipartite graph* used in recommendation, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In particular, there exists only two types of node in the graph - *user node* and *item node*. The interactions between users and items are denoted as the edge set $\mathcal{E}$. Edges only exist between a user node and an item node.

A GCN-based recommender model generally has the following modules: (i) an embedding layer, (ii) L neighbor aggregation & information update layers, and (iii) a final representation layer [12, 46].

Embedding layer. Like other deep learning based recommendation model [13, 194], a GCN-based recommendation model begins with an embedding layer. Utilizing user ID and item ID, the user embedding ($e_u \in \mathbb{R}^d$) and item embedding ($e_v \in \mathbb{R}^d$) can be obtained, representing the initial features of users and items respectively. In this context, d denotes the dimension of the embedding vector. Both the user and item embeddings are trainable parameters and are updated during the model training process.

Neighbor aggregation & information update. In a GCN-based recommender system, a graph structure is considered along with the features associated with user and item nodes. This architecture enables each node to gather information from

its neighboring nodes. For instance, the messages received from the neighbors (denoted as $N(u)$) of a user node u can be expressed as follows.

$$h_{N(u)} = AGGREGATION\left(h_v, \forall v \in N(u)\right) \tag{5.1}$$

To leverage the graph structure more effectively, neighbor aggregation can be extended to enable high-order message propagation. This entails stacking multiple layers of neighbor aggregation, allowing for recursive information gathering from neighbors up to L hops away. Hence, at the ℓ^{th} step, the message received from the neighbors of user u can be expressed as:

$$h_v^0 = e_v \tag{5.2}$$

$$h_{N(u)}^{(\ell-1)} = AGGREGATION\left(h_v^{(\ell-1)}, \forall v \in N(u)\right) \tag{5.3}$$

Based on the aggregated messages from neighbors, the representation of the user node is subsequently refined at each propagation layer with an update function, *e.g.,*, summation:

$$h_u^{\ell} = UPDATE\left(\left\{h_u^{\ell-1}, h_{N(u)}^{\ell-1}\right\}\right) \tag{5.4}$$

In the discussion above, the user node is used as an example. However, it is important to note that the same aggregation and update processes are also applicable to item nodes. By utilizing the user-item bipartite graph, a GCN-based recommender system can model not only the relationships between users and items but also the indirect relationships between users and other users, as well as between items and other items through the multi-hop aggregation mechanism.

Final representation layer. By stacking L message propagation layers, we obtain L node representations, one at each layer. In a recommender system, the approach for deriving a final representation for users/items may vary. For example, models like PinSage [48] and STAR-GCN [201] select the representation from the last layer, denoted as h_u^{final}, as the final representation. On the other hand, other models such as LightGCN [49], NGCF [50], and DGCF [129] aggregate all L node representations, including the initial embedding, using aggregation techniques

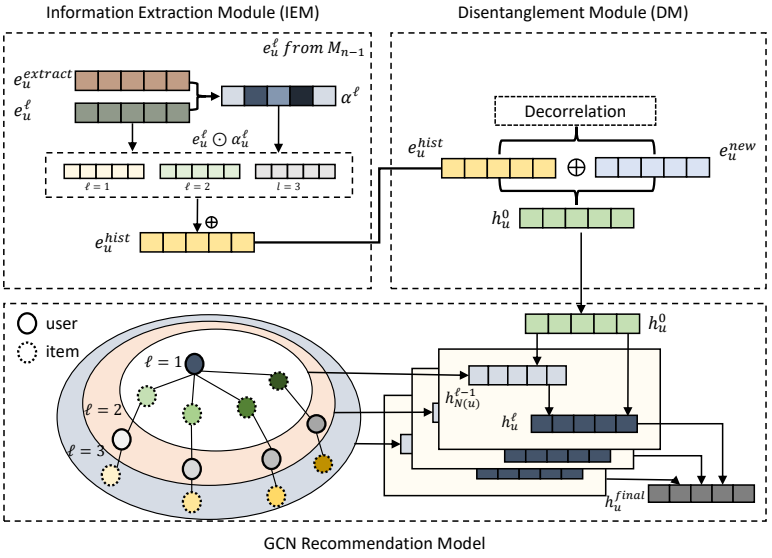


FIGURE 5.3: Model architecture of DIL using user node as an example. DIL consists of two components: Information Extraction Module (Design 1) and Disentanglement Module. The representations learned from DIL can be used as input for any GCN-based recommendation model.

like concatenation and mean pooling. These techniques allow for a comprehensive fusion of information across the layers to obtain the final representation.

$$h_u^{final} = AGGREGATION\left(h_u^{\ell}, \forall \ell \in [0, \ldots, L]\right) \tag{5.5}$$

To this end, the predicted preference score by user u on item v can be computed using the final representations h_u^{final} and h_u^{final}.

5.3 The Proposed Model

In accordance with the model retraining process depicted in Figure 5.2, it is important to note that at the end of time t_n, the retraining involves three key pieces of information. Firstly, R_n represents the newly observed user-item interactions that occurred within the time window t_n. Secondly, $R_{<n}$ encompasses the historical interactions that took place prior to the start time of t_n. Lastly, M_{n-1} refers to the most recent model that has been deployed for serving recommendations. During the model retraining process, these three components, namely R_n, $R_{<n}$, and M_{n-1}, can be utilized to update and refine the existing model M_{n-1} to get M_n. M_{n-1} is

commonly referred to as "the previous model" in subsequent discussions and serves as a basis for comparison and improvement during the retraining phase.

In order to capture the current trends and ensure the relevance of near-future recommendations, we construct a new user-item bipartite graph using the newly observed interactions, R_n. However, to incorporate long-term preferences, we also extract historical information from the previous model, M_{n-1}, and blend it into the new model, M_n. To avoid any confusion, we present our proposed Dynamic Incremental Learning (DIL) approach under the assumption that both users and items have interactions during t_{n-1}, which is applicable to the majority of users and items. In the following sections, we will outline how our model handles special cases related to other types of users/items. Additionally, we will discuss how the model deals with unseen users/items after being deployed for service at time t_{n+1}. This ensures that the model remains adaptive and capable of handling new users/items encountered during deployment.

Architecture Overview. The overall model architecture of the proposed DIL framework is presented in Figure 5.3. The process begins with the Information Extraction Module (IEM), which extracts historical information, denoted as e_{hist}, from the representations learned in the previous model, M_{n-1}. This information represents the long-term user preferences. Subsequently, the Disentanglement Module (DM) combines e_{hist} with e_{new} to generate the initial embedding, h_0, for the user node in the newly constructed bipartite graph. The learnable e_{new} represents the dynamic user preference that is to be learned from the new user-item interactions in R_n during model training. It can be initialized randomly or based on reference to previous models. In our implementation, we initialize e_{new} with the final representation, h_{final}, learned in M_{n-1}. We consider h_{final} to effectively capture the latest recommendation context before t_n, which is highly relevant to the current time period t_n. However, it does not represent purely long-term information, hence e_{hist} is needed in our model design. To ensure the differentiation between historical and new information, the DM decorrelates e_{hist} and e_{new}, as illustrated in Figure 5.3. The resulting h_0 serves as the initial feature for the user u and is fed into the base GCN recommendation model for training. This approach allows for the memorization of historical information while effectively distinguishing it from new information. A significant contribution of our work is the redesign of the embedding layer in the GCN-based recommender for model retraining, as highlighted

in our proposed DIL framework.

5.3.1 Information Extraction Module (IEM)

As mentioned earlier, the final representations learned in M_{n-1} primarily capture the recent recommendation context but may not fully represent the long-term user preferences and persistent characteristics of items. Consider the example of a user transitioning from badminton to tennis while still being a sports lover in general. Although recent interactions may be focused on badminton, a comprehensive analysis of all the user's interactions and related neighbors in the multi-hop user-item graph reveals his/her characteristics as a sports lover. Similar patterns can be observed when a user upgrades to the latest version of phone of a specific brand, where past interactions with various phone accessories reflect their loyalty to the brand. Thus, we believe that the long-term user preference and item characteristics cannot be simply represented by the previous model's representations alone. Instead, only certain aspects of the representations indicate the long-term preference and persistent characteristics over time. Furthermore, the long-term information is distributed across different layers as aggregated information from multi-hop neighbors.

Based on this understanding, we extract historical information from the representations at multiple layers in M_{n-1}. It is important to note that M_{n-1} is not limited to the set of interactions R_{n-1}, but also retains historical information extracted from its previous model, which in turn originates from M_{init} through multi-step backtracking. Our design aims to capture the slowly evolving historical information across models over time.

To determine what information to extract and how much information to extract from each layer, we introduce two parameters. The first parameter is called the "extractor embedding", which determines the specific parts of information to be extracted from each node. The second parameter, known as the "weight vector", then determines the quantity of information to be extracted from each layer. In this work, we propose two implementation designs for the extractor embedding and weight vector.

Design 1. For each node in the graph, we initialize a learnable extractor embedding, denoted as $e_{extract}$, which is responsible for determining which historical features should be extracted. This extractor embedding is unique for each node and enables us to capture the specific historical information relevant to that node. Next, we use e^{ℓ} to denote the output from the ℓ-th layer in M_{n-1}. It reflects the information gathered from the ℓ-hop neighbors in the time period $n-1$. Then, we compute the weight parameter for each layer, which determines the importance or weight α^{ℓ} assigned to the features at that layer, for the particular node.

$$\alpha^{\ell} = \sigma\left(e^{extract} \odot e^{\ell} + pos^{\ell}\right); \quad \ell \in [0, 1, \ldots, L] \tag{5.6}$$

Here, σ refers to a sigmoid function we employed to ensure that α^{ℓ} computed is within a range of $[0, 1]$. $\odot$ means element-wise product. We further introduce a positional embedding pos^{ℓ} to characterise the layer information, to facilitate the information extraction process. Having computed α^{ℓ}, the extracted information from each layer in model M_{n-1} is aggregated using an aggregated function denoted as $AGGREGATION(\cdot)$:

$$e^{hist} = AGGREGATION\left(\alpha^{\ell} \odot e^{\ell}, \forall \ell \in [0, 1, \ldots L]\right) \tag{5.7}$$

where, $AGGREGATION(\cdot)$ function can take different forms, including sum, mean pooling, or concatenation.

Design 2. Similar to design 1, we extract historical information from previous model by an extractor embedding $e^{extract}$. Each node is assigned with its own $e^{extract}$. Following that, we use the learnable matrix $W^{\ell} \in \mathbb{R}^{d \times d}$ to directly distill useful information from the ℓ-hop neighbors in model M_{n-1}, where $\ell \in [0, \ldots, L]$. The matrix W^{ℓ} acts as a filter or transformation applied to the features from the ℓ-hop neighbors. The resulting refined representation e^{ℓ}_{hist} captures the relevant long-lasting attributes from the ℓ-hop neighbors. It's worth noting that in this design, the learnable matrix W^{ℓ} is specific to each layer and enables direct extraction of useful information from the respective ℓ-hop neighbors.

$$e^{\ell}_{hist} = W^{\ell}(e^{extract} \odot e^{\ell}) \tag{5.8}$$

Next, we aggregate the information extracted from each layer by an aggregation function, *e.g.,* sum and concatenation.

$$e^{hist} = AGGREGATION(e^{\ell}_{hist}, \forall \in [0, \ldots, L]) \tag{5.9}$$

5.3.2 Disentanglement Module (DM)

To blend the historical information e^{hist} extracted from previous models and the new information e^{new} from the current set of interactions R_n, we sum the two embeddings together. This blending process results in a combined embedding h_0 for each node, which represents a fusion of both old and new information:

$$h^0 = e^{hist} + e^{new} \tag{5.10}$$

The combined embedding h_0 serves as the initial feature of each node in the newly built bipartite graph, capturing the integration of the long-term historical preferences and the dynamic preferences reflected in the new interactions. This initialization step enables the model to effectively leverage both the historical knowledge and the recent trends for accurate recommendation. The initial h_0 serves as the starting point for the base GCN-based recommendation model, such as LightGCN and NGCF, to capture the blended characteristics of the user or item.

