TOWARD SUSTAINABLE RECOMMENDATION SYSTEMS

A Dissertation

by

JIANLING WANG

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	James Caverlee
Committee Members,	Shuiwang Ji
	Ricardo Gutierrez-Osuna
	Irina Gaynanova
Head of Department,	Scott Schaefer

May 2022

Major Subject: Computer Science

ABSTRACT

Recommendation systems are ubiquitous, acting as an essential component in online platforms to help users discover items of interest. For example, streaming services rely on recommendation systems to serve high-quality informational and entertaining content to their users, and e-commerce platforms recommend interesting items to assist customers in making shopping decisions. Furthermore, the algorithms and frameworks driving recommendation systems provide the foundation for new personalized machine learning methods that have wide-ranging impacts.

While successful, many current recommendation systems are fundamentally not sustainable: they focus on short-lived engagement objectives, requiring constant fine-tuning to adapt to the dynamics of evolving systems, or are subject to performance degradation as users and items churn in the system. In this dissertation research, we seek to lay the foundations for a new class of *sustainable recommendation systems*. By sustainable, we mean a recommendation system should be fundamentally long-lived, while enhancing both current and future potential to connect users with interesting content. By building such sustainable recommendation systems, we can continuously improve the user experience and provide a long-lived foundation for ongoing engagement. Building on a large body of work in recommendation systems, with the advance in graph neural networks, and with recent success in meta-learning for ML-based models, this dissertation focuses on sustainability in recommendation systems from the following three perspectives with corresponding contributions:

- *Adaptivity*: The first contribution lies in capturing the temporal effects from the instant shifting of users' preferences to the lifelong evolution of users and items in real-world scenarios, leading to models which are *highly adaptive* to the temporal dynamics present in online platforms and provide improved item recommendation at different timestamps.

- *Resilience*: Secondly, we seek to identify the elite users who act as the "backbone" recommendation systems shape the opinions of other users via their public activities. By investi-

gating the correlation between user's preference on item consumption and their connections to the "backbone", we enable recommendation models to be *resilient to dramatic changes* including churn in new items and users, and frequently updated connections between users in online communities.

- *Robustness*: Finally, we explore the design of a novel framework for "learning-to-adapt" to the imperfect test cases in recommendation systems ranging from cold-start users with few interactions to casual users with low activity levels. Such a model is *robust to the imperfection* in real-world environments, resulting in reliable recommendation to meet user needs and aspirations.

DEDICATION

To my family and my love.

ACKNOWLEDGMENTS

First and foremost, I am sincerely grateful to my advisor Dr. James Caverlee who generously guided me throughout this journey, and carefully protected my curiosity and research interests with his support and encouragement. Besides, I would like to thank the rest of my dissertation committee, Dr. Shuiwang Ji, Dr. Ricardo Gutierrez-Osuna and Dr. Irina Gaynanova, for their continuous advice and support during my Ph.D.

In addition, many thanks to all my collaborators, labmates and colleagues in the past five years. At Texas A&M, I was fortunate to be a member of Infolab and the helpful discussions with my labmates was a strong boost to my research. Another big part of my Ph.D. time was my internships. I would like to thank Dr. Eser Kandogan for his mentorship in IBM, and Dr. Ainur Yessenalina and Alireza Roshan-Ghias for their help and guidance during my internship at Amazon. Also, I am grateful to Dr. Liangjie Hong who offered me the intern opportunity at Etsy and gave me absolute freedom for my research project. Besides, I would like to thank my host Dr. Ya Le at Google Brain for her support and thought-provoking hours of discussions, and thank Dr. Bo Chang, Dr. Yuyan Wang and other team member for their help and inspiration in the summer. Also, I want to express my thank to Dr. Minmin Chen, Dr. Zhiyuan Cheng and Dr. Ed Chi for their support and valuable suggestions about my career.

Also, I would like to thank all the reviewers to the papers I have submitted and every notification letter I have received. I would have quit my Ph.D. without my first paper acceptance from WSDM in 2019. Thanks to the incredible trip to Australia which marked a turning point point in my PhD journey and gave me lots of inspiration and courage. Additionally, I want to sincerely thank my buddy Kaize Ding for his criticism, his encouragement and his tremendous impact on my Ph.D.

Foremost, I would like to express my special thanks to my parents, who create me a safe haven all the time. I would not be here without their endless love, support, and sacrifice. At the bittersweet end to a consequential period of my life, let me also give a round of applause to myself.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor James Caverlee [advisor], Professor Shuiwang Ji and Professor Ricardo Gutierrez-Osuna of the Department of Computer Science and Engineering and Professor Irina Gaynanova of the Department of Statistics. Chapter 3 is conducted partially based on the work done while the student was interning at Etsy. Chapter 5 is conducted partially based on the work done while the student was interning at Google Brain. All work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by NSF grants IIS-1841138.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation and Challenges

Recommendation systems are ubiquitous. For example, e-commerce platforms recommend interesting items to assist customers in making shopping decisions. The recommender systems of many streaming services such as YouTube serve as conduits to high-quality informational and entertaining content. Smart home devices can make recommendations for events, alarm clocks, and home automation. Indeed, recommendation systems have became the essential components in many online platforms to alleviate the severe information overload issue by connecting users with the items of interest (e.g. products, services or social content). Furthermore, the algorithms and frameworks driving recommendation systems provide the foundation for new personalized machine learning methods that have wide-ranging impacts.

A key objective for recommendation systems is in *predicting* users' preferences on item consumption in the future. Generally, users leave feedback on items through different interactions such as ratings, views, clicks or purchases, which can be distilled to reveal their preferences. By leveraging such historic feedback, a variety of recommendation systems [1, 2, 3, 4, 5] have been proposed on top of collaborative filtering to model the compatibility between users and items, and show great success in providing personalized item recommendation. Grounded on the recent advances in deep learning techniques, many neural models that can mine the complex patterns and relationships from graph-structure or sequential interaction data have been developed to capture more informative representations of users and items in recommendation systems [6, 7, 8, 9].

While successful, many current recommendation systems are fundamentally not sustainable: they focus on short-lived engagement objectives, and could suffer from performance degradation in many real-world scenarios characterized by the presence of constant change and imperfection. In this dissertation, we identify three challenges to sustainability:

- **Online dynamics** present in the platforms, ranging from instant shifting to long-term evo-

lution. In the short-term, intrinsic user behavior may be shifted instantly by occasions or special events, leading to interactions that deviate from historic preferences and are not related to recent actions. Meanwhile, users and items display dynamics in their evolution over time: A fantasy reader may progress from the young adult Harry Potter novels to the grittier Game of Thrones. Similarly, items themselves evolve in how they are perceived; for example, iPhone 8 was an up-to-date device at the time it was released but became a budget choice after the new generation came out. Without a long-lived foundation, a recommendation system would require constant fine-tuning to adapt to these dynamics of evolving systems.

- **Dramatic changes** exist in real-world scenarios, e.g., users joining and leaving the community, churn in new items and frequent updates of user relationships, which limit the power of item recommendation. However, even in such changing environments, users are constantly influenced by a small group of "backbone" individuals – Key Opinion Leaders (KOL), who can shape their views, and further impact what items they buy, what media they consume, and how they interact with online platforms [10, 11]. For example, the KOLs on Instagram and Pinterest could influence shopping decisions by highlighting new fashion trends while KOLs on Yelp and TripAdvisor could guide customer restaurant selection by providing explanatory information (like photos and reviews) for restaurants . Such a "backbone" and its influence are ignored in current recommendation frameworks.

- **Imperfect situations** occur and can subvert models which rely on perfect assumptions. Previous research usually assumes that users have interacted with a few items they are interested in, from which the recommendation systems can extract their preferences on item consumption. However, this perfect assumption may not hold in real-world environments. For example, in many real-world scenarios, recommenders may face difficulty in dealing with cold-start users who have only limited interactions with the system, leading to inherently long-tailed interaction data. A model trained predominantly on interaction data from users

with high-activity levels often fail to capture the activity patterns of users with low-activity levels. Failure to handle these imperfect situations (e.g., cold-start, low-activity level user) will push users away from the platforms with unreliable recommendations.

1.2 Dissertation Contributions

In this dissertation research, we seek to lay the foundations for a new class of *sustainable recommendation systems*. By sustainable, we mean a recommendation system should be fundamentally long-lived, while enhancing both current and future potential to connect users with interesting content. By building such sustainable recommendation systems, we can continuously improve the user experience and provide a long-lived foundation for ongoing engagement. Concretely, this dissertation focuses on *sustainability* from three perspectives: **adaptivity**, **resilience** and **robustness**. Inspired by efforts to balance static latent factors with temporal dynamics, with the advance in graph neural networks, and with recent success in meta-learning for ML-based models, this dissertation makes three unique contributions toward sustainable recommendation systems:

- **Adaptivity.** To continuously improve the user experience, a recommendation system should be *highly adaptive to the temporal dynamics* present in these platforms, capable of balancing both the instant shift in user engagement with the lifelong evolution of users and items. Our initial trial focuses on the preference shifting caused by both instant and long-term reoccurring occasions. Based on the observed patterns of shopping occasions and how they can change users' behaviors from both a global and a personal perspective, we propose a novel next-item recommendation system which models a user's default, intrinsic preference, as well as two different kinds of occasion-based signals that may cause users to deviate from their normal behavior. Next, we focus on capturing the change of item semantics over time and across users while modeling users' long-term evolution. With the assumption that the meaning of an item can be revealed by the correlations defined by user interactions in the short term, we present a novel recommendation framework empowered by sequential hypergraphs to distill the short-term item correlations and uncover the long-term dynamic

user preferences.

- **Resilience.** For ongoing engagement, a recommendation system should be *resilient to dramatic changes* including churn in new items and users, and frequently updated connections between users in online communities. We seek to identify the "backbone" of recommendation systems. In the dissertation, we focus on the impact of the small groups of key opinion leaders (KOLs) who are often positioned importantly in the platform (with large numbers of followers) and can wield an outsize influence in the community. With our data analysis in several changing online environments, we observe that KOLs are able to constantly guide their followers' preferences and shape how users view the items. Thus, by identifying the "backbone" KOLs and their influence to the online systems, we propose to improve the recommendation to users by carefully eliciting opinions from KOLs and incorporating them into the collaborative signal diffusion process.

- **Robustness.** A long-lived recommendation system should be *robust to the imperfection* (i.e., cold-start, low-activity level) in real-world environments, providing reliable recommendation to meet users' actual needs and aspirations. Firstly, we explore the challenging problem of sequential recommendation for cold-start users with only minimal logged interactions without relying on auxiliary information. We present a novel "learning-to-learn" paradigm to model the transition patterns of users, which can make fast adaption for cold-start users in inferring their sequential interactions. Secondly, we center around the research problem of distilling informative transition patterns from core users and efficiently adapt to casual users with low activity level in the platforms. We propose a model-agnostic framework to automatically learn a data augmentation policy using REINFORCE and improve the recommendation system using generated augmented data.