To facilitate the differentiation between the historical information e^{hist} and the new information e^{new} during model retraining, we incorporate a disentanglement module. The primary objective of disentanglement is to enable the base recommendation model to acquire as much additional knowledge as possible in addition to the existing historical knowledge captured by e^{hist}. This additional knowledge specifically pertains to the new interactions and ensures that the model effectively captures the unique insights conveyed by these interactions.

To achieve disentanglement, we employ a decorrelation technique on the representations. Drawing inspiration from the work of DGCF [129], we utilize the distance correlation, denoted as L_{CORR}, as a loss term for this purpose. The distance correlation is computed using the following equation:

$$L_{CORR}\left(e^{hist}, e^{new}\right) = \frac{dCov\left(e^{hist}, e^{new}\right)}{\sqrt{dVar\left(e^{hist}\right) \cdot dVar\left(e^{new}\right)}} \tag{5.11}$$

In the equation provided, the term $dCov(\cdot)$ represents the distance covariance between two vectors, and $dVar(\cdot)$ represents the distance variance of a vector. The value of L_{CORR}, ranging from 0 to 1, indicates the level of dependency between the input vectors. A lower value of L_{CORR} suggests that the information learned in e^{hist} and e^{new} is less correlated, while still being valuable for message propagation and ultimately improving the accuracy of recommendations. The aim of decorrelation is to ensure that both historical and new information contribute independently to the recommendation process.

During the training process, both e^{hist} and e^{new} are used to form the initial embeddings of a node. While the model is trained using new interactions as ground-truth data instances, it is important to provide additional supervision for learning the long-term information represented by e^{hist}, as it encompasses the historical context across all time periods. To address this, we propose to supervise the learning of e^{hist} using historical interactions, which will be discussed in more detail in the following explanation.

5.3.3 Model Optimization

The node representations h_u^0 and h_i^0 are initialized by fusing the historical information e^{hist} and the new information e^{new}. These representations are then used as input for the base GCN recommendation model. After undergoing L layers of message propagation as described in Section 5.2, we obtain the final node representations h_u^{final} and h_i^{final} for the user and item, respectively. The dot product between h_u^{final} and h_i^{final} is used to predict a user's preference for an item, considering its effectiveness in recommendation [202]:

$$\hat{y}_{ui} = h_u^{final} \cdot h_i^{final} \tag{5.12}$$

In our model, we use the pairwise Bayesian Personalized Ranking (BPR) loss [174] as our objective function. The goal of this loss function is to optimize the model

parameters in a way that the predicted preference of a user is higher for items that the user has interacted with (*e.g.*,, clicked, rated, purchased) compared to items with which the user has no interactions. These interactions reflect the user's interests in the items. In particular, we optimize the BPR loss L_{new}:

$$L_{new} = \sum_{(u,i,j)\in O} -\ln \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) \tag{5.13}$$

where item i is an item interacted with user u in the current period t_n, while item j is a negative item sampled from the items that do not have interactions with user u till t_n.

As mentioned previously, we also aim to supervise e_u^{hist} to ensure it effectively represents historical information. To achieve this, we utilize historical interactions sampled from $R_{<n}$ to provide supervision for e_u^{hist}. Therefore, we modify the loss term in Equation 5.13 as follows:

$$L = \sum_{(u,i,k,j)\in O} -\ln \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) - \ln \sigma\left(e_u^{hist} \cdot e_k^{hist} - e_u^{hist} \cdot e_j^{hist}\right) \tag{5.14}$$

In Equation 5.14, item k represents an item that has interacted with user u in previous periods, and item j is a negative sample that does not have any interactions with user u. It's important to note that item j can be used as a negative sample for training on both new interactions and past interactions, as it represents an item that the user has not interacted with. By including this additional supervision, we ensure that the disentangled representation effectively separates the semantics of new and historical information, enhancing the overall model performance.

The complete objective function, incorporating both the pairwise BPR ranking loss and the disentanglement term, is as follows:

$$L_{final} = L + \lambda L_{CORR} \tag{5.15}$$

where the parameter λ is a hyperparameter that controls the weight of the disentanglement term relative to the ranking loss.

5.3.4 Inactive Users and Inference

In the incremental learning scenario, users can be classified into three types: active users, inactive users, and new users. "Active users" are those who have interactions in both the previous period t_{n-1} and the current period t_n. For these users, their information can be directly modeled using the current interaction graph constructed with the newly observed interactions in R_n. Additionally, we can obtain their historical information from the previous model M_{n-1}. "New users" are those who have their first interactions in the current period t_n. Their features are randomly initialized and will be updated during the current phrase of training. "Inactive users", on the other hand, are users who were active in the past but had no interactions in the previous period t_{n-1}. There are several ways to handle their historical information e^{hist} during retraining. One approach is to treat them as new users, initializing their features randomly. Another approach is to extract their historical information from the model when they were last active, which would require keeping all historical model parameters. The last approach is to retain the inactive users in model retraining from the beginning (t_0), even though they had no edges in the graphs during their inactive periods. In this case, their parameters are updated along with the model retraining process. The implementation used in this work adopts the last approach.

It is worth noting that active users comprise the largest group in a given time period t_n. In the datasets used in this study, the average percentage of active users in a time period was 81.1% for Amazon-books, 64.3% for Amazon-electronics, and 69.7% for Yelp. This statistic demonstrates that users tend to visit a platform regularly.

After retraining, the newly updated model M_n is deployed for service in the next time period t_{n+1}. At this stage, there may be "unseen users" who have their very first interactions in t_{n+1}. The features of these unseen users are randomly initialized, and the recommendation model uses these randomly initialized vectors for making recommendations. In practical scenarios, if the platform has access to certain attributes of the users (*e.g.,*, through user registration), their features can be initialized based on the known attributes. The same initialization process applies to items as well.

TABLE 5.1: Statistics of the datasets used in experiments.

Dataset	Time Period	Users Number	Items Number	#Warm-up interactions	#Retraining interactions
Amazon-books	01.12.2017 - 31.12.2017	34,195	25,798	458,145	483,293
Amazon-electronic	01.01.2013 - 31.12.2017	91,592	34,401	448,394	1,028,395
Yelp	01.01.2012 - 31.12.2017	42,124	29,099	453,470	649,627

5.4 Experiment Setup

In our experimental evaluation, we utilize three well-known datasets: Amazon-books [3], Amazon-electronics [3], and Yelp[1]. The Amazon-books and Amazon-electronics datasets are obtained from the Amazon product review dataset, while the Yelp dataset is collected from the Yelp platform.

To assess the performance of our proposed incremental learning framework, we choose two popular graph-based recommendation models, namely LightGCN [49] and NGCF [50], as our base models. These models have been widely studied and proven effective in recommendation tasks.

The objective of our experiments is to demonstrate the effectiveness and robustness of the incremental learning framework we propose, and we evaluate its performance using the selected datasets and base recommendation models.

5.4.1 Incremental Learning Scenario

In our experimental setup, we follow a standard approach for incremental learning in recommender systems, which has been commonly used in prior studies [110, 111, 120]. The process is illustrated in Figure 5.2.

First, we initialize the base recommendation model and conduct a warm-up training using all the interactions available in the warm-up period. During this warm-up training, the base model is batch trained to establish a basic understanding of the users and items.

After the warm-up period, we divide the remaining interactions into six distinct periods for incremental learning. These periods are denoted as R_0 to R_5. The

[1] `https://www.yelp.com/dataset`

retraining process begins with R_0, and the model is updated using the interactions from this period. The updated model is then used to predict the interactions in R_1. The accuracy of these predictions on R_1 is used as a validation metric for hyperparameter tuning.

The remaining four sets of interactions (R_2 to R_5) are treated as test sets. For each test set, we perform early stopping using the first 10% of the interactions, and the recommendation accuracy is calculated based on the remaining 90% of the interactions. The average recommendation accuracy across these four test sets is considered as the accuracy metric for the specific retraining strategy.

This experimental setup allows us to evaluate and compare different retraining strategies in terms of their recommendation accuracy over multiple test sets.

Datasets. Table 5.1 presents the statistics of the three datasets used in our experiments. These datasets, namely Amazon-books, Amazon-electronics, and Yelp, have undergone a preprocessing step known as "10-core filtering". This filtering ensures that only users and items with at least 10 interactions are included in the dataset, resulting in a more focused and meaningful analysis without extremely inactive users/items.

For the *Amazon-books* dataset, we focus on interactions that occurred in the year 2017. The first six months of 2017 are designated as the warm-up period, during which the base model is trained. The remaining six months of interactions are divided into six sets, with each set representing one month of data. These sets are used for subsequent retraining and evaluation. Regarding the *Amazon-electronics* dataset, we consider interactions spanning from 2013 to 2017. The initial two years (2013-2014) are designated as the warm-up period for training the base model. The subsequent three years of interactions are divided into six sets, with each set representing a half-year period, for retraining and evaluation. For the *Yelp* dataset, we utilized interactions recorded from 2012 to 2017. The warm-up period is defined as the time span from January 1, 2012, to December 31, 2014. The interactions recorded between 2015 and 2017 are divided into six sets, with each set representing a half-year period. Then, retraining and evaluation are conducted on each set sequentially.

Evaluation Metrics. We approach the task as a top-N recommendation problem with implicit feedback. To evaluate the performance of our proposed framework

and baselines, we assess the accuracy of the top-20 recommendations. This means that at each time t_n, we recommend 20 items to each user from the set of *all items* observed until t_n. The utilization of all-item-ranking [181] helps eliminate potential biases in the recommendation results [160].

Regarding the evaluation metrics, we report the Recall@20 and NDCG@20 scores. Recall measures the percentage of correctly recommended instances among the test instances. NDCG (Normalized Discounted Cumulative Gain) takes into account the ranking positions of the correctly recommended items. It is worth noting that the Recall@20 and NDCG@20 scores may appear relatively low due to two reasons. Firstly, since we employ all-item-ranking, recommending the correct item from a large set of candidates is challenging. Secondly, the train-test data split strictly follows the timeline in our setup, which means that the recommendation accuracy is influenced by both unseen users and unseen items in a new time window.

5.4.2 Baselines

In our evaluation, we assess DIL against five baseline approaches. These baselines encompass a range of retraining strategies and incremental learning techniques. The baselines are as follows:

Fine-tune: This strategy involves fine-tuning the base recommendation model using the newly observed interactions in the most recent time period.

Full-retrain: In this approach, the base recommendation model is retrained from scratch using both the historical interactions and the newly observed interactions in the most recent time period.

SPMF: SPMF [116] is an experience replay strategy. It is designed for general retraining purpose, but not necessarily on graph-based recommendation model. Specifically, It maintains a memory of past interactions which will be replayed during retraining. The memory size, *i.e.,* number of past interactions, is tuned from $\{20000, 40000, 60000\}$.

GraphSAIL: GraphSAIL targets retraining on GCN-based recommendation model. In particular, distillation loss terms are incorporated to ensure the local structure of the previous graph, global structure of the previous graph and self-information

in the previous graph are well maintained even after retraining with the current graph. We experiment with different values for the term weights, specifically in the range of $\{0.000001, 0.001, 1\}$, to find the optimal configuration. Additionally, GraphSAIL constructs a global graph structure by performing k-means clustering on the node representations learned from the GCN-based recommendation model. The number of clusters, denoted as k, is another hyperparameter that needs to be tuned. We explore different values of k in the range of $\{5, 10, 20\}$ to determine the optimal number of clusters for capturing the global graph structure effectively.

CI: CI is a model-based incremental learning framewor for GCN-based recommendation model. It combines historical node representation with new representation learned from newly observed interactions via convoultional neural network, to aggregate historical information and new information. In addition, the representations of the inactive nodes are updated even if there are no observed interactions for these nodes. This is accomplished by establishing connections between the inactive nodes and their k nearest neighbor active nodes in the graph. In our experiment, We tune the number of neighbors in k-nearest neighbor from $\{10, 30, 50\}$. We tune the coefficient on inactive node updates from $\{0.25, 0.5, 0.75, 1\}$.

For all the models used, we tune learning rate from $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ and embedding dimension from $\{32, 64, 128\}$, and regularization coefficient from $\{10^{-6}, 10^{-4}, 10^{-2}\}$. By including these baselines, we cover a diverse set of incremental learning techniques, including experience replay, knowledge distillation and model-based techniques. It enables a comprehensive evaluation of DIL's effectiveness and performance in comparison with the existing approaches.

5.4.3 DIL Implementation Details

During the grid search for hyperparameters, we identify the optimal design between **Design 1** and **Design 2** of the IEM module. For Amazon-books and Yelp, **Design 1** performs better, while for Amazon-electronic with LightGCN, **Design 2** shows better results. These optimal designs are used for the respective datasets in the remaining experiments.

For training the DIL model, we utilized the Adam optimizer with initial learning rates of 0.0001 for Amazon-books and Yelp, and 0.00001 for Amazon-electronic.

TABLE 5.2: Recall@20 and NDCG@20 results on Amazon-books, Amazon-electronic and Yelp, upon instantiating different retraining strategies on LightGCN and NGCF. Best results are in boldface and second best underlined.

Dataset		Amazon-books		Amazon-electronic		Yelp	
Base Model	Baseline	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
LightGCN	Fine-tune	0.0368	0.0140	0.0256	0.0099	0.0591	0.0237
	Full-retrain	0.0332	0.0116	0.0210	0.0080	0.0560	0.0217
	SPMF	0.0459	0.0169	0.0262	<u>0.0101</u>	<u>0.0627</u>	<u>0.0249</u>
	GraphSAIL	<u>0.0490</u>	<u>0.0180</u>	<u>0.0265</u>	<u>0.0101</u>	0.0598	0.0240
	CI	0.0390	0.0159	0.0192	0.0071	0.0521	0.0205
	DIL	**0.0690**	**0.0263**	**0.0271**	**0.0102**	**0.0649**	**0.0259**
NGCF	Fine-tune	<u>0.0618</u>	0.0221	0.0206	0.0078	0.0355	0.0136
	Full-retrain	0.0285	0.0106	0.0187	0.0071	0.0362	0.0137
	SPMF	0.0582	<u>0.0224</u>	<u>0.0265</u>	<u>0.0102</u>	<u>0.0445</u>	0.0170
	GraphSAIL	0.0590	0.0204	0.0240	0.0094	0.0425	0.0165
	CI	0.0571	0.0200	0.0236	0.0091	0.0442	<u>0.0176</u>
	DIL	**0.0635**	**0.0233**	**0.0274**	**0.0105**	**0.0478**	**0.0186**

The regularization coefficient used is 0.0001 for Amazon-books and Yelp, and 0.000001 for Amazon-electronic. The dimension size for node representations is set to 128 for all three datasets. The weight on the disentanglement term, denoted by λ in equation 5.15, varied across datasets and models. We performed tuning and select values from the set $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ to determine the optimal weights.

Regarding the base models, we employ mean pooling as the aggregation function for the final representation. Although the original aggregation function in NGCF involves concatenation of all node representations, we find that mean pooling yields better recommendation performance based on our experiments.

5.5 Experiment Results

We compare the recommendation accuracy of DIL with five baselines on three datasets, using two base models. Then, we test the effectiveness of the two main components, Information Extraction Module and Disentanglement Module, in DIL.

5.5.1 Effectiveness & Robustness

The experimental results, as shown in Table 5.2, demonstrate that DIL outperforms all baselines on all datasets, based on both Recall and NDCG evaluation metrics.

Specifically, the version of DIL with LightGCN as the base model achieves superior performance compared to its counterpart with NGCF on the Amazon-books and Yelp datasets, by a significant margin. However, on the Amazon-electronic dataset, DIL with LightGCN reports slightly lower results than its NGCF version. Notably, GraphSAIL and SPMF are among the best-performing baselines, with GraphSAIL on LightGCN ranking second in terms of performance on Amazon-books and Amazon-electronic.

To illustrate the performance of DIL over time, we use the example of LightGCN on the Amazon-books dataset and plot the Recall@20 scores over the four testing periods in Figure 5.4. It is observed that the ranking orders of the baselines vary across different periods, which aligns with findings in prior studies such as [18] on other datasets. This observation emphasizes the importance of monitoring evaluation results over time to avoid potential bias in a particular period. In this study, we average the evaluation scores over time and still observe superior performance by DIL, indicating its robustness and effectiveness as a retraining framework.

Among all baselines, CI is considered the most similar to DIL as it also fuses historical knowledge from M_{n-1} with new information learned in the current period. However, the key difference lies in the ways of extracting and fusing the old and new information. CI conducts "post-learning fusion" by aggregating representations outputted by GCN in two different periods, while DIL conducts "pre-learning fusion" by fusing old and new information to form the input (initial feature of nodes) of GCN. Through disentanglement, DIL can effectively distinguish between historical and new information. Moreover, DIL adjusts the extraction of historical information during retraining by refining the information extraction module. Experimental results demonstrate that DIL significantly outperforms CI.

Fine-tune generally outperforms Full-retrain in most settings, except for the version with NGCF on Yelp. This indicates that the more recent interactions are more relevant to the current recommendation context. However, Fine-tune does not exhibit competitive results in most cases, except for Fine-tune on Amazon-books with NGCF. This suggests that completely forgetting historical information is not an ideal choice either. Both SPMF and DIL involve historical interactions sampled from $R_{<n}$ during training. Interestingly, the optimal number of interactions required by SPMF is not the largest among the hyperparameter candidates,

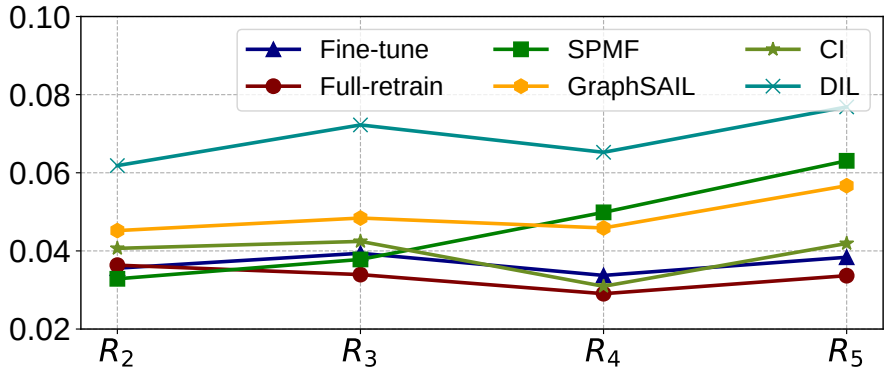


FIGURE 5.4: Recall@20 when retraining LightGCN on Amazon-books, over the 4 testing periods.

indicating that a higher number of historical interactions does not necessarily lead to better recommendation.

Finally, the absolute values of the recommendation accuracy on the Amazon-electronic dataset are much lower than those on the other datasets. This can be attributed to the sparsity of this dataset, where the user pool is two or three times larger than other datasets, while the number of interactions is only about 50% more. Additionally, the electronic category tends to have a larger variety of items with shorter lifespans compared to books and restaurants, making it a challenging dataset for recommendation tasks.

5.5.2 Ablation Study

In order to verify the effectiveness of each component in DIL, we conduct experiments to explore different designs in the Information Extraction Module (IEM) and the Disentanglement Module.

The IEM is responsible for distinguishing different neighbors and treating the outputs of different layers differently. We conduct experiments to examine the need for differentiating neighbors. Specifically, we modify DIL to create three variants: DIL (mean), DIL (first), and DIL (last).

In DIL (mean), we extract information from the mean node representations across all L layers and the initial embedding of a node. In DIL (first) and DIL (last), we extract historical information from the initial embedding and the last-layer representation, respectively.

TABLE 5.3: Ablation study to verify the effectiveness of designs in the Information Extraction Module.

Base Model		LightGCN		NGCF	
Dataset	Baseline	Recall@20	NDCG@20	Recall@20	NDCG@20
Amazon-books	DIL (mean)	<u>0.0677</u>	<u>0.0258</u>	<u>0.0609</u>	0.0229
	DIL (first)	0.0613	0.0234	**0.0635**	<u>0.0230</u>
	DIL (last)	0.0663	0.0197	0.0582	0.0221
	DIL	**0.0690**	**0.0263**	**0.0635**	**0.0233**
Amazon-electronic	DIL (mean)	<u>0.0285</u>	**0.0111**	0.0261	<u>0.0101</u>
	DIL (first)	0.0269	0.0099	0.0183	0.0071
	DIL (last)	**0.0287**	0.0085	**0.0283**	0.0084
	DIL	0.0271	<u>0.0102</u>	<u>0.0274</u>	**0.0105**
Yelp	DIL (mean)	0.0640	0.0256	0.0435	0.0165
	DIL (first)	**0.0658**	**0.0262**	<u>0.0458</u>	<u>0.0176</u>
	DIL (last)	0.0630	0.0187	0.0413	0.0160
	DIL	<u>0.0649</u>	<u>0.0259</u>	**0.0478**	**0.0180**

The results, presented in Table 5.3, show that different hops of neighbors provide different information, leading to varying Recall@20 and NDCG@20 scores. Moreover, the ranking orders of the DIL variants change across datasets and models. For example, when instantiated on LightGCN, the mean node representation of GCN contributes more to the recommendation performance on Amazon-books compared to using the initial embedding. On the other hand, for the Yelp dataset, the more important factor is the initial embedding. These observations indicate that neighbors should be given different weights depending on the specific domain of the recommendation task.

Our proposed DIL architecture distinguishes neighbors from different hops in the IEM. According to the results in Table 5.3, our design is effective in most cases, as DIL achieves the best recommendation accuracy. However, it is important to note that the IEM is an independent module within DIL, and it is not necessary to strictly adhere to the proposed designs. The IEM can be revisited and redesigned to suit different recommendation scenarios. Replacing the IEM does not affect the overall architecture of DIL for retraining a GCN-based recommendation model.

Disentanglement Module (DM) decorrelates the long-term information and new information. We evaluate the impact of the decorrelation and supervision components in DIL by conducting experiments and comparing the results. The findings,

TABLE 5.4: Ablation study to verify the efficacy of designs in the Disentanglement Module.