1.3 Dissertation Overview

This dissertation presents my research studies about how to learn a recommendation system to continuously improve the user experience and provide a long-lived foundation for ongoing en-

gagement. In chapter 2, we give a brief background introduction on model-based recommendation systems. The remainder of this dissertation is organized as follows:

- **Chapter 3. Adaptivity: Balancing Instant Change and Lifelong Evolution.** In this chapter, we first introduce the challenges and conduct a data analysis over evolution in real-world systems. Then we will present two models based on the motivation: (i) an Occasion-Aware Recommender system aim to model the repeated personal occasion signals with attention layers, while modeling the global occasion signals by memorizing the temporal trends of shopping behaviors; and (ii) a novel end-to-end framework with sequential Hypergraphs to enhance next-item Recommendation, which can generate dynamic item embeddings incorporating the short-term correlations between items. The effectiveness of both models are evaluated through experiments on datasets from the e-commerce sites Amazon and Etsy and the information curation platform Goodreads.

- **Chapter 4. Resilience: Identifying the Resilient Recommender "Backbone".** In this chapter, we start from the data analysis to explore the relationships among users, items and key opinion leaders within real-world online platforms, and uncover the importance of explicitly modeling the influence of KOLs in recommendation systems. Then we elaborate the design of our a novel end-to-end graph-based neural model – GoRec to incorporate the influence of KOLs for recommendation. Specifically, GoRec is able to elicit elite opinions from KOLs and model their diffusion in the community.

- **Chapter 5. Robustness: Learning from Imperfect Environments.** In this chapter, we will first discuss the learning-to-learn framework for improving recommendations for users with only a few interactions (cold-start) in sequential recommendation. Then, motivated by this data-driven analysis, we demonstrate the gap between users of high-activity level and users of low-activity level and how it could degrade the recommendation performance. Finally, we propose a model-agnostic learning-to-augment framework to mitigate this particular imperfect situation by bridging the gap between users of different activity levels.

2. BACKGROUND

In this chapter, we introduce the foundational model of personalized recommendation systems, the loss function and evaluation metrics for different recommendation objectives, and a dynamic setup of recommendation systems focusing on sequential user interactions.

2.1 Modeling Foundation

The principle of generating personalized recommendation is to predict the compatibility between users and items, from which the items with high compatibility to a user can make up the unique recommendation list for the user. Thus, we can define a recommendation model as:

$$\hat{y}_{u,i} = f_{\boldsymbol{\theta}}(u, i), \tag{2.1}$$

in which $\hat{y}_{u,i}$ denotes the predicted compatibility of user u on item i, and $\boldsymbol{\theta}$ represents the set of model parameters.

The design of a successful recommendation system relies on the selection of the function $f_{\boldsymbol{\theta}}$. Collaborative Filtering (CF) is the basic foundation for many personalized recommendation systems. Generally, it encompasses techniques for making recommendation to users based on someone with the similar tastes to themselves. As one of the most effective variants of matrix factorization (MF) [12], latent factor model-based recommendation has attracted lots of attention due to the Netflix Prize. The high level idea of latent factor models is to approximate compatibility between a user and an item with the dot product of the corresponding latent factor vectors. Given that $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ denote the latent factor vector for user u on item i, a latent factor model will calculate the compatibility via

$$\hat{y}_{u,i} = \boldsymbol{v}_u^T \boldsymbol{v}_i, \tag{2.2}$$

in which the bias terms are modeled with a constant padded into the latent factors [13]. Most of the widely-adopted recommendation frameworks [13, 12, 7] can be regarded as a generalization of

the latent factor model. A key question for us is how to learn informative latent factors for both users and items to model their compatibility accurately.

2.2 Inference Objective

Recommendation models are trained on users' feedback interactions in history (i.e., training samples) and aim to accurately infer the future interactions. In many online platforms, there exist both *explicit* feedback signals (e.g., ratings and thumbs ups/downs) which can directly reveal the preference levels of users and implicit feedback signals (e.g., views, clicks, purchases) which indirectly reflects users' preference.

2.2.1 Rating-based Objective

One line of research focuses on extracting users' preference from the explicit feedback and formulate recommendation as a rating prediction problem.

Loss Function. With the rating-based objective, $\hat{y}_{u,i}$ is used to predict the explicit rating user u left on item i. During the training process, the objective is to minimize the rating prediction error for all the training samples. The Mean Squared Error (MSE) loss function is calculated S_{train} within the set of training samples and can be denoted as

$$\sum_{\boldsymbol{S}_{train}} (r_{u,i} - \hat{y}_{u,i})^2. \tag{2.3}$$

Evaluation Metrics. To examine the test performance, we usually adopt Root Mean Square Error (RMSE) to evaluate our recommendation system, which is widely used in related work for user-item rating prediction and recommendation [14, 15]. For item and user pairs (i, u) in test set $\boldsymbol{S}_{test}$, we denote the ground truth rating as r_{iu} and the predicted rating as $\hat{y}_{u,i}$. Then RMSE is calculated as:

$$\sqrt{\frac{1}{|\boldsymbol{S}_{test}|} \sum_{(i,u)\in\boldsymbol{S}_{test}} (r_{u,i} - \hat{y}_{u,i})^2}. \tag{2.4}$$

And smaller RMSE means the rating prediction is more accurate.

2.2.2 Pairwise Ranking-based Objective

Compared to the explicit ratings, implicit feedback is widely available and much easier to collect, which can help to alleviate the data sparsity issue in recommendation systems. With the historic implicit feedback (i.e., views), we can assume that if user u views item i, then u is interested in i. However, if u does not view item j, we cannot conclude that u is not interested in j because it is also possible that u is unaware of j. Hence, to overcome this implicit feedback challenge, we usually adopt the *pairwise ranking-based objective* and assume that the user prefers the positive item i over all other non-viewed items.

Loss Function. With the pairwise-ranking based objective, we usually pair each of the positive user-item pair with a negative item, which can be sampled from the non-viewed items. Suppose that user u has already viewed item i and hasn't viewed j yet, we assume u prefers i over j and construct a tuple in the form of (u, i, j). As in Bayesian Personalized Recommendation (BPR) [4], the goal is to maximize the gap between the ground truth positive user-item pair and negative sampled pairs with the loss function as follows:

$$\sum_{(u,i,j)\in \boldsymbol{S}_{train}} - \ln \sigma(\hat{y}_{u,i} - \hat{y}_{u,j}) + \lambda ||\theta||^2. \tag{2.5}$$

where $||\theta||^2$ is a regularization term and $\sigma(\cdot)$ is the Sigmoid function. Meanwhile, the cross-entropy loss can also be adopted for the pairwise-ranking based objective with the following loss function:

$$\ell(\theta) = \sum_{(u,i,j)\in \boldsymbol{S}_{train}} - \left[\log(\sigma(\hat{y}_{u,i})) + \log(1 - \sigma(\hat{y}_{u,j}))\right]. \tag{2.6}$$

Evaluation Metrics. Under the implicit feedback setup and ranking-based objective, the recommendation consists of the items with the Top-K predicted scores $\hat{y}$. We adopt the Precision, Recall, F1 score and NDCG of Top-K recommendation as metrics. Let $\mathbf{I}_u$ represent the set of items with implicit feedback by user u in test data and $\mathbf{Rec}_u^K$ is the Top-K recommendation for u. Then for

user u, we have:

$$Recall@K = \frac{|\mathbf{I}_u \cap \mathbf{Rec}_u^K|}{|\mathbf{I}_u|} \quad Precision@K = \frac{|\mathbf{I}_u \cap \mathbf{Rec}_u^K|}{K}$$

Precision@k represents the percentage of correctly predicted items among the Top-k recommendations, and Recall@k represents the fraction of relevant items which are discovered by the Top-k recommendations. We also consider both recall and precision with their harmonic mean with the F1 score,

$$F1@k = \frac{2 \cdot Precision@k \cdot Recall@k}{Precision@k + Recall@k}.$$

In addition, to gain more insights on the rankings of the Top-K recommendation, NDCG is also widely adopted in evaluating a recommendation system. NDCG is the ratio between discounted cumulative gain (DCG) and ideal discounted cumulative Gain (IDCG):

$$DCG@K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)} \qquad IDCG@K = \sum_{i=1}^{|REL|} \frac{rel_i}{\log_2(i+1)},$$

in which rel_i denotes the relevance score the recommendation with rank i. If the recommendation is in the test set, then $rel_i = 1$, otherwise, $rel_i = 0$. $|REL|$ represents the size of the test set. Then NDCG is calculated as:

$$NDCG@K = \frac{DCG@K}{IDCG@K}.$$

2.3 A Dynamic Setup – Sequential Recommendation

Instead of treating users as static, sequential recommendation aims to capture the sequential patterns from historical user interactions and infer the interesting items based on users' dynamic preferences. Early works propose to leverage Markov chains (MC) to model the transition among items and predict the subsequent interaction [16, 17]. To handle more complex sequential signals, grounded on the recent advances of deep learning techniques, many neural models that can produce informative representations of users' interaction sequences have been developed to obtain the

dynamic user states in sequential recommendation. There are lots of efforts on Recurrent Neural Networks (RNNs) to investigate users' sequential interactions [6, 9, 18, 19]. As an extension to GRU4Rec [6], which directly generates the session embeddings with a Gated Recurrent Neural Network (GRU), GRU4Rec+ [19] develops a new class of loss functions for the Top-K recommendation problem. Meanwhile, Convolutional Neural Networks (CNN)-based recommenders also show superior performance. Caser [20] is built on top of the 2D convolutional sequence embedding model and NextitNet [21] investigates 1D CNNs with the dilated convolution filters for better performance. With its success in handling the textual data, self-attention layer (transformer) [22] is adopted in SASRec [23] and Bert4Rec [24] to generate dynamic user embedding based on their interaction sequences. More recently, Graph Neural Networks have been exploited in [25, 26, 27] to encode the contextual information for more accurate user modeling in sequential recommendation.

Problem Definition. We use $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$ to denote the set of $|\mathcal{U}|$ users and the set of $|\mathcal{I}|$ items on the platform. Given the sequence of items that user u has interacted with in chronological order $\mathbf{S}^u = [i_1^u, i_2^u, \ldots, i_p^u, \ldots, i_n^u]$, where i_p^u represents the pth item u interacted with, the objective of a sequential recommendation model is to infer the next interesting item i_{n+1}^u for user u.