Base Model		LightGCN		NGCF	
Dataset	Baseline	Recall@20	NDCG@20	Recall@20	NDCG@20
Amazon-books	DIL-decorr	0.0617	0.0231	0.0620	0.0229
	DIL-supervision	0.0514	0.0192	0.0502	0.0182
	DIL-decorr-supervision	0.0575	0.0208	0.0516	0.0188
	DIL	**0.0690**	**0.0263**	**0.0635**	**0.0233**
Amazon-electronic	DIL-decorr	0.0255	0.0095	0.0255	0.0098
	DIL-supervision	0.0249	0.0092	0.0245	0.0095
	DIL-decorr-supervision	0.0251	0.0093	0.0262	0.0101
	DIL	**0.0271**	**0.0102**	**0.0274**	**0.0105**
Yelp	DIL-decorr	0.0643	0.0258	0.0468	0.0181
	DIL-supervision	0.0640	0.0257	0.0461	0.0181
	DIL-decorr-supervision	0.0640	0.0251	0.0446	0.0172
	DIL	**0.0649**	**0.0259**	**0.0478**	**0.0186**

presented in Table 5.4, demonstrate that both decorrelation and supervision contribute to improve recommendation accuracy. Removing either component leads to a degradation in Recall@20 and NDCG@20 values compared to the complete DIL model. Interestingly, on the Amazon-books dataset, we observe that DIL-decorr-supervision outperforms DIL-supervision alone, indicating that decorrelation may not be effective without supervision. This could be attributed to the model converging to a trivial solution that lacks meaningful recommendations in the absence of supervision.

5.5.3 Discussion

In our experiments, we have compared DIL with other incremental learning frameworks to demonstrate its effectiveness. It is important to note that our experimental setup focuses on evaluating different retraining strategies using a fixed time interval schedule. However, in real-world scenarios, the retraining schedule itself is a crucial hyperparameter that significantly affects recommendation accuracy.

Furthermore, we acknowledge that DIL does not directly incorporate information from e^{hist} learned in previous periods before t_{n-1} when making updates in the time window t_n. This means that e^{hist}_{n-2} learned at time t_{n-2} is ignored when updating in time window t_n. However, e^{hist}_{n-2} contains information from both R_{n-3} and $R_{<n-3}$, while e^{hist}_{n-1} contains information from $R_{n-2} \cup R_{<n-2}$. In cases where

there are long-lasting attributes that influence a user's behavior over time, there can be intersections between e^{hist}_{n-1} and e^{hist}_{n-2}. DIL does not explicitly model the continuity of e^{hist} to memorize long-term preferences. Instead, it maintains continuity through the supervision operation in the disentanglement module. Modeling the direct continuity of e^{hist} is non-trivial because e^{hist} from different periods may exist in different latent spaces, and continuity only applies to certain features of e^{hist}. Exploring and addressing the modeling of continuity in e^{hist} is an interesting avenue for future research.

5.6 Summary

In this chapter, we show the importance of retraining a recommendation model to model the dynamic temporal context along the global timeline. Although it has been shown in Chapter 4 that recent interactions are useful for predicting the user interactions in a recent context, we argue that long-term preference not reflected in the recent interactions is also an essential factor in recommendation.

In order to capture both short-term interest and long-term preference, we propose to adopt the incremental learning technique. We propose a novel and versatile incremental learning strategy, referred to as Dynamic Information Learning (DIL), specifically designed for recommender systems based on Graph Convolutional Networks (GCNs). The fundamental concept behind DIL is the integration of pre-learning fusion. In this approach, we extract long-term historical information from a previously trained model and combine it with the learnable new information. The resulting embedding, obtained after the fusion process, serves as the initial feature for a node in the GCN-based recommender model.

To ensure that the learnable new information accurately reflects the latest trends in the recommender system, we employ disentanglement techniques to ensure its independence from historical information. By implementing DIL on two prominent GCN models, namely LightGCN and NGCF, we demonstrate its effectiveness in retraining and incorporating both short-term and long-term information for recommendation purposes. The concept of "pre-learning fusion" offers an alternative perspective on how to extract and merge historical information during model retraining.

Chapter 6

Conclusion and Future Work

In this final chapter, we present the concluding remarks and highlight the significant contributions made in this thesis. Furthermore, we delve into the potential areas for future research that can build upon the findings of this study.

6.1 Conclusion

In recent years, the exponential growth of web-based services has resulted in an overwhelming abundance of online information. Consequently, users face challenges in sifting through this vast amount of data to obtain relevant and personalized information. To tackle this problem of information overload, recommender systems have emerged as valuable tools. The fundamental concept behind a recommender system is to identify and suggest items or information that align with a user's interests and preferences. One approach to achieve this is through content-based recommendation, whereby a user's profile is matched with relevant items based on their content characteristics. In addition to content-based recommendation, collaborative filtering recommender systems have gained considerable attention. These systems operate under the assumption that users who have shown a similarity in their past item selections are likely to exhibit similar preferences in the future. Unlike content-based recommender systems, which rely on user profiles and item descriptions for recommendation, a collaborative filtering recommender systems primarily rely on historical user-item interactions for learning purposes. In this

thesis, our focus centers on the investigation and analysis of collaborative filtering recommender systems.

Chapter 1 of this thesis raises a fundamental question regarding the progress of research in recommender systems, given the substantial volume of publications in this field. To explore this question, we refer to two works [6, 7] that investigate the performance of various recommendation algorithms, ranging from simple nearest neighbor models to more complex neural network-based approaches. The findings suggest that the extent to which recommender systems have experienced significant improvement remains uncertain. This finding can be attributed to two primary factors. Firstly, the reproducibility of recommender system performance is not consistently ensured across different studies. This lack of reproducibility hinders the ability to interpret and compare recommendation performance effectively. Secondly, it has been observed that simpler models can exhibit superior recommendation capabilities in certain scenarios compared to more complex models. The authors suggest that the aforementioned observations are attributed to the inconsistent evaluation settings adopted in various research papers. Furthermore, in most cases, evaluations are conducted without considering the temporal aspects of recommender systems, introducing biases in experimental results. To address these issues and overcome the limitations of existing research, this thesis proposes a comprehensive investigation of evaluation in recommender systems from a global timeline perspective. By considering the temporal dynamics of user-item interactions, we aim to provide a more comprehensive understanding of recommender system performance and facilitate reproducible research in this field.

Following Chapter 1, we conduct comprehensive reviews on existing works in the research of recommender system in Chapter 2. The primary focus is on recommender systems that consider the temporal factors in user preference modelling as well as the works that study the evaluation processes of recommender system. Upon reviewing existing works, we highlight in Chapter 3 that the widely adopted evaluation protocols suffer from the data leakage issue, due to the ignorance of the global timeline. With data leakage, the recommendation results obtained differ from the experiment without data leakage. Notably, the impact of data leakage on recommendation performance is not uniformly positive or negative. Instead, the severity of data leakage leads to inconsistent recommendation performance. This finding offers an explanation for the uncertain progress observed in recommender

system research. In Chapter 4, we further demonstrate that ignorance of global timeline leads to the failure in capturing the temporal context. It is essential to capture the temporal context with the recently observed interactions. Lastly in Chapter 5, we propose an incremental learning framework that incorporates long-term user preferences alongside the most recent user-item interactions. It is important to note that only relevant long-term information is retained to enhance recommendation accuracy.

In conclusion, a global timeline should be followed in both training stage and evaluation stage of a recommender system. It not only ensures the reproducibility and meaningful interpretation of the recommendation results, but also ensures that temporal context can be well captured to model the dynamics in user preference and market trends.

6.2 Future Work

6.2.1 Session-based Recommender System

Chapter 4 of this thesis reveals the significant influence of the recency of interactions in recommender systems. Specifically, it is found that incorporating the most recent interactions in the training of a recommendation model is crucial for capturing the prevailing temporal context along the global timeline. One potential solution to address this is through the implementation of a session-based recommender system.

A session-based recommender system focuses on learning user preferences based on the interactions that occur within a session [11, 203]. Recommendations are then made for the subsequent interactions or a few interactions within the same session. As sessions have a limited duration, the recency of interactions is inherently ensured in this approach. By utilizing session-based recommender systems, the most recent user behavior is effectively captured, enabling more accurate and timely recommendations that align with the evolving preferences of users over time.

In many session-based recommender systems, the focus is solely on the interactions that occur within the sessions for predicting the next interaction. These systems treat sessions as anonymous entities, without considering the user's identity or

their historical interactions prior to the session. Consequently, valuable information from a user's past interactions is disregarded. GRU4Rec [83] is an example of such a session-based recommender system. It utilizes a Gated Recurrent Unit (GRU) network to model sequential dependencies between items within a session and predicts the next item in the sequence. Similarly, Li et al. [85] propose the use of GRU with an attention mechanism to model anonymous sessions. In another study [204], anonymous sessions are modeled as hyperbolic session graphs, which capture both chronological and hierarchical information. The authors also employ clustering strategies to capture collaborations within and between sessions, enhancing the modeling of session dynamics.

Session-based recommenders that are trained from anonymous sessions holds a strong assumption that interactions that happen within a session are mainly driven by the short-term preference that exists at the time of the session. Hence, user identity or a user's historical interactions are not needed in recommendation. Apart from this line of work, session-based recommender systems can learn from non-anonymous sessions to capture dynamic preferences reflected across interactions. Hu et al. [205] incorporates user ID with sessions to conduct personalized session-based recommendation. The authors in [206] extends personalized session-based recommendation beyond leveraging only user ID. They propose HGNN which construct a global heterogeneous graph containing user-item interactions over all sessions, item-item connections in all sessions and global co-occurrence of items. HGCN then learns the long-term user preference and item characteristics from the global graph.

Session-based recommender system can capture a user's instant purpose within a short-duration session. Research efforts can be devoted to infer a user's long-term preference from multiple sessions he/she has to enhance recommendation performance. More importantly, sessions shall be considered in chronological order to correctly model the dynamics.

6.2.2 Intent-aware Recommender System

In many scenarios, not only should a recommender system concern about *what a user likes*, but also about *what a user intends to do*. Identifying a user's intention

base on his/her recent behaviours ensure that the items recommended fit his/her needs. We term this line of study as **intent-aware recommender system**.

Bhattacharya et al. [207] propose to combine frequency-based recommender system and context-based recommender system for intent learning. In particular, the frequency-based recommender system constructs user navigation graph based on user behaviours. It models the transition probabilities between items. When it comes to the context-based recommender system, the contextual information such as temporal features are incorporated to predict a user's intent for recommendation. The prediction score of an item is then combined with the transitional probability learned from frequency-based recommender system to form the final recommendation score. This algorithm works under the assumption that only one specific intention can be observed in a user's interaction sequence. Contrary to that, the authors of [208] argue that multiple intentions can be observed in one single user interaction sequence. They design purpose-specific recurrent unit (PSRU) to extract purpose from user behaviours. Multiple PSRUs form purpose-specific recurrent network (PSRN) to model purpose-specific dependencies between items. One PSRN caters to one specific purpose. In together, multiple PSRNs form the mixture-channel recurrent networks to extract multi-purpose in a user interaction sequence. Similarly in [209], the authors design an *implicit intent mining*(IIM) module to detect user intents underlying a user's interaction sequence. In particular, the IIM module is designed with intent-specific attention mechanisms to place intent-specific weights on different items in an interaction sequence. Unlike [208] and [209] which concern multi-intent modelling, [210] concerns multi-level user intents modelling. Specifically, given a user session, multi-level user intent information is used to construct a pool of attention maps that can be used to obtain session representation via attention.

Notably, intent detection should be conducted with the consideration of temporal context that user-item interactions happen. Sun [177] shares an in-depth discussion on this point. The key idea is that user-item interactions shall be interpreted from a decision making perspective. Across different temporal context, decision making processes are different despite the selection of the same items. Hence, it is essential to consider real-time intent-aware recommendation to avoid potential bias induced by different temporal contexts..

List of Author's Awards, Patents, and Publications

6.3 Journal Articles

- **Yitong Ji**, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A Critical Study on Data Leakage in Recommender System Offline Evaluation. ACM Trans. Inf. Syst. 41, 3, Article 75 (July 2023), 27 pages.

6.4 Conference Proceedings

- **Yitong Ji**, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20). Association for Computing Machinery, New York, NY, USA, 1749–1752.

- **Yitong Ji**, Aixin Sun, Jie Zhang, and Chenliang Li. 2022. Do Loyal Users Enjoy Better Recommendations? Understanding Recommender Accuracy from a Time Perspective. In Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '22). Association for Computing Machinery, New York, NY, USA, 92–97.

- **Yitong Ji**, Aixin Sun, and Jie Zhang. 2023. Blending Old and New: Disentangled Incremental Learning for Graph-based Recommenders. Under Review.

Bibliography

[1] Angela Edmunds and Anne Morris. The problem of information overload in business organisations: a review of the literature. *Int. J. Inf. Manag.*, 20(1): 17–28, 2000. doi: 10.1016/S0268-4012(99)00051-1. URL `https://doi.org/10.1016/S0268-4012(99)00051-1`. 1

[2] David Bawden, Clive Holtham, and Nigel Courtney. Perspectives on information overload. In *Aslib proceedings*. MCB UP Ltd, 1999. 1

[3] Brent Smith and Greg Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Comput.*, 21(3):12–18, 2017. doi: 10.1109/MIC.2017.72. URL `https://doi.org/10.1109/MIC.2017.72`. 1, 12, 94

[4] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 191–198. ACM, 2016. doi: 10.1145/2959100.2959190. URL `https://doi.org/10.1145/2959100.2959190`. 1

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 7–10. ACM, 2016. doi: 10.1145/2988450.2988454. URL `https://doi.org/10.1145/2988450.2988454`. 1

[6] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 101–109. ACM, 2019. doi: 10.1145/3298689.3347058. URL `https://doi.org/10.1145/3298689.3347058`. 2, 13, 33, 64, 106

[7] Yushun Dong, Jundong Li, and Tobias Schnabel. When newer is not better: Does deep learning really benefit recommendation from implicit feedback? *CoRR*, abs/2305.01801, 2023. doi: 10.48550/arXiv.2305.01801. URL `https://doi.org/10.48550/arXiv.2305.01801`. 2, 33, 48, 106

[8] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52 (1):5:1–5:38, 2019. doi: 10.1145/3285029. URL https://doi.org/10.1145/3285029. 10, 13, 33, 34, 48

[9] Pedro G. Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User Adapt. Interact.*, 24(1-2):67–119, 2014. doi: 10.1007/s11257-012-9136-x. URL https://doi.org/10.1007/s11257-012-9136-x. 2, 31, 36, 40

[10] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, 51(4):66:1–66:36, 2018. doi: 10.1145/3190616. URL https://doi.org/10.1145/3190616. 2, 17

[11] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. A survey on session-based recommender systems. *ACM Comput. Surv.*, 54(7):154:1–154:38, 2022. doi: 10.1145/3465401. URL https://doi.org/10.1145/3465401. 2, 107

[12] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5): 97:1–97:37, 2023. doi: 10.1145/3535101. URL https://doi.org/10.1145/3535101. 2, 13, 84

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 173–182. ACM, 2017. doi: 10.1145/3038912.3052569. URL https://doi.org/10.1145/3038912.3052569. 2, 13, 37, 48, 50, 57, 64, 69, 84

[14] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 689–698. ACM, 2018. doi: 10.1145/3178876.3186150. URL https://doi.org/10.1145/3178876.3186150. 2

[15] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Exploring data splitting strategies for the evaluation of recommendation models. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 681–686. ACM, 2020. doi: 10.1145/3383313.3418479. URL https://doi.org/10.1145/3383313.3418479. 3, 27, 30, 31, 33

[16] Milena Filipovic, Blagoj Mitrevski, Diego Antognini, Emma Lejal Glaude, Boi Faltings, and Claudiu Musat. Modeling online behavior in recommender systems: The importance of temporal context. In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop*

2021 co-located with the 15th ACM Conference on Recommender Systems (RecSys 2021), Amsterdam, The Netherlands, September 25, 2021, volume 2955 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL https://ceur-ws.org/Vol-2955/paper4.pdf. 30

[17] Rocío Cañamares, Pablo Castells, and Alistair Moffat. Offline evaluation options for recommender systems. *Inf. Retr. J.*, 23(4):387–410, 2020. doi: 10.1007/s10791-020-09371-3. URL https://doi.org/10.1007/s10791-020-09371-3. 26, 27, 30, 40

[18] Sergey Kolesnikov and Mikhail Andronov. CVTT: cross-validation through time. *CoRR*, abs/2205.05393, 2022. doi: 10.48550/arXiv.2205.05393. URL https://doi.org/10.48550/arXiv.2205.05393. 30, 31, 99

[19] Robin D. Burke. Evaluating the dynamic properties of recommendation algorithms. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 225–228. ACM, 2010. doi: 10.1145/1864708.1864753. URL https://doi.org/10.1145/1864708.1864753. 30

[20] Teresa Scheidt and Joeran Beel. Time-dependent evaluation of recommender systems. In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2021 co-located with the 15th ACM Conference on Recommender Systems (RecSys 2021), Amsterdam, The Netherlands, September 25, 2021*, volume 2955 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL https://ceur-ws.org/Vol-2955/paper10.pdf. 30, 31

[21] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. A revisiting study of appropriate offline evaluation for top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 41(2), dec 2022. ISSN 1046-8188. doi: 10.1145/3545796. URL https://doi.org/10.1145/3545796. 27, 33, 51

[22] Olivier Jeunen, Koen Verstrepen, and B. Goethals. Fair offline evaluation methodologies for implicit-feedback recommender systems with mnar data. In *REVEAL18 Workshop on offline Evaluation for Recommender Systems (RecSys'18)*, 2018. 3

[23] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446. URL https://doi.org/10.1109/TPAMI.2021.3057446. 5, 21

[24] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. doi: 10.1016/J.NEUCOM.2021.10.021. URL https://doi.org/10.1016/j.neucom.2021.10.021. 5

[25] Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. *Recommender Systems Handbook*. Springer US, 2022. ISBN 978-1-0716-2196-7. doi: 10.1007/978-1-0716-2197-4. URL https://doi.org/10.1007/978-1-0716-2197-4. 10, 11, 28, 33, 34, 48

[26] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowl. Based Syst.*, 46:109–132, 2013. doi: 10.1016/j.knosys.2013.03.012. URL https://doi.org/10.1016/j.knosys.2013.03.012. 10, 33

[27] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005. doi: 10.1109/TKDE.2005.99. URL https://doi.org/10.1109/TKDE.2005.99. 10

[28] Özlem Özgöbek, Jon Atle Gulla, and Riza Cenk Erdur. A survey on challenges and methods in news recommendation. In *WEBIST 2014 - Proceedings of the 10th International Conference on Web Information Systems and Technologies, Volume 2, Barcelona, Spain, 3-5 April, 2014*, pages 278–285. SciTePress, 2014. doi: 10.5220/0004844202780285. URL https://doi.org/10.5220/0004844202780285. 10

[29] Shaina Raza and Chen Ding. News recommender system: a review of recent progress, challenges, and opportunities. *Artif. Intell. Rev.*, 55(1):749–800, 2022. doi: 10.1007/s10462-021-10043-x. URL https://doi.org/10.1007/s10462-021-10043-x.

[30] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems - survey and roads ahead. *Inf. Process. Manag.*, 54(6):1203–1227, 2018. doi: 10.1016/j.ipm.2018.04.008. URL https://doi.org/10.1016/j.ipm.2018.04.008. 10

[31] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007. doi: 10.1007/978-3-540-72079-9_10. URL https://doi.org/10.1007/978-3-540-72079-9_10. 10

[32] Ho Chung Wu, Robert Wing Pong Luk, Kam-Fai Wong, and Kui-Lam Kwok. Interpreting TF-IDF term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26(3):13:1–13:37, 2008. doi: 10.1145/1361684.1361686. URL http://doi.acm.org/10.1145/1361684.1361686. 10

[33] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artif. Intell.*, 2009:421425:1–421425:19, 2009. doi: 10.1155/2009/421425. URL https://doi.org/10.1155/2009/421425. 11

[34] Yue Shi, Martha A. Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45, 2014. doi: 10.1145/2556270. URL `https://doi.org/10.1145/2556270`. 11, 33

[35] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 285–295. ACM, 2001. doi: 10.1145/371920.372071. URL `https://doi.org/10.1145/371920.372071`. 12

[36] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW '94, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, October 22-26, 1994*, pages 175–186. ACM, 1994. doi: 10.1145/192844.192905. URL `https://doi.org/10.1145/192844.192905`. 12

[37] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263. URL `https://doi.org/10.1109/MC.2009.263`. 12

[38] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264. Curran Associates, Inc., 2007. URL `https://proceedings.neurips.cc/paper/2007/hash/d7322ed717dedf1eb4e6e52a37ea7bcd-Abstract.html`.

[39] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 556–562. MIT Press, 2000. URL `https://proceedings.neurips.cc/paper/2000/hash/f9d1152547c0bde01830b7e8bd60024c-Abstract.html`.

[40] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 549–553. SIAM, 2006. doi: 10.1137/1.9781611972764.58. URL `https://doi.org/10.1137/1.9781611972764.58`.

[41] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *J. Mach. Learn. Res.*, 13:3619–3622, 2012. doi: 10.5555/2503308.2503357. URL `https://dl.acm.org/doi/10.5555/2503308.2503357`. 12

[42] Steffen Rendle. Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000. IEEE Computer Society, 2010. doi: 10.1109/ICDM.2010.127. URL https://doi.org/10.1109/ICDM.2010.127. 12

[43] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 811–820. ACM, 2010. doi: 10.1145/1772690.1772773. URL https://doi.org/10.1145/1772690.1772773. 13, 18

[44] Zeynep Batmaz, Ali Yürekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artif. Intell. Rev.*, 52(1):1–37, 2019. doi: 10.1007/s10462-018-9654-y. URL https://doi.org/10.1007/s10462-018-9654-y. 13

[45] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1725–1731. ijcai.org, 2017. doi: 10.24963/ijcai.2017/239. URL https://doi.org/10.24963/ijcai.2017/239. 13

[46] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. 1(1), 2023. doi: 10.1145/3568022. URL https://doi.org/10.1145/3568022. 13, 84

[47] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386. URL https://doi.org/10.1109/TNNLS.2020.2978386. 13

[48] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018. doi: 10.1145/3219819.3219890. URL https://doi.org/10.1145/3219819.3219890. 14, 85

[49] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 639–648. ACM, 2020.

doi: 10.1145/3397271.3401063. URL https://doi.org/10.1145/3397271.3401063. 14, 37, 49, 69, 83, 84, 85, 94

[50] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 165–174. ACM, 2019. doi: 10.1145/3331184.3331267. URL https://doi.org/10.1145/3331184.3331267. 14, 83, 84, 85, 94

[51] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. Multi-graph convolution collaborative filtering. In *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pages 1306–1311. IEEE, 2019. doi: 10.1109/ICDM.2019.00165. URL https://doi.org/10.1109/ICDM.2019.00165. 14

[52] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2320–2329. ACM, 2022. doi: 10.1145/3485447.3512104. URL https://doi.org/10.1145/3485447.3512104. 83

[53] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 417–426. ACM, 2019. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488. 14

[54] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, ICUIMC 2008, Suwon, Korea, January 31 - February 01, 2008*, pages 208–211. ACM, 2008. doi: 10.1145/1352793.1352837. URL https://doi.org/10.1145/1352793.1352837. 14

[55] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 253–260. ACM, 2002. doi: 10.1145/564376.564421. URL https://doi.org/10.1145/564376.564421. 14

[56] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *Intell. Data Anal.*, 21(6):1487–1524, 2017. doi: 10.3233/IDA-163209. URL https://doi.org/10.3233/IDA-163209. 14

[57] Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, and Matthew M. Sartin. Combining content-based and collaborative filters

in an online newspaper. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999. 14