Time-dependent Sequential Recommendation. Additionally, we will need the timestamp information while exploring the temporal dynamics in the real-world systems. Under such a time-dependent sequential recommendation setup, let $\mathbf{T} = \{t_1, t_2, ..., t_M\}$ be the set of timestamps, which can be days, weeks or months in a calendar year. We sort the set of products user u has purchased in chronological order as $\mathbf{L}^u = ((i_1^u, t_1^u), (i_2^u, t_2^u), ..., (i_{|\mathbf{L}^u|}^u, t_{|\mathbf{L}^u|}^u))$. Each pair ($i_n^u$, t_n^u), $n \in [1, |\mathbf{L}^u|]$ denotes that user u purchases product p_n^u at time t_n^u. In real-world applications, we want to predict what a user want to purchase when he/she starts a (shopping) session at a future timestamp and the goal of time-dependent sequential recommendation is to generate a list of top-k interesting items for user u at a future timestamp $t_{|\mathbf{L}^u|+1}^u$.

3. ADAPTIVITY: BALANCING INSTANT CHANGE AND LIFELONG EVOLUTION[1]

3.1 Introduction

A sustainable recommendation system should be highly adaptive to the temporal dynamics present in a platform, capable of balancing both the instant shift in user engagement with the lifelong evolution of users and items. To handle the complex situation where user preferences can develop and change along time, recent efforts have focused on modeling users in a dynamic manner, which can adjust the recommendation based on the sequential behaviors of users [28, 29]. They either rely on the sequential transition between recent purchases [16, 30] or model the intrinsic preferences of users with different neural structures based on their historic sequential behaviors [20, 21, 23]. However, these models may lead to poor predictive power in dynamic real-world scenarios with both instant shifting and long-term evolution.

In this chapter, we will motivate the challenge of adaptivity by showing evidence of different occasions that may instantly shift users' preferences and the dynamic patterns of items from both short-term and long-term perspectives. Based on the observed phenomenons, we discuss our efforts on two different framework: (i) an occasion-aware recommender system aim to model the repeated personal occasion signals with attention layers, while modeling the global occasion signals by memorizing the temporal trends of shopping behaviors; and (ii) a novel end-to-end framework with sequential hypergraphs to enhance next-item Recommendation, which can generate dynamic item embeddings incorporating the short-term correlations between items. We showcase the experiment results of the proposed framework following their methodology explanation respectively.

[1]Reprinted with permission from "Time to Shop for Valentine's Day: Shopping Occasions and Sequential Recommendation in E-commerce" by Jianling Wang, Raphael Louca, Diane Hu, Caitlin Cellier, James Caverlee and Liangjie Hong, 2020. Proceedings of the 13th International Conference on Web Search and Data Mining. Copyright 2020 by ACM; "Next-item Recommendation with Sequential Hypergraphs" by Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu and James Caverlee, 2020. Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. Copyright 2020 by ACM.

3.2 Motivation

To motivate the problem, we conduct an initial investigation with data sampled from three online platforms – the e-commerce sites Amazon and Etsy and the information sharing platform Goodreads. We will show evidence of both instant shifting and long-term dynamics existing in those real-world environments.

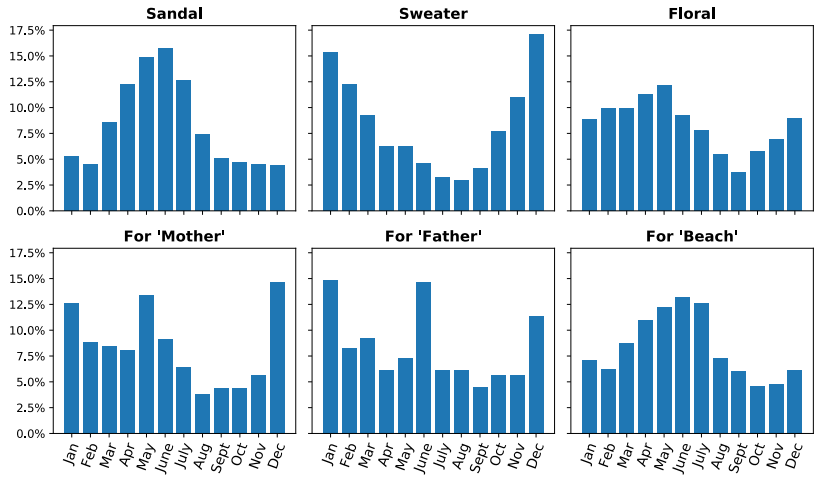

Figure 3.1: Users' shopping preferences are dynamic and can reflect reoccurring occasions (festivals, holidays, seasonal activities).

Shopping Occasions and Instant Preference Shifting. In the public Amazon dataset [31], we can roughly infer users' shopping occasions or intentions with keywords that were mentioned in the reviews. Thus we summarize the occurrences of different keywords over different calendar months and show several examples in Figure 3.1 of possible occasion influences and how they change with time. For example, in the summer, users are likely to look for sandals instead of sweaters, while floral items are more popular in the spring and summer. As for gifting, we find that people tend to purchase for their mothers for Mother's Day (happening in May) or for Christmas. While approaching Father's Day, purchases peak in June, at which time people tend to purchase gifts for their fathers. We can conclude that users have changing preferences within a year for different

occasions (festivals, holidays, seasonal activities) and crowds of users tend to purchase related items during similar occasions. This analysis shows that capturing shopping trends as a function of time and season is useful for understanding purchase preference. We can detect occasion-based shopping trends from crowd behavior.

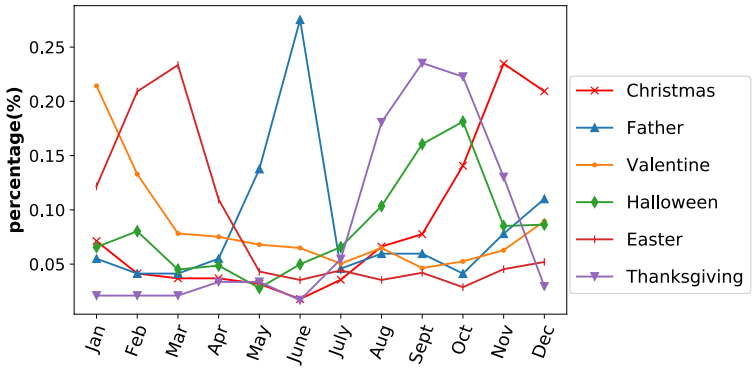

Figure 3.2: The reasons an infant's items shopper changes his/her shopping behaviors.

In addition, there may be occasions which may or may not be related to trending behavior, but can reoccur for individual users. For example, a user may look for birthday gifts for a parent every year as the birthday is approaching. These reoccurring occasions may lead to similar shopping behaviors across years, which we define as *personal occasions*. We want to explore the patterns of these personal occasions and whether they are traceable. In Figure 3.2, we focus on users who purchase items for "infants/newborns/toddlers" in more than 50% of their transactions. We can assume that "buying products for infants" is their intrinsic preference for shopping, which are not related to occasions. Then, we summarize the tags/occasions of their "abnormal" purchases, e.g. the transactions without any infants items. While deviating from their intrinsic preference, these users tend to shop for Father's Day around June and Valentine's in January. Those occasions may reoccur each year and influence their purchase preference at a similar timestamp each year. Additionally, we find that preparation time for different occasions can vary. Users tend to start shopping for Christmas earlier than Valentine's or Father's Day. From a personal perspective,

each user can deviate instantly from their intrinsic preference and desire for different occasions. It is important to capture these personal occasion signals and adjust the recommendation when the reoccurring occasions is approaching.

The Dynamic Patterns of Items Along Time. Then, we retrieve the Bestsellers (i.e., products ranked in the top 1% of purchases) on Amazon in each month from 2001 to 2005. We then calculate the Jaccard Similarity between the list of Bestsellers of each month with the Bestsellers after 1 month, 2 months, 3 months, 8 months, 1 year or more. In Figure 3.3 (a), as illustrated by the blue line, the intersection of Bestsellers between consecutive months is only around 30%. And there is little overlap between the list of Bestsellers after a gap of 6 months (with Jaccard similarity less than 10%). While the popularity of an item can reflect how the community views the item, the change in the list of Bestsellers along time indicates that the meaning of items in the community can change along time.

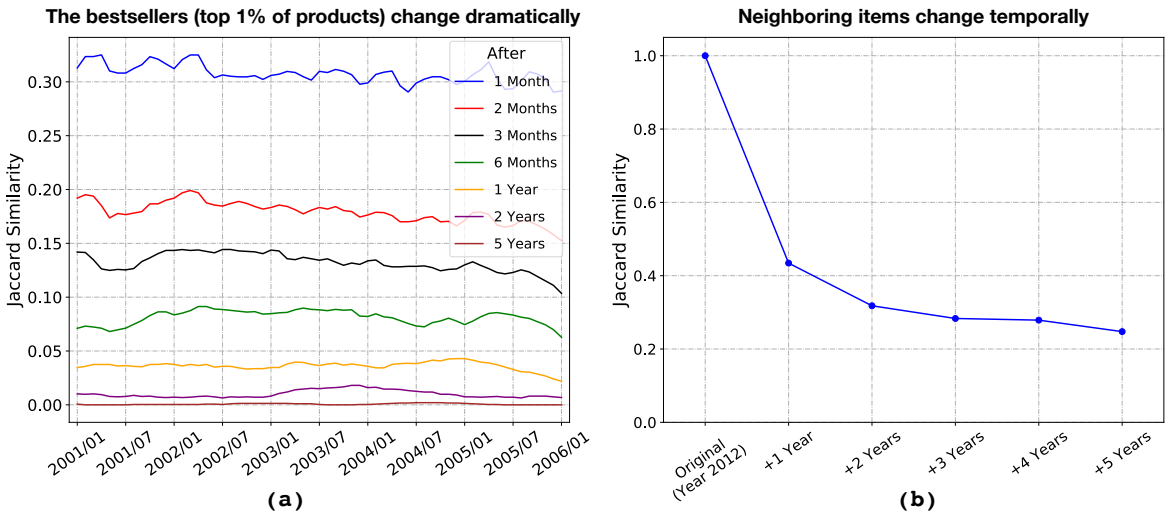

Figure 3.3: (a) The overlap of monthly Bestsellers in Amazon decreases as the time gap grows larger (i.e., from 1 month to 5 years). (b) The neighboring books (books with large co-occurrence) on Goodreads are changing as time goes on.