[58] Chi-Chih Yu, Toru Yamaguchi, and Yasufumi Takama. A hybrid recommender system based non-common items in social media. In *International Joint Conference on Awareness Science and Technology & Ubi-Media Computing, iCAST 2013 & UMEDIA 2013, Aizuwakamatsu, Japan, November 2-4, 2013*, pages 255–261. IEEE, 2013. doi: 10.1109/ICAwST.2013.6765443. URL https://doi.org/10.1109/ICAwST.2013.6765443. 14

[59] Florian Strub, Romaric Gaudel, and Jérémie Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 11–16. ACM, 2016. doi: 10.1145/2988450.2988456. URL https://doi.org/10.1145/2988450.2988456. 14

[60] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. A hybrid collaborative filtering model with deep structure for recommender systems. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1309–1315. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14676. 14

[61] Xiangwu Meng, Yulu Du, Yujie Zhang, and Xiaofeng Han. A survey of context-aware recommender systems: From an evaluation perspective. *IEEE Trans. Knowl. Data Eng.*, 35(7):6575–6594, 2023. doi: 10.1109/TKDE.2022.3187434. URL https://doi.org/10.1109/TKDE.2022.3187434. 15

[62] Imen Ben Sassi, Sehl Mellouli, and Sadok Ben Yahia. Context-aware recommender systems in mobile environment: On the road of future research. *Inf. Syst.*, 72:27–61, 2017. doi: 10.1016/J.IS.2017.09.001. URL https://doi.org/10.1016/j.is.2017.09.001. 15

[63] Quan Fang, Changsheng Xu, M. Shamim Hossain, and Ghulam Muhammad. STCAPLRS: A spatial-temporal context-aware personalized location recommendation system. *ACM Trans. Intell. Syst. Technol.*, 7(4):59:1–59:30, 2016. doi: 10.1145/2842631. URL https://doi.org/10.1145/2842631. 15

[64] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 831–840. ACM, 2014. doi: 10.1145/2623330.2623638. URL https://doi.org/10.1145/2623330.2623638. 15

[65] Bairan Fu, Wenming Zhang, Guangneng Hu, Xinyu Dai, Shujian Huang, and Jiajun Chen. Dual side deep context-aware modulation for social recommendation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 2524–2534. ACM / IW3C2, 2021.

doi: 10.1145/3442381.3449940. URL https://doi.org/10.1145/3442381.3449940. 15

[66] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 417–426. ACM, 2019. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488. 15

[67] Gediminas Adomavicius and Alexander Tuzhilin. Multidimensional recommender systems: A data warehousing approach. In *Electronic Commerce, Second International Workshop, WELCOM 2001 Heidelberg, Germany, November 16-17, 2001, Proceedings*, volume 2232 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2001. doi: 10.1007/3-540-45598-1_17. URL https://doi.org/10.1007/3-540-45598-1_17. 16

[68] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1): 103–145, 2005. doi: 10.1145/1055709.1055714. URL https://doi.org/10.1145/1055709.1055714. 16

[69] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 447–456. ACM, 2009. doi: 10.1145/1557019.1557072. URL https://doi.org/10.1145/1557019.1557072. 16

[70] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 165–172. ACM, 2011. doi: 10.1145/2043932.2043964. URL https://doi.org/10.1145/2043932.2043964. 16

[71] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. Time matters: Sequential recommendation with complex temporal information. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1459–1468. ACM, 2020. doi: 10.1145/3397271.3401154. URL https://doi.org/10.1145/3397271.3401154. 16, 17

[72] Sicong Xie, Qunwei Li, Weidi Xu, Kaiming Shen, Shaohu Chen, and Wenliang Zhong. Denoising time cycle modeling for recommendation. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 1950–1955. ACM, 2022. doi: 10.1145/3477495.3531785. URL https://doi.org/10.1145/3477495.3531785. 16, 17

[73] Jibang Wu, Renqin Cai, and Hongning Wang. Déjà vu: A contextualized temporal attention mechanism for sequential recommendation. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2199–2209. ACM / IW3C2, 2020. doi: 10.1145/3366423.3380285. URL https://doi.org/10.1145/3366423.3380285. 16, 17

[74] Yuyue Zhao, Xiang Wang, Jiawei Chen, Wei Tang, Yashen Wang, Xiangnan He, and Haiyong Xie. Time-aware path reasoning on knowledge graph for recommendation. *CoRR*, abs/2108.02634, 2021. URL https://arxiv.org/abs/2108.02634. 17

[75] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. Contextual and sequential user embeddings for large-scale music recommendation. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 53–62. ACM, 2020. doi: 10.1145/3383313.3412248. URL https://doi.org/10.1145/3383313.3412248. 17

[76] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6332–6338. ijcai.org, 2019. doi: 10.24963/ijcai.2019/883. URL https://doi.org/10.24963/ijcai.2019/883. 17

[77] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. Deep learning-based sequential recommender systems: Concepts, algorithms, and evaluations. In *Web Engineering - 19th International Conference, ICWE 2019, Daejeon, South Korea, June 11-14, 2019, Proceedings*, volume 11496 of *Lecture Notes in Computer Science*, pages 574–577. Springer, 2019. doi: 10.1007/978-3-030-19274-7_47. URL https://doi.org/10.1007/978-3-030-19274-7_47. 17, 18

[78] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 669–672. IEEE Computer Society, 2002. doi: 10.1109/ICDM.2002.1184025. URL https://doi.org/10.1109/ICDM.2002.1184025. 17

[79] Miki Nakagawa and Bamshad Mobasher. Impact of site characteristics on recommendation models based on association rules and sequential patterns. 2003. 17

[80] Shoujin Wang and Longbing Cao. Inferring implicit rules by learning explicit and hidden item dependency. *IEEE Trans. Syst. Man Cybern. Syst.*, 50(3): 935–946, 2020. doi: 10.1109/TSMC.2017.2768547. URL https://doi.org/10.1109/TSMC.2017.2768547. 17

[81] Ghim-Eng Yap, Xiaoli Li, and Philip S. Yu. Effective next-items recommendation via personalized sequential pattern mining. In *Database Systems for Advanced Applications - 17th International Conference, DASFAA 2012, Busan, South Korea, April 15-19, 2012, Proceedings, Part II*, volume 7239 of *Lecture Notes in Computer Science*, pages 48–64. Springer, 2012. doi: 10.1007/978-3-642-29035-0_4. URL https://doi.org/10.1007/978-3-642-29035-0_4. 17

[82] Ruining He and Julian J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 191–200. IEEE Computer Society, 2016. doi: 10.1109/ICDM.2016.0030. URL https://doi.org/10.1109/ICDM.2016.0030. 18

[83] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06939. 18, 108

[84] Balázs Hidasi and Alexandros 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, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 843–852. ACM, 2018. doi: 10.1145/3269206.3271761. URL https://doi.org/10.1145/3269206.3271761. 18

[85] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1419–1428. ACM, 2017. doi: 10.1145/3132847.3132926. URL https://doi.org/10.1145/3132847.3132926. 18, 40, 108

[86] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 130–137. ACM, 2017. doi: 10.1145/3109859.3109896. URL https://doi.org/10.1145/3109859.3109896. 18

[87] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5941–5948. AAAI Press, 2019.

doi: 10.1609/aaai.v33i01.33015941. URL https://doi.org/10.1609/aaai.v33i01.33015941. 18

[88] Yu Zheng, Chen Gao, Jianxin Chang, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Disentangling long and short-term interests for recommendation. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2256–2267. ACM, 2022. doi: 10.1145/3485447.3512098. URL https://doi.org/10.1145/3485447.3512098. 18

[89] Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 197–206. IEEE Computer Society, 2018. doi: 10.1109/ICDM.2018.00035. URL https://doi.org/10.1109/ICDM.2018.00035. 19, 37, 40, 49, 69

[90] Jiacheng Li, Yujie Wang, and Julian J. McAuley. Time interval aware self-attention for sequential recommendation. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, pages 322–330. ACM, 2020. doi: 10.1145/3336191.3371786. URL https://doi.org/10.1145/3336191.3371786. 19, 69

[91] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S. Yu. Sequential recommendation via stochastic self-attention. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2036–2047. ACM, 2022. doi: 10.1145/3485447.3512077. URL https://doi.org/10.1145/3485447.3512077. 19

[92] Enming Yuan, Wei Guo, Zhicheng He, Huifeng Guo, Chengkai Liu, and Ruiming Tang. Multi-behavior sequential transformer recommender. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 1642–1652. ACM, 2022. doi: 10.1145/3477495.3532023. URL https://doi.org/10.1145/3477495.3532023. 19

[93] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. Denoising self-attentive sequential recommendation. In *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*, pages 92–101. ACM, 2022. doi: 10.1145/3523227.3546788. URL https://doi.org/10.1145/3523227.3546788. 19

[94] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 1441–1450. ACM, 2019.

doi: 10.1145/3357384.3357895. URL https://doi.org/10.1145/3357384.3357895. 19

[95] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 346–353. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.3301346. URL https://doi.org/10.1609/aaai.v33i01.3301346. 19

[96] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. Memory augmented graph neural networks for sequential recommendation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5045–5052. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5945. 19

[97] Tianyu Zhu, Leilei Sun, and Guoqing Chen. Graph-based embedding smoothing for sequential recommendation. *IEEE Trans. Knowl. Data Eng.*, 35(1): 496–508, 2023. doi: 10.1109/TKDE.2021.3073411. URL https://doi.org/10.1109/TKDE.2021.3073411. 19

[98] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 169–178. ACM, 2020. doi: 10.1145/3397271.3401142. URL https://doi.org/10.1145/3397271.3401142. 20

[99] Zihao Li, Xianzhi Wang, Chao Yang, Lina Yao, Julian J. McAuley, and Guandong Xu. Exploiting explicit and implicit item relationships for session-based recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 553–561. ACM, 2023. doi: 10.1145/3539597.3570432. URL https://doi.org/10.1145/3539597.3570432. 20

[100] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. doi: 10.1016/J.NEUCOM.2021.10.021. URL https://doi.org/10.1016/j.neucom.2021.10.021. 21

[101] Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages

6523–6541. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.574. URL https://doi.org/10.18653/v1/2020.coling-main.574. 21

[102] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL http://arxiv.org/abs/1612.00796. 21

[103] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgwNerKvB. 21

[104] Muhammad Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An efficient domain-incremental learning approach to drive in all weather conditions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022*, pages 3000–3010. IEEE, 2022. doi: 10.1109/CVPRW56347.2022.00339. URL https://doi.org/10.1109/CVPRW56347.2022.00339. 21

[105] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey. *CoRR*, abs/2302.03648, 2023. doi: 10.48550/ARXIV.2302.03648. URL https://doi.org/10.48550/arXiv.2302.03648. 21

[106] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11816–11825, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e562cd9c0768d5464b64cf61da7fc6bb-Abstract.html. 22

[107] Jihwan Bang, Heesu Kim, Youngjoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 8218–8227. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00812. URL https://openaccess.thecvf.com/content/CVPR2021/html/Bang_Rainbow_Memory_Continual_Learning_With_a_Memory_of_Diverse_Samples_CVPR_2021_paper.html. 22

[108] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=Sk7KsfW0-`. 22

[109] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=S07feAlQHgM`. 22

[110] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. How to retrain recommender system?: A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1479–1488. ACM, 2020. doi: 10.1145/3397271.3401167. URL `https://doi.org/10.1145/3397271.3401167`. 22, 23, 27, 82, 94

[111] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. Graphsail: Graph structure aware incremental learning for recommender systems. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 2861–2868. ACM, 2020. doi: 10.1145/3340531.3412754. URL `https://doi.org/10.1145/3340531.3412754`. 22, 24, 27, 82, 94