Finally, we turn to the items in Goodreads, a platform in which users share their thoughts on

books. Each user has a sequence of items that the user has interacted with via rating, tagging or commenting in chronological order. We split the sequences of items the users have interacted with based on the timestamps (by year) and train different item embedding models with sequences in different years. Following the idea in [32, 33, 18], we adopt word2vec [34] to generate embeddings of books based on the co-occurrence of items (i.e., books read by a user consequently). Based on these embeddings, we find the Top-10 neighbors of each book in different years. Then we calculate the Jaccard similarity between neighbors of each book in 2012 with its neighbors in 1 to 5 years later and show the average results in Figure 3.3 (b). We find that the similarity between neighbors in 2012 and 2013 for books is 40% and the similarity keeps decreasing as the time gap become larger. That is, the relationships between items are changing along time and the variations are larger the longer the time gap.

In summary, it is important to capture the reoccurring occasion signals that can instantly change users' preference for continuously improving personalized recommendation. Meanwhile, as relationships between items are changing from the long-term perspective, leading to the change in the semantic meanings of items, we are motivated to exploit the short-term correlations between items while modeling their dynamic patterns for a sustainable recommendation recommendation.

3.3 Occasion Signals and Preference Shifting in Recommendation Systems

Most sequence-based recommendation models aim to predict a user's next actions (e.g. next purchase) based on their past actions. These models either capture users' intrinsic preference (e.g. a comedy lover, or a fan of fantasy) from their long-term behavior patterns or infer their current needs by emphasizing recent actions. in e-commerce, users' shopping decisions can also be influenced by different occasions that lead to behavior which is not related to their recent actions or long-term intrinsic preferences. For example, a user who buys a pair of sandals in June would not want to be recommended an item for "Summer vacation" during the user's next shopping session in December. A "boho" style lover may purchase clothes or accessories that match her style, however, she may occasionally purchase a birthday gift for a friend whose style is not "boho". Previous works assuming that users' actions are coherent or change smoothly along time can not handle

such scenarios where users' behaviors can also be driven by different occasions. As illustrated in Figure 3.4, a mom who frequently buys clothing for her infant will look for Christmas decorations near Christmas. A buyer who routinely purchases crochet supplies may purchase a birthday gift for her son every year. User behavior in E-commerce is not always related to their recent actions or long-term intrinsic preferences, as assumed by many previous sequential recommendation systems.

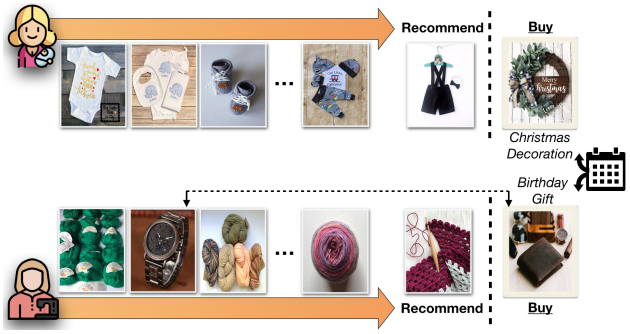

Figure 3.4: Example of Occasion-driven Purchases.

Concretely, an *occasion* is a particular time or instance of an event that causes or triggers a purchase. There are *global occasions* which happen at the same time for a large number of users; examples include festivals or celebrations (like Christmas, Valentines' Day, Mother's Day) or seasonal events (like buying a snowboard in the Winter and a surfboard in the Summer). These global occasions are able to encourage or lead to similar shopping decisions for crowds of users. On the other hand, there are also *personal occasions*, which may happen at different timestamps for different users; examples include birthdays (for themselves or friends) and anniversaries. Those occasions usually occur in a periodic and repeated pattern for a specific user.

It is important to exploit the linkage between different occasions and shopping behaviors in e-commerce, so that we can: (i) recommend more time or season-aware candidates (like recommending a surfboard in the Summer while recommending snowboard in the Winter), which may alleviate the cold-start problem; (ii) reduce the noise in modeling users' intrinsic preferences since occasion-driven purchases (like gifts for others) may show different patterns compared to normal

purchases from the same users; (iii) recommend relevant items to the user for upcoming reoccurring occasions. (Though the user may not purchase the exact same item for a reoccurring event, like consecutive Mother's Days events, the items purchased for Mother's Day previously will likely be related.)

There are several key challenges with using occasion signals in recommendation systems: (i) Are there traceable patterns distinguishing different occasions that we can use to holistically model a user's preference? (ii) Can we capture reoccurring shopping trends based on large crowd behavior? (iii) Can we model a flexible time-window for when occasions may reoccur? (iv) Can we properly balance a user's intrinsic preference versus the impact of a particular occasion in order to accurately predict their next purchase? Solutions to these challenges lead to a novel recommendation framework.

Contributions. Building on top of related efforts on modeling temporal effects and dynamic user modeling in E-commerce, in Section 3.3.2, we propose to model the repeated personal occasion signals with attention layers, while modeling the global occasion signals by memorizing the temporal trends of shopping behaviors. With a gating component, we balance global and local effects of different occasions and propose *OAR* – an ***O***ccasion-***A***ware ***R***ecommender system for e-commerce while centering around each user's intrinsic preferences. In Section 3.3.3, we conduct extensive experiments on real-world datasets from Etsy and Amazon and find that the proposed *OAR* outperforms the state-of-the-art approach in sequential recommendation.

3.3.1 Related Work

There are works which have been done on dynamic user modeling considering the temporal effects. The work in [35] proposes to divide the long time series into slices and training for different latent representations at each slice in TimeSVD++ [35]. Utilizing the explicit time stamp, in [9, 18], they use parallel RRN structure to model the dynamics of users and items simultaneously. The work in [28] explores how users' shopping decisions can be influenced by the life-stage along time, and proposes to select corresponding recommendation model after labeling consumer's life-stage.

While focusing on the sequential behavior patterns of users in e-commerce, there are previous

works assuming that a user would behave centering around the intense shopping intent and tend to interact with the exact same items repeatedly [36, 37]. RepeatNet [38] predicts the probability of being repeated for a user at each timestamp, and then decide whether to recommend from the purchased items or new items. In [39], they model the repeat consumption of different products with Hawkes Process and integrate the resulting signals into Collaborative Filtering to generate recommendations. However, these models can not be generalized to many shopping platforms where a user seldom purchases the exact same item repeatedly (like clothes, accessories and books).

There are also works trying to capture both the long-term dynamics and short-term effects simultaneously building on top of hierarchical structures. HRNN [40] consists of a two-layer hierarchical RNN, which learns the representation for each short-term session with a lower layer RNN and then aggregates the resulting outputs from the same user with a higher layer RNN. The work of [41] achieves a similar goal with hierarchical attention layers. HPMN [29] is proposed to model the periodic patterns of users with a hierarchical recurrent memory network. Although these methods can model the dynamic users preferences, they do not take the influence of different occasions into consideration.

3.3.2 Methodology

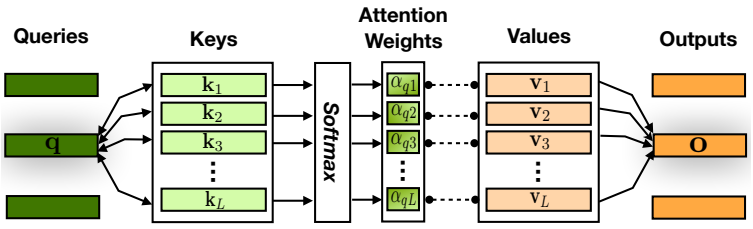

Figure 3.5: Attention Module.

Preliminary: Attention Mechanism. To provide accurate recommendation, our goal is to understand how to aggregate the purchase record of a user in the past to infer the user's preferences in a future timestamp. The neural attention mechanism [22] can be applied to capture the correlation

between the target query (recent purchased items or the future timestamp for prediction) and the context contents (purchase history). For different types of attention modules, the input usually consists of a *Query*, and *Key-Value* pairs. The goal is to map the query with a set of key-value pairs to generate the output (as shown in Figure 3.5). An attention module can be divided into two steps. The first step entails computing the relationship/similarity scores between the *query* and a set of *keys*, which are used as the attention weights to aggregate the corresponding set of *values* [22]. Mathematically, given the input query $\mathbf{q}$ and a set of key-value pairs $\mathbf{P} = \{(\mathbf{k}_l, \mathbf{v}_l) \mid l \in [1, L]\}$, the resulted output $\mathbf{o}$ is calculated as:

$$\mathbf{o} = \sum_{l=1}^{L} \alpha_{ql} \mathbf{v}_l, \quad where \quad \alpha_{ql} = \frac{\exp(s(\mathbf{q}, \mathbf{k}_l))}{\sum_{l=1}^{L} \exp(s(\mathbf{q}, \mathbf{k}_l))} \tag{3.1}$$

where $s(\cdot, \cdot)$ is the similarity scoring function used to calculate the correlation between a query and a key. Based on our analysis in Section 3.2, we propose to make use of the attention mechanism for computing user profiling by taking different types of occasion signals into consideration, in addition to their intrinsic preferences. Next, we will explain the details of each component in our OAR model (in Figure 3.6) through a discussion on the *design for query, key-value pair and the appropriate weight scoring function* to answer the following questions:

- **RQ1:** How to utilize the correlation between recent and historic purchased items to identify a user's intrinsic preferences which are mainly driven by a user's personal taste and self-desire?
- **RQ2:** How to model and predict user preferences for reoccurring personal occasions by tracing their personal shopping history?
- **RQ3:** How to memorize the crowd behavior at different time periods and perform dynamic mapping to aggregate the relevant global occasion signals?
- **RQ4:** How can we fuse intrinsic user preferences and different types of occasion signals to obtain a complete user profile that will inform what to recommend next?

RQ1: Intrinsic Preference Modeling. Users' intrinsic preferences on items are comparatively stable or change smoothly [42]. Thus previous works in recommendation usually model users in a static way with collaborative filtering-based methods [12, 43, 5, 13], or in a dynamics way by capturing the behavior patterns with the chronological order of user-item interactions via Markov Chains [16], RNNs [40, 6, 9], and CNNs [20, 21]. Recently, self-attention [22], has demonstrated its effectiveness in sequential recommendation by capturing both the long-term semantics and relevant items with the recent interactions [23]. In a similar way, we try to model users' dynamic intrinsic preferences based on the correlation between the most recent purchase and the personal historic purchases.