[112] João Vinagre, Alípio Mário Jorge, and João Gama. Evaluation of recommender systems in streaming environments. *CoRR*, abs/1504.08175, 2015. URL `http://arxiv.org/abs/1504.08175`. 23, 27

[113] João Vinagre, Alípio Mário Jorge, Conceição Rocha, and João Gama. Statistically robust evaluation of stream-based recommender systems. *IEEE Trans. Knowl. Data Eng.*, 33(7):2971–2982, 2021. doi: 10.1109/TKDE.2019.2960216. URL `https://doi.org/10.1109/TKDE.2019.2960216`. 23, 27

[114] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. Real-time top-n recommendation in social streams. In Padraig Cunningham, Neil J. Hurley, Ido Guy, and Sarabjot Singh Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 59–66. ACM, 2012. doi: 10.1145/2365952.2365968. URL `https://doi.org/10.1145/2365952.2365968`. 23

[115] Chen Chen, Hongzhi Yin, Junjie Yao, and Bin Cui. Terec: A temporal recommender system over tweet stream. *Proc. VLDB Endow.*, 6(12):1254–1257, 2013. doi: 10.14778/2536274.2536289. URL `http://www.vldb.org/pvldb/vol6/p1254-chen.pdf`. 23

[116] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. Streaming ranking based recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 525–534. ACM, 2018. doi: 10.1145/3209978.3210016. URL https://doi.org/10.1145/3209978.3210016. 23, 82, 96

[117] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1569–1577. ACM, 2019. doi: 10.1145/3292500.3330839. URL https://doi.org/10.1145/3292500.3330839. 23, 27, 82

[118] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. GAG: global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 669–678. ACM, 2020. doi: 10.1145/3397271.3401109. URL https://doi.org/10.1145/3397271.3401109. 23, 82

[119] Sihao Ding, Fuli Feng, Xiangnan He, Yong Liao, Jun Shi, and Yongdong Zhang. Causal incremental graph convolution for recommender system retraining. *CoRR*, abs/2108.06889, 2021. URL https://arxiv.org/abs/2108.06889. 23, 82

[120] Danni Peng, Sinno Jialin Pan, Jie Zhang, and Anxiang Zeng. Learning an adaptive meta model-generator for incrementally updating recommender systems. In *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*, pages 411–421. ACM, 2021. doi: 10.1145/3460231.3474239. URL https://doi.org/10.1145/3460231.3474239. 24, 82, 94

[121] Jiafeng Xia, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. Incremental graph convolutional network for collaborative filtering. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 2170–2179. ACM, 2021. doi: 10.1145/3459637.3482354. URL https://doi.org/10.1145/3459637.3482354. 24

[122] Fei Mi, Xiaoyu Lin, and Boi Faltings. ADER: adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 408–413. ACM, 2020. doi: 10.1145/3383313.3412218. URL https://doi.org/10.1145/3383313.3412218. 24

[123] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. A practical incremental method to train deep CTR models. *CoRR*, abs/2009.02147, 2020. URL `https://arxiv.org/abs/2009.02147`. 24

[124] Renchu Guan, Haoyu Pang, Fausto Giunchiglia, Ximing Li, Xuefeng Yang, and Xiaoyue Feng. Deployable and continuable meta-learning-based recommender system with fast user-incremental updates. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 1423–1433. ACM, 2022. doi: 10.1145/3477495.3531964. URL `https://doi.org/10.1145/3477495.3531964`. 24

[125] Kian Ahrabian, Yishi Xu, Yingxue Zhang, Jiapeng Wu, Yuening Wang, and Mark Coates. Structure aware experience replay for incremental learning in graph-based recommender systems. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 2832–2836. ACM, 2021. doi: 10.1145/3459637.3482193. URL `https://doi.org/10.1145/3459637.3482193`. 24, 81

[126] Yichen Wang, Nan Du, Rakshit Trivedi, and Le Song. Coevolutionary latent feature processes for continuous-time user-item interactions. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4547–4555, 2016. URL `https://proceedings.neurips.cc/paper/2016/hash/53ed35c74a2ec275b837374f04396c03-Abstract.html`. 24

[127] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation, 2017. 24

[128] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1269–1278. ACM, 2019. doi: 10.1145/3292500.3330895. URL `https://doi.org/10.1145/3292500.3330895`. 25

[129] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and Philip S. Yu. Dynamic graph collaborative filtering. In *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, pages 322–331. IEEE, 2020. doi: 10.1109/ICDM50108.2020.00041. URL `https://doi.org/10.1109/ICDM50108.2020.00041`. 25, 85, 90

[130] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. How good your recommender system is? A survey on evaluations in recommendation. *Int. J. Mach. Learn. Cybern.*, 10(5):813–831, 2019. doi: 10.1007/s13042-017-0762-9. URL `https://doi.org/10.1007/s13042-017-0762-9`. 25, 28

[131] Jöran Beel, Marcel Genzmehr, Stefan Langer, Andreas Nürnberger, and Bela Gipp. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation, 2013. URL https://doi.org/10.1145/2532508.2532511. 25

[132] Marco Rossetti, Fabio Stella, and Markus Zanker. Contrasting offline and online results when evaluating recommendation algorithms. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 31–34. ACM, 2016. doi: 10.1145/2959100.2959176. URL https://doi.org/10.1145/2959100.2959176.

[133] Eva Zangerle and Christine Bauer. Evaluating recommender systems: Survey and framework. *ACM Comput. Surv.*, 55(8):170:1–170:38, 2023. doi: 10.1145/3556536. URL https://doi.org/10.1145/3556536. 26, 28, 34, 39

[134] Mingang Chen and Pan Liu. Performance evaluation of recommender systems. *International journal of performability engineering*, 13:1246, 2017. 25, 26

[135] Jöran Beel, Stefan Langer, Marcel Genzmehr, Bela Gipp, Corinna Breitinger, and Andreas Nürnberger. Research paper recommender system evaluation: a quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys 2013, Hong Kong, China, October 12, 2013*, pages 15–22. ACM, 2013. doi: 10.1145/2532508.2532512. URL https://doi.org/10.1145/2532508.2532512. 25, 26, 27, 39

[136] James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu, and Barry Smyth. Evaluating compound critiquing recommenders: a real-user study. In *Proceedings 8th ACM Conference on Electronic Commerce (EC-2007), San Diego, California, USA, June 11-15, 2007*, pages 114–123. ACM, 2007. doi: 10.1145/1250910.1250929. URL https://doi.org/10.1145/1250910.1250929. 25

[137] Jöran Beel and Stefan Langer. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In *Research and Advanced Technology for Digital Libraries - 19th International Conference on Theory and Practice of Digital Libraries, TPDL 2015, Poznań, Poland, September 14-18, 2015. Proceedings*, volume 9316 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 2015. doi: 10.1007/978-3-319-24592-8_12. URL https://doi.org/10.1007/978-3-319-24592-8_12. 25

[138] Henry A Landsberger. Hawthorne revisited: Management and the worker, its critics, and developments in human relations in industry. 1958. 26

[139] Dimitris Paraschakis. Recommender systems from an industrial and ethical perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 463–466.

ACM, 2016. doi: 10.1145/2959100.2959101. URL https://doi.org/10.1145/2959100.2959101. 26, 33

[140] Preetam Nandy, Divya Venugopalan, Chun Lo, and Shaunak Chatterjee. A/B testing for recommender systems in a two-sided marketplace. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6466–6477, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/32e19424b63cc63077a4031b87fb1010-Abstract.html. 26

[141] Tobias Schnabel. Where do we go from here? guidelines for offline recommender evaluation. *CoRR*, abs/2211.01261, 2022. doi: 10.48550/arXiv.2211.01261. URL https://doi.org/10.48550/arXiv.2211.01261. 26

[142] Dietmar Jannach and Christine Bauer. Escaping the mcnamara fallacy: Towards more impactful recommender systems research. *AI Mag.*, 41(4):79–95, 2020. doi: 10.1609/aimag.v41i4.5312. URL https://doi.org/10.1609/aimag.v41i4.5312. 26

[143] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 23–32. ACM, 2020. doi: 10.1145/3383313.3412489. URL https://doi.org/10.1145/3383313.3412489. 27, 30, 33, 40, 42, 45, 47

[144] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2405–2414. ACM, 2021. doi: 10.1145/3404835.3463245. URL https://doi.org/10.1145/3404835.3463245. 33

[145] Alan Said and Alejandro Bellogín. Rival: a toolkit to foster reproducibility in recommender system evaluation. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 371–372. ACM, 2014. doi: 10.1145/2645710.2645712. URL https://doi.org/10.1145/2645710.2645712. 27

[146] João Vinagre, Alípio Mário Jorge, and João Gama. Online bagging for recommender systems. *Expert Syst. J. Knowl. Eng.*, 35(4), 2018. doi: 10.1111/exsy.12303. URL https://doi.org/10.1111/exsy.12303. 27

[147] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans.*

Inf. Syst., 22(1):5–53, 2004. doi: 10.1145/963770.963772. URL https://doi.org/10.1145/963770.963772. 28

[148] Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 213–220. ACM, 2013. doi: 10.1145/2507157.2507160. URL https://doi.org/10.1145/2507157.2507160. 28

[149] Zahid Younas Khan, Zhendong Niu, Sulis Sandiwarno, and Rukundo Prince. Deep learning techniques for rating prediction: a survey of the state-of-the-art. *Artif. Intell. Rev.*, 54(1):95–135, 2021. doi: 10.1007/s10462-020-09892-9. URL https://doi.org/10.1007/s10462-020-09892-9. 28

[150] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. Assessing ranking metrics in top-n recommendation. *Inf. Retr. J.*, 23(4):411–448, 2020. doi: 10.1007/s10791-020-09377-x. URL https://doi.org/10.1007/s10791-020-09377-x. 28

[151] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 39–46. ACM, 2010. doi: 10.1145/1864708.1864721. URL https://doi.org/10.1145/1864708.1864721. 28

[152] Matevz Kunaver and Tomaz Pozrl. Diversity in recommender systems - A survey. *Knowl. Based Syst.*, 123:154–162, 2017. doi: 10.1016/j.knosys.2017.02.009. URL https://doi.org/10.1016/j.knosys.2017.02.009. 29

[153] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, 7(1):2:1–2:42, 2017. doi: 10.1145/2926720. URL https://doi.org/10.1145/2926720.