Given the sequence of products user u has purchased $\mathbf{P}^u = (i_1^u, i_2^u, ..., i_{|P^u|}^u)$ in chronological order, we use the combination $\mathbf{m}_{i_d^u} = \mathbf{e}_{i_d^u} + \mathbf{x}_d$ to represent the item at position d (the dth item in the sequence). Here, $\mathbf{e}_{i_d^u}$ is the embedding for item i_d^u and $\mathbf{x}_d$ is the positional embedding of position d, which is used to retain the order information. Self-attention [22] is designed to match a sequence against itself and thus uses the same objects as the queries, keys and values. In our case, we will map the query item p_d^u to the sequence of items $(i_1^u, i_2^u, ..., i_d^u)$, which have been purchased by u no later than i_d^u. Before calculating the attention weights and aggregation, we conduct linear projections for each $\mathbf{m}_{i_d^u}$, $i_d^u \in \mathbf{P}^u$ with matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$ to generate embedding $\hat{\mathbf{m}}_{i_d^u}^Q = \mathbf{m}_{i_d^u}\mathbf{W}^Q$, $\hat{\mathbf{m}}_{i_d^u}^K = \mathbf{m}_{i_d^u}\mathbf{W}^K$, $\hat{\mathbf{m}}_{i_d^u}^V = \mathbf{m}_{i_d^u}\mathbf{W}^V$ for queries, keys and values correspondingly. Thus, we have:

$$\begin{gathered} \textit{Query} : \hat{\mathbf{m}}_{i_d^u}^Q \qquad \textit{Scoring} : s(\mathbf{q}, \mathbf{k}_j) = \frac{\mathbf{q}\mathbf{k}_j^T}{\sqrt{D}} \\ \textit{(Key, Value)} : (\hat{\mathbf{m}}_{i_1^u}^K, \hat{\mathbf{m}}_{i_1^u}^V), (\hat{\mathbf{m}}_{i_2^u}^K, \hat{\mathbf{m}}_{i_2^u}^V), ..., (\hat{\mathbf{m}}_{i_d^u}^K, \hat{\mathbf{m}}_{i_d^u}^V) \end{gathered} \tag{3.2}$$

Here we adopt the scaled dot-product to calculate the score between keys and queries. D denotes the dimension of the embedding. The output $\mathbf{o}_{u,t_{d+1}^u}^I$ based on the most recent item i_d^u can represent the dynamic intrinsic preference of user u after purchasing i_d^u, and will be used to infer the user's next purchase at t_{d+1}^u.

RQ2: Personal Occasion Elicitation. Users can deviate from their intrinsic preferences because of some personally reoccurring occasions. For each user, the shopping behaviors driven by the

same personal occasions are likely to fall into a small time window. For example, a user will often purchase a birthday gift two to three weeks in advance of the birthday. Thus while predicting a user's preference, we also need to elicit the personal occasion signal by tracing the user's previous shopping behavior in the neighboring days. In this component, we want to map the upcoming timestamp (query) with the timestamps of the user's previous purchases (keys) and the corresponding items (values). Given the $\mathbf{L}^u = ((i_1^u, t_1^u), (i_2^u, t_2^u), ..., (i_{|\mathbf{L}^u|}^u, t_{|\mathbf{L}^u|}^u))$, we use $\mathbf{t}_{t_d^u}$ to denote the embedding of timestamp t_d^u. As in modeling intrinsic preference with attention mechanism, the time embedding for queries or keys will be multiplied with the matrices $\mathbf{W}^{Q'}$ and $\mathbf{W}^{K'}$ respectively, with $\hat{\mathbf{t}}^{Q'}_{t^u_{d+1}} = \mathbf{t}_{t^u_{d+1}}\mathbf{W}^{Q'}$ and $\hat{\mathbf{t}}^{K'}_{t^u_d} = \mathbf{t}_{t^u_d}\mathbf{W}^{K'}$. We also apply linear projection for the item embedding with $\mathbf{W}^{V'}$ to generate the embedding for values. While predicting for u at a future time t^u_{d+1}, the personal occasion preference can be obtained with the attention operation below:

$$\textit{Query} : \hat{\mathbf{t}}^{Q'}_{t^u_{d+1}} \qquad \textit{(Key, Value)} : (\hat{\mathbf{t}}^{K'}_{t^u_1}, \hat{\mathbf{e}}^{V'}_{i^u_1}), (\hat{\mathbf{t}}^{K'}_{t^u_2}, \hat{\mathbf{e}}^{V'}_{i^u_2}), ..., (\hat{\mathbf{t}}^{K'}_{t^u_d}, \hat{\mathbf{e}}^{V'}_{i^u_d})$$

We use the same similarity function $s(\cdot, \cdot)$ as in Equation 3.2. While generating the output $\mathbf{o}^P_{u,t^u_{d+1}}$, items which were purchased a long time ago but within a small time window with the query's upcoming timestamp can also get high attention from the model. In this way, *OAR* can capture personally reoccurring occasions.

RQ3: Global Occasion Memorization. By only tracing the personal purchase history, the model is still unable to predict upcoming global occasions. However, these occasion signals can be captured from the behaviors of the crowd from a neighboring time period in the past. Under a global occasion, the crowd of users tends to have similar purchases, like shopping for costumes before Halloween or green shirts near St Patrick's day. We aim to memorize the shopping behaviors of the crowd under different global occasions, which can be used to enrich the preference representation of individual users when a certain occasion is coming. Following a similar idea as in the key-value memory network [44], we use the timestamps as keys and pair each of the keys with a memory slot to represent preferences of the crowd at the timestamp.

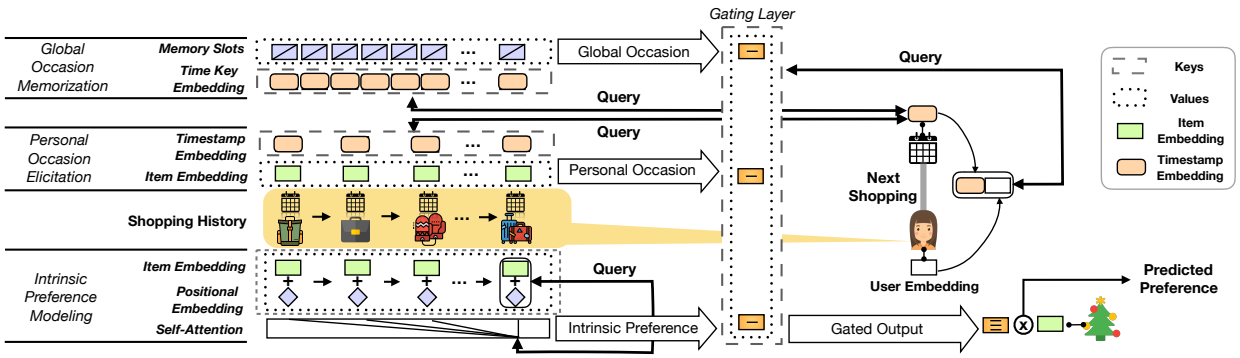

Figure 3.6: The proposed *Occasion-Aware Recommendation* (OAR) model.

Let $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_M\}$ denote the set of embedding for each timestamp. We use $\hat{\mathbf{t}}_i, i \in [1, M]$, which is the projected version of the embedding for timestamp $\hat{\mathbf{t}}_i = \mathbf{t}_i \mathbf{W}^{K''}$, to be the key. Furthermore, we set a separate memory slot $\mathbf{r}_i, i \in [1, M]$ to store global behaviors. Given a query timestamp, we will multiply its embedding $\mathbf{t}_{t^u_{d+1}}$ with matrix $\mathbf{W}^{Q''}$ to get $\hat{\mathbf{t}}^{Q''}_{t^u_{d+1}}$. Then we want to map it with all the key-value memory slots to get the corresponding global occasion representation. We still use scaled dot-product as the similarity scoring function $s(\cdot, \cdot)$ and:

$$\textit{Query} : \hat{\mathbf{t}}^{Q''}_{t^u_{d+1}} \qquad \textit{(Key, Value)} : (\hat{\mathbf{t}}_1, \mathbf{r}_1), (\hat{\mathbf{t}}_2, \mathbf{r}_2), ..., (\hat{\mathbf{t}}_M, \mathbf{r}_M)$$

The output $\mathbf{o}^G_{t^u_{d+1}}$ of the attention operation can be the representation of global occasions at t^u_{d+1}.

RQ4: Gating Layer. Lastly, we discuss how to balance a user's intrinsic preferences with occasion signals for personalization? Here we turn to an attention (gating) layer which can control how we assign different weights to each of the components we have developed in the previous sections. The query will be a user-timestamp pair because the status for a user at different timestamps will be different. For example, there are users who have strong personal desire for handcrafted supplies and seldom purchase other items on a site like Etsy. Or users may tend to be influenced by their surroundings in December but may stick to their own intrinsic preference in June. While predicting

for user u at timestamp t^u_{d+1}, with embedding $\mathbf{u}$ and $\mathbf{t}^u_{d+1}$ for u and t^u_{d+1}:

$$\textit{Query} : \mathbf{u}||\mathbf{t}^u_{d+1} \qquad \textit{Scoring} : s(\mathbf{q}, \mathbf{k}_j) = \mathbf{a}^T tanh(\mathbf{W}[\mathbf{q}||\mathbf{k}_j])$$

$$\textit{(Key, Value)} : (\mathbf{o}^I_{u,t^u_{d+1}}, \mathbf{o}^I_{u,t^u_{d+1}}), (\mathbf{o}^P_{u,t^u_{d+1}}, \mathbf{o}^P_{u,t^u_{d+1}}), (\mathbf{o}^G_{t^u_{d+1}}, \mathbf{o}^G_{t^u_{d+1}})$$

in which $||$ denotes concatenation, and $\mathbf{a}$ and $\mathbf{W}$ represent the transform vector and matrix, respectively, for this additive attention operation. Thus we get the output $\mathbf{o}_{u,t^u_{d+1}}$, which can be used to accurately represent u's preference at future timestamp t^u_{d+1}.

Prediction and Loss. After generating $\mathbf{o}_{u,t^u_{d+1}}$ as the complete representation of a user's current status, we can predict the preference score on item i with $\bar{y}^{t^u_{d+1}}_{u,i} = \mathbf{o}^T_{u,t^u_{d+1}} \mathbf{e}_i$. We modify the pairwise loss in Equation 2.5 to maximize the gap between the ground truth positive user-item pair and negative sampled pairs given the timestamp. The loss function is:

$$L = \sum_{(u,t,i,j)\in\mathbf{D}} - \ln \sigma(\bar{y}^t_{u,i} - \bar{y}^t_{u,j}) + \lambda||\theta||^2 \tag{3.3}$$

where $||\theta||^2$ is a regularization term and $\sigma(\cdot)$ is the Sigmoid function. Each element (u, t, i, j) in the training data set $\mathbf{D}$ is generated by combining the ground truth interaction pair (u, t, i), which means u purchased i at t, with a negative sampled item j that u did not purchase at time t.

3.3.3 Experiment

Datasets. To avoid data leakage while modeling the crowd behaviors in the global occasion component, we split the datasets for training and testing with a cutting time. We only use data before the cutting date to train the model. In both datasets, we keep users who purchased at least twice after the cutting time, so that we can use the first purchase of each user after the cutting date as a validation case and the second purchase as a test case. The detailed information is summarized in Table 3.1. In the experiments, we consider each day in the calender year as a timestamp, that is $t_1 \in \mathbf{T}$ means the first day in a year (January 1).