[154] Saul Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 109–116. ACM, 2011. URL https://dl.acm.org/citation.cfm?id=2043955. 29

[155] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In *New Frontiers in Artificial Intelligence, JSAI 2007 Conference and Workshops, Miyazaki, Japan, June 18-22, 2007, Revised Selected Papers*, volume 4914 of *Lecture Notes in Computer Science*, pages 40–46. Springer, 2007. doi: 10.1007/978-3-540-78197-4_5. URL https://doi.org/10.1007/978-3-540-78197-4_5. 29

[156] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In

Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010, pages 257–260. ACM, 2010. doi: 10.1145/1864708.1864761. URL `https://doi.org/10.1145/1864708.1864761`. 29

[157] Hung-Hsuan Chen, Chu-An Chung, Hsin-Chien Huang, and Wen Tsui. Common pitfalls in training and evaluating recommender systems. *SIGKDD Explor.*, 19(1):37–45, 2017. doi: 10.1145/3137597.3137601. URL `https://doi.org/10.1145/3137597.3137601`. 29

[158] Elisa Mena-Maldonado, Rocío Cañamares, Pablo Castells, Yongli Ren, and Mark Sanderson. Agreement and disagreement between true and false-positive metrics in recommender systems evaluation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 841–850. ACM, 2020. doi: 10.1145/3397271.3401096. URL `https://doi.org/10.1145/3397271.3401096`. 29

[159] Amir Hossein Jadidinejad, Craig Macdonald, and Iadh Ounis. The simpson's paradox in the offline evaluation of recommendation systems. *ACM Trans. Inf. Syst.*, 40(1):4:1–4:22, 2022. doi: 10.1145/3458509. URL `https://doi.org/10.1145/3458509`. 29

[160] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. *Commun. ACM*, 65(7):75–83, 2022. doi: 10.1145/3535335. URL `https://doi.org/10.1145/3535335`. 29, 30, 49, 70, 96

[161] Asela Gunawardana and Guy Shani. Evaluating recommender systems. In *Recommender Systems Handbook*, pages 265–308. 2015. 30

[162] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 129–136. ACM, 2014. doi: 10.1145/2645710.2645746. URL `https://doi.org/10.1145/2645710.2645746`. 30

[163] Pedro G. Campos, Fernando Díez, and Manuel A. Sánchez-Montañés. Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 309–312. ACM, 2011. doi: 10.1145/2043932.2043990. URL `https://doi.org/10.1145/2043932.2043990`. 30, 31, 36

[164] Robin Verachtert, Lien Michiels, and Bart Goethals. Are we forgetting something? correctly evaluate a recommender system with an optimal training window. In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2022 co-located with the 16th ACM Conference on Recommender Systems (RecSys 2022), Seattle, WA, USA, September 22,*

2022, volume 3228 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022. URL https://ceur-ws.org/Vol-3228/paper1.pdf. 31

[165] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1101–1110. ACM, 2020. doi: 10.1145/3397271.3401133. URL https://doi.org/10.1145/3397271.3401133. 31, 36, 40

[166] Xavier Amatriain and Justin Basilico. Past, present, and future of recommender systems: An industry perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 211–214. ACM, 2016. doi: 10.1145/2959100.2959144. URL https://doi.org/10.1145/2959100.2959144. 33

[167] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. Recbole 2.0: Towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 4722–4726. ACM, 2022. doi: 10.1145/3511808.3557680. URL https://doi.org/10.1145/3511808.3557680. 33

[168] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. A re-visit of the popularity baseline in recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1749–1752. ACM, 2020. doi: 10.1145/3397271.3401233. URL https://doi.org/10.1145/3397271.3401233. 33

[169] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *CoRR*, abs/2010.03240, 2020. URL https://arxiv.org/abs/2010.03240. 34

[170] Yang Gao, Yi-Fan Li, Yu Lin, Hang Gao, and Latifur Khan. Deep learning on knowledge graph for recommender system: A survey. *CoRR*, abs/2004.00387, 2020. URL https://arxiv.org/abs/2004.00387. 34

[171] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 10:2935–2962, 2009. doi: 10.5555/1577069.1755883. URL https://dl.acm.org/doi/10.5555/1577069.1755883. 34

[172] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 188–197. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1018. URL https://doi.org/10.18653/v1/D19-1018. 35

[173] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 555–563. ACM, 2019. doi: 10.1145/3289600.3290989. URL https://doi.org/10.1145/3289600.3290989. 35

[174] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press, 2009. URL https://www.auai.org/uai2009/papers/UAI2009_0139_48141db02b9f0b02bc7158819ebfa2c7.pdf. 36, 48, 69, 91

[175] Daniel Woolridge, Sean Wilner, and Madeleine Glick. Sequence or pseudo-sequence? an analysis of sequential recommendation datasets. In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2021 co-located with the 15th ACM Conference on Recommender Systems (RecSys 2021), Amsterdam, The Netherlands, September 25, 2021*, volume 2955 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL https://ceur-ws.org/Vol-2955/paper8.pdf. 42

[176] Jin Yao Chin, Yile Chen, and Gao Cong. The datasets dilemma: How much do we really know about recommendation datasets? In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, pages 141–149. ACM, 2022. doi: 10.1145/3488560.3498519. URL https://doi.org/10.1145/3488560.3498519. 42

[177] Aixin Sun. Take a fresh look at recommender systems from an evaluation standpoint, 2023. 45, 109

[178] Ting Bai, Lixin Zou, Wayne Xin Zhao, Pan Du, Weidong Liu, Jian-Yun Nie, and Ji-Rong Wen. Ctrec: A long-short demands evolution model for continuous-time recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 675–684. ACM, 2019. doi: 10.1145/3331184.3331199. URL https://doi.org/10.1145/3331184.3331199. 49, 70

[179] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on*

Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018, pages 505–514. ACM, 2018. doi: 10.1145/3209978.3210017. URL https://doi.org/10.1145/3209978.3210017. 70

[180] Rajiv Pasricha and Julian J. McAuley. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 63–71. ACM, 2018. doi: 10.1145/3240323.3240356. URL https://doi.org/10.1145/3240323.3240356. 49, 64

[181] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *RecSys*, pages 333–336. ACM, 2011. 49, 96

[182] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*, pages 5329–5336. AAAI Press, 2019. 50, 57

[183] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. A critical study on data leakage in recommender system offline evaluation. *ACM Trans. Inf. Syst.*, 41(3):75:1–75:27, 2023. doi: 10.1145/3569930. URL https://doi.org/10.1145/3569930. 63

[184] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. Do loyal users enjoy better recommendations?: Understanding recommender accuracy from a time perspective. In *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*, pages 92–97. ACM, 2022. doi: 10.1145/3539813.3545124. URL https://doi.org/10.1145/3539813.3545124. 63

[185] Rocío Cañamares, Marcos Redondo, and Pablo Castells. Multi-armed recommender system bandit ensembles. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 432–436. ACM, 2019. doi: 10.1145/3298689.3346984. URL https://doi.org/10.1145/3298689.3346984. 64

[186] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. Collaborative similarity embedding for recommender systems. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2637–2643. ACM, 2019. doi: 10.1145/3308558.3313493. URL https://doi.org/10.1145/3308558.3313493.

[187] Rasaq Otunba, Raimi A. Rufai, and Jessica Lin. MPR: multi-objective pairwise ranking. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 170–178. ACM, 2017. doi: 10.1145/3109859.3109903. URL https://doi.org/10.1145/3109859.3109903.

[188] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 32–36. ACM, 2017. doi: 10.1145/3109859.3109889. URL `https://doi.org/10.1145/3109859.3109889`.

[189] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 297–305. ACM, 2018. doi: 10.1145/3240323.3240361. URL `https://doi.org/10.1145/3240323.3240361`. 64

[190] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 265–274. ACM, 2019. doi: 10.1145/3331184.3331211. URL `https://doi.org/10.1145/3331184.3331211`. 64

[191] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. Sequential scenario-specific meta learner for online recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2895–2904. ACM, 2019. doi: 10.1145/3292500.3330726. URL `https://doi.org/10.1145/3292500.3330726`.

[192] Feng Yuan, Lina Yao, and Boualem Benatallah. Adversarial collaborative auto-encoder for top-n recommendation. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8851902. URL `https://doi.org/10.1109/IJCNN.2019.8851902`. 64

[193] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015*, volume 1388 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. URL `https://ceur-ws.org/Vol-1388/demo_paper1.pdf`. 64

[194] Robin Devooght and Hugues Bersini. Long and short-term recommendations with recurrent neural networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, July 09 - 12, 2017*, pages 13–21. ACM, 2017. doi: 10.1145/3079628.3079670. URL `https://doi.org/10.1145/3079628.3079670`. 64, 84

[195] Michael Hahsler. recommenderlab: An R framework for developing and testing recommendation algorithms. *CoRR*, abs/2205.12371, 2022. doi: 10.

48550/arXiv.2205.12371. URL https://doi.org/10.48550/arXiv.2205.12371. 64

[196] Arthur F. Da Costa, Eduardo P. Fressato, Fernando Soares de Aguiar Neto, Marcelo G. Manzato, and Ricardo J. G. B. Campello. Case recommender: a flexible and extensible python framework for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 494–495. ACM, 2018. doi: 10.1145/3240323.3241611. URL https://doi.org/10.1145/3240323.3241611. 64

[197] Scott Graham, Jun-Ki Min, and Tao Wu. Microsoft recommenders: tools to accelerate developing recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 542–543. ACM, 2019. doi: 10.1145/3298689.3346967. URL https://doi.org/10.1145/3298689.3346967. 64

[198] Dominik Kowald, Simone Kopeinik, and Elisabeth Lex. The tagrec framework as a toolkit for the development of tag-based recommender systems. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, July 09 - 12, 2017*, pages 23–28. ACM, 2017. doi: 10.1145/3099023.3099069. URL https://doi.org/10.1145/3099023.3099069. 64

[199] Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI*, pages 4714–4722. AAAI Press, 2021. 81

[200] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary?: Simple graph contrastive learning for recommendation. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 1294–1303. ACM, 2022. doi: 10.1145/3477495.3531937. URL https://doi.org/10.1145/3477495.3531937. 83

[201] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4264–4270. ijcai.org, 2019. doi: 10.24963/ijcai.2019/592. URL https://doi.org/10.24963/ijcai.2019/592. 85

[202] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil,*

September 22-26, 2020, pages 240–248. ACM, 2020. doi: 10.1145/3383313.3412488. URL https://doi.org/10.1145/3383313.3412488. 91

[203] Tran Khanh Dang, Quang Phu Nguyen, and Van Sinh Nguyen. A study of deep learning-based approaches for session-based recommendation systems. *SN Comput. Sci.*, 1(4):216, 2020. doi: 10.1007/s42979-020-00222-y. URL https://doi.org/10.1007/s42979-020-00222-y. 107

[204] Jiajie Su, Chaochao Chen, Weiming Liu, Fei Wu, Xiaolin Zheng, and Haoming Lyu. Enhancing hierarchy-aware graph networks with deep dual clustering for session-based recommendation. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 165–176. ACM, 2023. doi: 10.1145/3543507.3583247. URL https://doi.org/10.1145/3543507.3583247. 108

[205] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. Diversifying personalized recommendation with user-session context. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1858–1864. ijcai.org, 2017. doi: 10.24963/ijcai.2017/258. URL https://doi.org/10.24963/ijcai.2017/258. 108

[206] Yitong Pang, Lingfei Wu, Qi Shen, Yiming Zhang, Zhihua Wei, Fangli Xu, Ethan Chang, Bo Long, and Jian Pei. Heterogeneous global graph neural networks for personalized session-based recommendation. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, pages 775–783. ACM, 2022. doi: 10.1145/3488560.3498505. URL https://doi.org/10.1145/3488560.3498505. 108

[207] Biswarup Bhattacharya, Iftikhar Burhanuddin, Abhilasha Sancheti, and Kushal Satya. Intent-aware contextual recommendation system. In *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 1–8. IEEE Computer Society, 2017. doi: 10.1109/ICDMW.2017.8. URL https://doi.org/10.1109/ICDMW.2017.8. 109

[208] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Longbing Cao. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3771–3777. ijcai.org, 2019. doi: 10.24963/ijcai.2019/523. URL https://doi.org/10.24963/ijcai.2019/523. 109

[209] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten de Rijke. Improving end-to-end sequential recommendations with intent-aware diversification. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 175–184. ACM, 2020. doi: 10.1145/3340531.3411897. URL https://doi.org/10.1145/3340531.3411897. 109

[210] Peiyan Zhang, Jiayan Guo, Chaozhuo Li, Yueqi Xie, Jaeboum Kim, Yan Zhang, Xing Xie, Haohan Wang, and Sunghun Kim. Efficiently leveraging multi-level user intent for session-based recommendation via atten-mixer network. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 168–176. ACM, 2023. doi: 10.1145/3539597.3570445. URL https://doi.org/10.1145/3539597.3570445. 109