We conduct experiments on two real-world datasets from e-commerce platforms: (i) *Etsy:* the

Dataset	#Users	#Items	#Purchases	Density	Cutting Time
Amazon	84,191	100,946	1.0M	0.0124%	2013/8/1
Etsy	118,668	80,214	5.3M	0.0561%	2018/1/1

Table 3.1: Dataset Statistics.

purchase data ranges from November 2006 to December 2018 in Etsy. We filter out users with fewer than 5 purchases before the cutting time. To examine the long-term effects, we keep only users who are active for at least two years, requiring that the time gap between their last purchase and their first purchase be larger than 365 days; (ii) *Amazon:* we test over a public Amazon review dataset [31], containing product reviews from May, 1996 to July, 2014. We treat each review as a purchase record and use the time they input the review to approximate the purchase time. We filter out users who purchased fewer than 5 items before the cutting time.

Baselines. We compare the proposed OAR with the following baseline methods:

- **MP:** *Most Popular*. It ranks all the products based on their overall popularity and recommends the most popular products.
- **MF-BPR:** *Matrix Factorization with Bayesian Personalized Ranking* [4]. This model predicts user's preference on a product based on the multiplication between their latent factors (MF) and is optimized with Bayesian personalized ranking (BPR) loss.
- **Fossil:** *Fusing Similarity Models with Markov Chains* [17]. It improves the method of factorizing personalized Markov Chain (FPMC) with item similarity-based algorithm (FISM) to capture the long-term and short-term dynamics of users simultaneously.
- **GRU4Rec+**: *Recurrent Neural Networks with Top-k Gains* [19]. It is similar to GRU4Rec [6] in utilizing GRU model to capture the sequential patterns, but with a modified loss function and sampling strategy to achieve better performance in the Top-K recommendation task.
- **TCN**: *A Simple Convolutional Generative Network for Next Item Recommendation* [21].

This is an improved dilated convolution neural network (CNN) modeling both short and long-range item dependencies in a sequence to recommend the next item.

- **HPMN**: *Lifelong Sequential Modeling with Hierarchical Periodic Memory Network* [29]. It capture the multi-scale sequential patterns of users in e-commerce with a hierarchical and periodical updating mechanism. It is able to model users' periodic behavior patterns appearing in both long-term or short-term.

- **SARec**: *Self-attentive sequential recommendation* [23] With the self-attention layers, this model is able to balance the long-term effect of a sequence and from recent products.

Overall Model Comparison. We summarize the best performance of all the baseline models and the proposed model in Table 3.2. We can see that OAR achieves the best performance under different metrics in both datasets. It gains 7.62% and 6.06% MRR improvement in Etsy and Amazon compared with the state-of-the-art.

Compared with the basic general MP, we can see that MF-BPR which represents users and items with static latent factors can achieve a 177.43% and 39.67% improvement on average in Etsy and Amazon. Then by introducing the Markov Chains to capture the transition of users among different items, we find that Fossil works better than MF in Etsy but performs worse in Amazon. Presumably it is because the Amazon data is extremely sparse and results in an unstable factorized Markov Chains component in Fossil.

Comparing the recent neural-based sequential models, we find that GRU4Rec+ works slightly better than TCN, which is based on dilated CNN. And HPMN utilizing hierarchical multi-layer memory networks outperforms GRU4Rec+ in both data, which proves that there are periodic pattern in users' shopping behaviors. However, HPMN model assumes that the period of shopping behavior is constant for all the users along the time and thus lack of flexibility to handle the real-world scenarios. We find that SARec, which is utilizing self-attention to model users' intrinsic preference, works even better than HPMN. This shows that attention mechanisms are a good fit for modeling sequential behaviors. And by carefully eliciting the occasion signals and combining

them with the intrinsic preferences, OAR achieves the best performance in the next-item prediction via an accurate user model.

Evaluation of OAR. To examine whether each component in OAR achieves its goal and to understand how it contributes to the recommendation, we analyze their impacts with an ablation test (in Table 3.3).

The Global occasion component (G), in which we set up a certain number of memory slots to record the crowd behavior in different occasions, does not provide personalized recommendation individually. It can outperform the general Most Popular (MP) model by 17.17% and 4.89% in Etsy and Amazon, which demonstrates that it can capture the temporal global occasion signals hidden in the crowd behavior. Additionally, we can infer that the users in Etsy are more likely to follow the temporal global trends in shopping. Both Intrinsic and Personal components can provide personalized next-item recommendation. In Intrinsic (I), it maps the most recent purchase to the items purchased before to infer the "relevant" items in the future. While in Personal (P), the main idea is to trace back to the previous behaviors in the related time periods. We can see that P performs slightly better than I, which means that in e-commerce, it is important to predict the shopping occasion and pay more attention to the items purchased around similar occasions while inferring the next purchase. While combining the I and G or I and P, we can see the joint models can improve each of the individual components. Thus we find that in e-commerce platforms, it is necessary to take the occasion signals into consideration while making recommendations.

To examine the impact of the gating component, which is designed for a personalized and temporal-aware fusing of intrinsic preference and the occasion signals, we replace it with a simple addition layer. That is we use the addition among $\mathbf{o}^{I}_{u,t^{u}_{d+1}}$, $\mathbf{o}^{P}_{u,t^{u}_{d+1}}$ and $\mathbf{o}^{G}_{t^{u}_{d+1}}$ as a representation of user u at time t^{u}_{d+1} while removing the Gating Layer. We find that there is a large drop in recommendation quality, which supports the assumption that the influence of different occasions does vary for different users at different timestamps. Thus, it is important to take the personalization and temporal information into consideration simultaneously while utilizing the occasion signals.

Model	*Etsy*					*Amazon*				
	NDCG		HR		MRR	NDCG		HR		MRR
	K=5	K=10	K=5	K=10		K=5	K=10	K=5	K=10	
MP	0.1531	0.1919	0.2304	0.3511	0.1673	0.2129	0.2509	0.3020	0.4199	0.2195
MF-BPR	0.4519	0.5001	0.5947	0.7434	0.4376	0.2663	0.3012	0.3619	0.4698	0.2668
Fossil	0.4946	0.5354	0.5511	0.7630	0.4746	0.2160	0.2483	0.2967	0.3969	0.2221
TCN	0.5199	0.5726	0.6698	0.8059	0.5090	0.2632	0.3029	0.3664	0.4893	0.2650
GRU4Rec+	0.5346	0.5771	0.6830	0.8136	0.5126	0.2763	0.3169	0.3828	0.5087	0.2770
HPMN	0.5480	0.5883	0.6962	0.8201	0.5245	0.2820	0.3216	0.3881	0.5109	0.2819
SARec	0.5665	0.6047	0.7102	0.8278	0.5433	0.3009	0.3385	0.4085	0.5251	0.2984
OAR	**0.6078***	**0.6415***	**0.7425***	**0.8462***	**0.5847***	**0.3200***	**0.3580***	**0.4301***	**0.5476***	**0.3165***

Table 3.2: Comparison of Different Models. $*$ indicates that the improvement of the best result is statistically significant compared with second best result with $p < 0.01$.

Model	Etsy		Amazon	
	NDCG@5	MRR	NDCG@5	MRR
Global (G)	0.1816	0.1953	0.2238	0.2294
Intrinsic (I)	0.5665	0.5433	0.3009	0.2984
Personal (P)	0.5791	0.5582	0.3069	0.3047
I + G	0.5885	0.5642	0.3099	0.3063
I + P	0.5916	0.5677	0.3136	0.3108
Remove Gate	0.5859	0.5618	0.3074	0.3039
OAR	**0.6078**	**0.5847**	**0.3200**	**0.3165**

Table 3.3: Ablation Test Results for OAR.

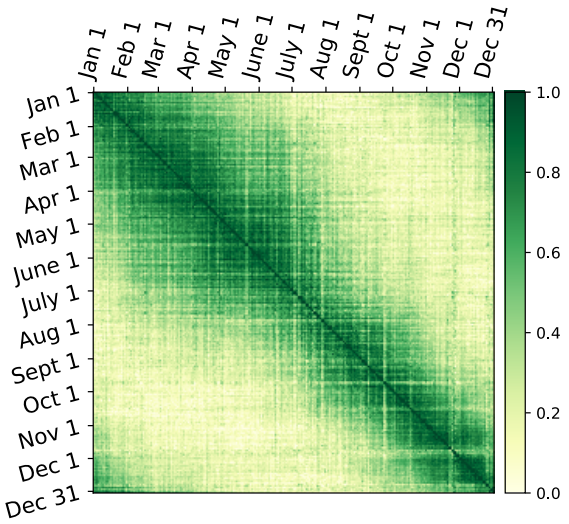

Figure 3.7: Similarity between different calendar days.

Case Study – Temporal Information and Occasions. To examine whether the proposed model is able to capture the occasions by linking the neighboring time periods to the relevant occasions, we plot out the attention weights (or similarity scores) between each timestamp (in Figure 3.7). Near the diagonal (similarity between the exact same timestamp), we can see there are many dark regions, indicating the strong correlation between nearby time periods. For some of the regions, the dark color diffuses to a large area (like around March and April), meaning that the occasions at that time have a continuous lasting influence. Since the occasion calendar is a loop, there is high correlation between dates in December and dates in January, which results in the dark region at

the left bottom and right top corner. Thus we find that *OAR* is capable in modeling the occasion signals along time.

Case Study – Visualization of Occasion-driven Purchases in Etsy. To explore whether *OAR* captures the occasion signals to adjust the recommendation at different timestamps, we predict users' preferences on several items every day in the test year. We calculate and plot out the average predicted preference scores for all the users in Figure 3.8. We find that the preferences for the hooded scarf drop down when the weather gets warmer and increases in Fall and Winter time, while the preferences for shorts are in a totally opposite pattern. And for the Christmas decoration tab (red line), the preferences on it reach the peak in early December but drop down rapidly after Christmas, meaning the product is sensitive to the occasion. However, for items which are fit for occasions that can happen all year round (like birthdays), the average preference on it is flat during the year.

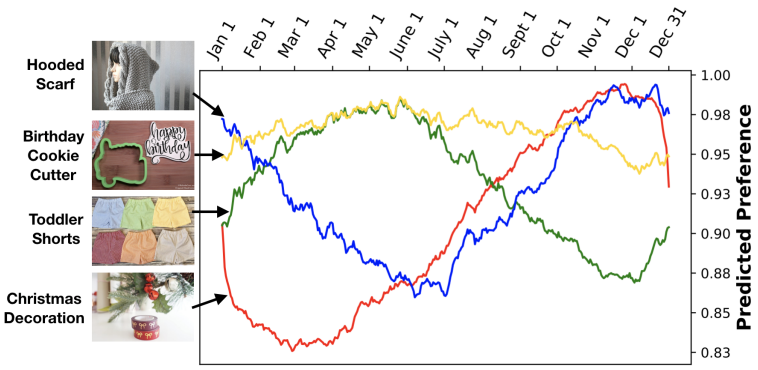

Figure 3.8: The average preferences predicted by OAR.

Case Study – How Occasion Signals compensate the intrinsic preference.

We show the results for an Etsy user as an example (in Figure 3.9) to examine how the occasion signals supplement the intrinsic preference for improved recommendation. In intrinsic preference modeling, a high score will be assigned to the most recent purchase (shorts for the 6-month) and items relevant to that (baby's clothing). Thus, while predicting for August 31 with the intrinsic

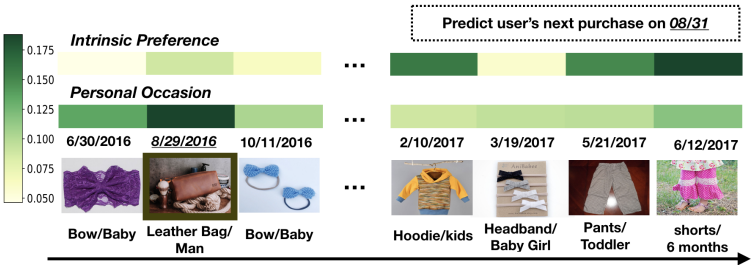

Figure 3.9: The attention weights by different components in OAR.

preference individually, we will keep recommending similar items. However, in personal occasion elicitation, it traces the history and assigns high score to items which are purchased in the related time windows. So that in this case, though the user purchased lots of baby clothing, by capturing the occasion signals, the "leather bag for man" purchased on August 29 two years ago still receives high attention. We can see that *OAR* is able to recall the purchase for the leather bag and thus recommend some related items for the upcoming occasion.

3.4 Long-term Evolution and Short-term Correlation in Recommendation Systems

In Section 3.3, we focus on modeling the instant preference shifting cased by different occasions and events. Motivated by Section 3.2, in this section, we turn to handling the critical issue in designing a sustainable recommendation system – *how items are treated* in models aiming to infer the dynamic user preferences with sequential user interactions. Specifically, for a certain time period in next-item recommendation, we adopt the view that *the meaning of an item can be revealed by the correlations defined by user interactions in the short term.* As shown in Figure 4.1, the iPhone 8 was purchased together with several other up-to-date devices at the time it was released (like a Nintendo Switch) in 2017, indicating that it was a hot new technology item at that time. Once a new version is released in 2019 like the iPhone 11, the iPhone 8 becomes a budget choice since it may be purchased with other devices that are also budget-priced (e.g., the Lite version of the Nintendo Switch or early generation AirPods). In the same way, we can infer that the bouquet purchased by User A was for a wedding since she also purchased items typically associated with weddings. To capture these changes in item semantics, we propose to model such short-term item

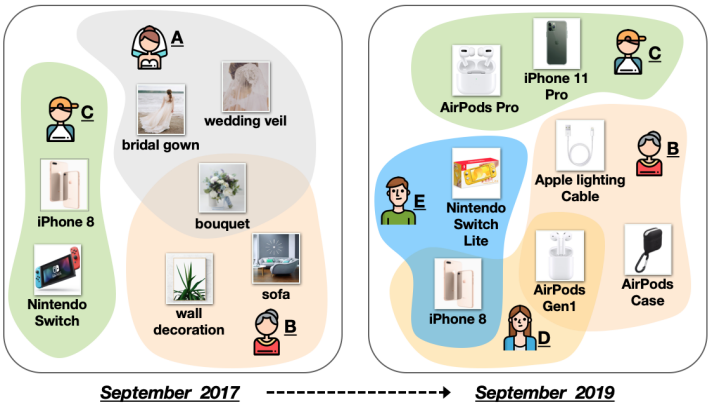

Figure 3.10: The meaning of an item at a certain time period can be revealed by the correlations defined by user interactions in the short term. And the meaning of an item can change over time and change across users. Such dynamics can help to uncover the preference patterns of users.

correlations in a *hypergraph* [45, 46], in which each *hyperedge* can connect multiple nodes on a single edge. In this regard, while each node in the hypergraph denotes an item, a hyperedge can connect the set of items a user interacts with in the short time period altogether.

However, it is non-trivial to extract expressive item semantics from the item-correlation hypergraph. On the one hand, the item correlations encoded by the hyperedges are no longer dyadic (pairwise), but rather triadic, tetradic or of a higher-order. Such complex relationships cannot be handled by conventional methods since they only focus on pairwise associations; on the other hand, the item semantics could propagate over multiple hops. For example, in Figure 1 (Sept 2019), though not purchased by the same user, the iPhone 8 is also related to the Apple Lightning cable with a 2-hop connection to it. Thus it necessitates a design to effectively exploit the hypergraph for learning expressive item semantics.

Furthermore, how to capture the dynamic meanings of items is another challenge for next-item recommendation, since the semantics of an item can change *over time* and *across users*. And such change can help to uncover the preference patterns of users. As illustrated in Figure 1, User C purchasing the iPhone 8 in 2017 gives evidence that User C chases the latest devices; whereas User D purchasing the iPhone 8 in 2019 indicates that User D is looking for a deal. Although the item is the same in both cases, the fundamental semantics of the iPhone 8 have changed. Even at a

single timepoint, an item can carry different meanings to different users. For example, a bouquet of flowers for User B in Figure 1 can reflect home decoration, whereas the same bouquet for User A can reflect a wedding. Though there are previous works in next-item recommendation treating items as dynamic [47, 18, 9], they usually model the variation of an item as a function of time. How to capture the aforementioned two perspectives – change over time and change across users – is vital for high-quality next-item recommendations, but still remains to be explored.

To tackle the aforementioned challenges, we propose *HyperRec*, a novel end-to-end framework with sequential ***Hyper***graphs to enhance next-item ***Rec***ommendation. To untangle the short-term correlations at different time periods, HyperRec truncates the user interactions based on the timestamp to construct a series of hypergraphs. With a hypergraph convolutional network (HGCN), HyperRec is able to aggregate the correlated items with direct or high-order connections to generate the dynamic embedding at each time period. To model the influence of item embeddings in the past time periods, we develop a residual gating layer to combine the dynamic item embeddings of the previous time period with the static item embeddings to generate the input for the HGCN. With change happening both over time and across users, the resulting embeddings from the HGCN will be fed into a fusion layer to generate the final representation for each specific user-item interaction incorporating both the dynamic item embedding and short-term user intent. In personalized next-item recommendation, the dynamic user preferences can be inferred from the sequence of interactions from the user. Thus we use a self-attention layer to capture the dynamic user patterns from the interaction sequences. While predicting a user's preference on an item, both the static and the most recent dynamic item embedding are considered.

Contributions. We develop a novel next-item recommendation framework with sequential hypergraphs to generate dynamic item embeddings incorporating the short-term correlations between items. Two of the unique aspects of the framework are a residual gating layer to control the residual information from the past, and a fusion layer to encode each interaction with both the dynamic item embedding and short-term user intent for sequential pattern modeling. With extensive experiments on datasets covering different online platforms including ecommerce websites (Amazon

and Etsy) and an information sharing community (Goodreads), the proposed model outperforms state-of-the-art models in providing Top-K next-item recommendation.

3.4.1 Related Work

There is an increasing attention on exploiting graph structures for various recommendation scenarios with the recent advance in neural graph embedding algorithms [48, 49, 50, 51, 52]. Many of these works make use of the high-order connections in a static graph to generate enriched latent representations for users or items. In social recommendation, social connections between users can be investigated with GNN to model the propagation of user preference in social networks [53, 54, 55]. Differently, PinSage [48] proposes to generate item embeddings on a graph constructed with item-item connections, which can be applied for downstream recommendation. In addition, there are also works focusing on the user-item interaction graph [56, 7] in which they construct a static graph connecting users and items based on their interaction. However, these methods are not designed for capturing the sequential patterns in recommendation systems.

To model the temporally dynamic patterns and predict for future behaviors, Session-based Temporal Graph (STG) [57] is proposed to connect users, items and sessions in a graph. With random walk process starting from different type of nodes (user/session), it is able to model users' long-term and short-term preferences for recommendation. The work of [58] consists of an RNN to capture dynamic user behaviors and a graph attention layer to model the social influence on a static user-user graph. SR-GNN [25] proposes to construct various graphs of items with session sequences and use GNN to extract item co-occurrences from those session graphs. It generates next-click prediction based on attentive aggregation of item embedding in a session.

As a generalization of the ordinary graph in which each hyperedge can encode the correlations among various numbers of objects, hypergraph has been adopted to unify various types of contents for context-aware recommendation. In terms of modeling the correlations among various types of objects, there are early efforts [59, 60, 61, 62] in applying hypergraphs to assist conventional collaborative filtering for incorporating context information. In [59], in order to integrate both the social relationships and music contents for music recommendation, they propose to use hypergraph

to model the relations among various types of objects (e.g., users, groups, music tracks, tags, albums) in music social communities. Similarly, the work of [60] models the correlations among readers, articles, entities and topics with a hypergraph for personalized news recommendation. These methods are designed based on the properties of the specific communities and can not be easily generalized to the task of next-item recommendation.

3.4.2 Methodology

Since the items purchased by a user in a short time period are correlated, it is vital to define appropriate connections among them. While users may interact with various numbers of items, the conventional graph structure usually only supports pairwise relations between items and is not fit for this case. Thus, we propose to model such short-term correlations with a hypergraph [45, 46], in which multiple items can be connected with one *hyperedge*. For the example in Figure 4.1, the hypergraph for Sept 2017 consists of 7 nodes (items) with 3 hyperedges. The three items purchased by User A are linked together by one hyperedge. Furthermore, besides the direct connections in the hypergraph, the high-order connections between items can also hint on their correlations. For example, in Figure 1 (Sept 2019), though not purchased by the same user, the iPhone 8 is also related to the Apple Lightning cable with a 2-hop connection. With a hypergraph convolutional network (HGCN), we can exploit both the direct and high-order connections to extract the short-term correlations between items. Meanwhile, an item should not be treated as discrete at different time periods, since its features in the past can hint on its features in the future. For example, although the iPhone 8 has fundamentally changed in meaning from 2017 to 2019 in Figure 4.1, the representation in 2019 should inherit some of the characteristics of the iPhone's representation in 2017. In the following, with hypergraph as a principled topology structure, we will discuss about how to effectively generate such dynamic item representations considering both the item correlations in the short term and the connections among different time periods.

Short-term Hypergraphs. To capture the item correlations for different time periods, we can split the user-item interactions into multiple subsets based on the timestamps. Let $\mathbf{G} = \{\mathbf{G}^{t_1}, \mathbf{G}^{t_2}, ..., \mathbf{G}^{t_Q}\}$ represent a series of hypergraphs. $\mathbf{G}^{t_n} = (\mathcal{V}^{t_n}, \mathcal{E}^{t_n}, \mathbf{W}^{t_n}, \mathbf{H}^{t_n})$ is constructed based on all

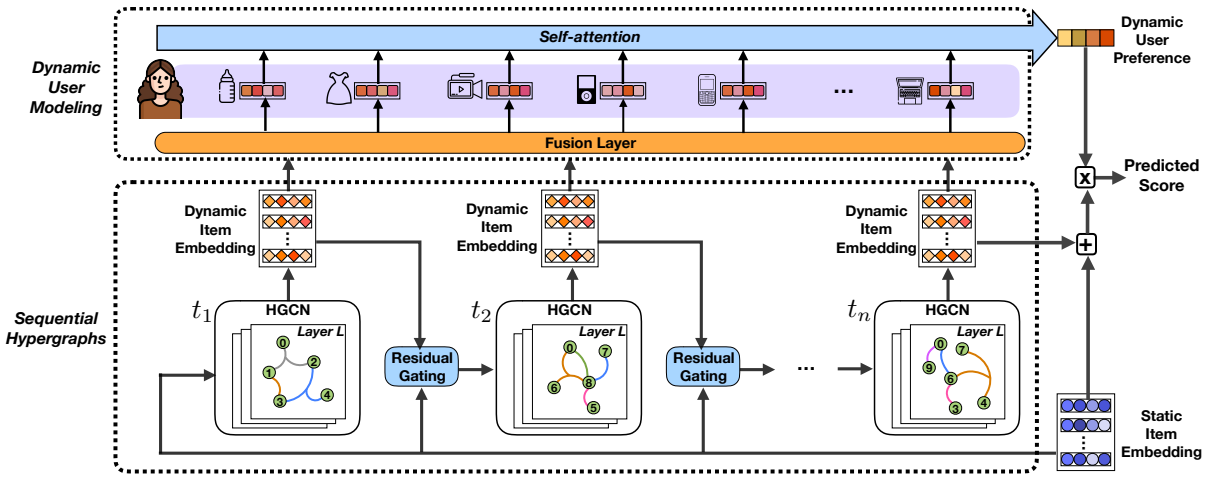

Figure 3.11: The structure of HyperRec.

the user-item interactions happening during time period t_n. $\mathcal{V}^{t_n} \subset \mathbf{I}$ represents the set of nodes in $\mathbf{G}^{t_n}$, that is all the items with interactions in t_n. And $\mathcal{E}^{t_n} \subset \mathbf{U}$ denotes the set of hyperedges, which is similar as all the users who have interactions during t_n. Each $\mathbf{G}^{t_n} \in \mathbf{G}$ is associated with an incidence matrix $\mathbf{H}^{t_n}$ of size $|\mathcal{V}^{t_n}| \times |\mathcal{E}^{t_n}|$. It is also associated with a matrix $\mathbf{W}^{t_n}$, which is a diagonal matrix with $W^{t_n}_{\epsilon\epsilon}$ representing the weight of the hyperedge ϵ. In this work, we let all the hyperedges share the same weights and let $W^{t_n}_{\epsilon\epsilon} = 1, \forall \epsilon \in \mathcal{E}^{t_n}$. When $\upsilon \in \mathcal{V}^{t_n}$ is incident with edge ϵ during time period t_n (i.e., user ϵ purchased υ at t_n), we have $H^{t_n}_{\upsilon\epsilon} = 1$, otherwise $H^{t_n}_{\upsilon\epsilon} = 0$. $\mathbf{D}^{t_n}$ and $\mathbf{B}^{t_n}$ are the diagonal degree matrices for vertex and hyperedge correspondingly, in which:

$$\mathbf{D}^{t_n}_{\upsilon\upsilon} = \sum_{\epsilon=1}^{|\mathcal{E}^{t_n}|} W^{t_n}_{\epsilon\epsilon} H^{t_n}_{\upsilon\epsilon} \quad \mathbf{B}^{t_n}_{\epsilon\epsilon} = \sum_{i=1}^{|\mathcal{V}^{t_n}|} H^{t_n}_{i\epsilon}$$

At different time periods, there will be a different set of user-item interactions, leading to hypergraphs with changing topology. We aim to extract the item semantics from each of the short-term hypergraphs by capturing item correlations.

Hypergraph Convolution Network (HGCN). At each time period, we aim to exploit the correlations among items for their temporally dynamic embeddings, in which the correlated items should be close with each other for the short time period. To achieve that, an item should aggregate in-

formation (i.e., latent representations) from all its neighboring items (i.e., items with connection to it). This naturally fits the assumption of the convolution operation [63, 46, 64, 48] that more propagation should be done between connected items. Given that nodes in $\mathcal{V}^{t_n}$ have a set of initial latent representation $\mathbf{X}^{t_n,0} = [\mathbf{x}_1^{t_n,0}, \mathbf{x}_2^{t_n,0}, ..., \mathbf{x}_{|\mathcal{V}^{t_n}|}^{t_n,0}]$, the convolution operation can be defined as:

$$\mathbf{x}_i^{t_n,1} = \tau(\sum_{\upsilon=1}^{|\mathcal{V}^{t_n}|} \sum_{\epsilon=1}^{|\mathcal{E}^{t_n}|} H_{i\epsilon}^{t_n} H_{\upsilon\epsilon}^{t_n} W_{\epsilon\epsilon}^{t_n} \mathbf{x}_\upsilon^{t_n,0} \mathbf{P}^0)$$

in which $\tau(\cdot)$ represents the activation function (ReLu in our experiment). $\mathbf{P}^0$ represents the trainable weight matrix between the initial and the 1^{th} layer. This convolution operation will encode each hyperedge with all the nodes connected to it and then output the embedding for each node by aggregating information of all the hyperedges it is on. We can formulate this convolution process into a matrix form as:

$$\mathbf{X}^{t_n,1} = \tau(\mathbf{H}^{t_n} \mathbf{W}^{t_n} {\mathbf{H}^{t_n}}^T \mathbf{X}^{t_n,0} \mathbf{P}^0)$$

To prevent numerical instabilities caused by stacking multiple convolutional layers, we need to add in symmetric normalization. Then we end up with:

$$\begin{aligned} \mathbf{X}^{t_n,1} &= f(\mathbf{X}^{t_n,0}, \mathbf{H}^{t_n}, \mathbf{W}^{t_n} | \mathbf{P}^0) \\ &= \tau(\mathbf{D}^{t_n\,-1/2} \mathbf{H}^{t_n} \mathbf{W}^{t_n} \mathbf{B}^{t_n\,-1} {\mathbf{H}^{t_n}}^T \mathbf{D}^{t_n\,-1/2} \mathbf{X}^{t_n,0} \mathbf{P}^0) \end{aligned} \tag{3.4}$$

Here $f(\cdot)$ is used to denote the operation for one hypergraph convolutional layer to update each node with its one-hop neighbors. We can stack multiple convolution layers to recursively aggregate the information from high-order neighbors in the hypergraph. In such a hypergraph convolutional network (HGCN), The output from the L^{th} layer can be calculated as:

$$\mathbf{X}^{t_n,L} = f(\mathbf{X}^{t_n,(L-1)}, \mathbf{H}^{t_n}, \mathbf{W}^{t_n} | \mathbf{P}^{(L-1)})$$

The resulting $\mathbf{X}^{t_n,L}$ from layer L can inherit embeddings from previous layers to capture the propagation of item correlations in the hypergraph. While at different time periods, the topology of

hypergraphs is changing, leading to dynamic item embeddings reflecting the short-term correlations at different time periods.

Residual Gating. While items are changing, there is still linkage between their features at different timestamps. Some characteristics of an item will retain from the last time period to the next time period. For example, items may have some intrinsic features that change smoothly or are unchanged at all times. In order to propagate the residual information from the previous time periods to the future, we introduce a residual gating to generate the initial embedding of each node by combining the set of dynamic embeddings for t_1,..., t_{n-1} with the static embedding. The initial embedding of item i at t_n can be calculated as:

$$\mathbf{x}_i^{t_n,0} = g\mathbf{x}_i^{t<n,L} + (1-g)\mathbf{e}_i, \quad g = \frac{e^{\mathbf{z}_R^T \sigma(\mathbf{W}_R \mathbf{x}_i^{t<n,L})}}{e^{\mathbf{z}_R^T \sigma(\mathbf{W}_R \mathbf{x}_i^{t<n,L})} + e^{\mathbf{z}_R^T \sigma(\mathbf{W}_R \mathbf{e}_i)}}$$

in which $\mathbf{W}_R$ and $\mathbf{z}_R$ is the transformation matrix and vector for the gate. $\sigma(\cdot)$ is the *tanh* function. We use $\mathbf{x}_i^{t<n,L}$ to denote the dynamic embedding from the most recent hypergraph before t_n for item i. If item i doesn't appear in any previous hypergraph, we ignore the residual component and let $\mathbf{x}_i^{t_n,0} = \mathbf{e}_i$. The value g calculated with the gating function is used to control the percentage of residual information that will be retained. With this residual gating, we connect the hypergraph sequentially, leading to the major component of HyperRec – the sequential hypergraphs (as in Figure 4.4). At each time period, each item will be initialized from both the static item embedding and residual information from the past. And then the HGCN can incorporate the short-term item correlations to generate the expressive dynamic item embedding.

Short-term User Intent. As introduced in Figure 4.1, the short-term user intent can be inferred from all the items the user has interacted with in a certain time period. This naturally falls into the definition of the *hyperedge* which accounts for all the items a user has interacted with in the short-term altogether. Thus moving one step forward, we can aggregate the dynamic node embedding on each hyperedge to infer each user's short-term intent with the following operation

$$\mathbf{U}^{t_n} = \tau(\mathbf{B}^{t_n\,-1/2}\mathbf{H}^{t_n\,T}\mathbf{D}^{t_n\,-1/2}\mathbf{X}^{t_n,L}\mathbf{P}^L) \tag{3.5}$$

The resulting matrix $\mathbf{U}^{t_n} = [\mathbf{u}_1^{t_n}, \mathbf{u}_2^{t_n}, ..., \mathbf{u}_{|\mathcal{E}^{t_n}|}^{t_n}]$ can be regarded as an assembly of short-term user intents at t_n.

Fusion Layer. Then we want to incorporate both the dynamic item embedding and the short-term user intent for a more expressive representation of each interaction in the sequence. We propose the fusion layer as below to generate the representation of the interaction between user u and item i at t_n:

$$\begin{aligned} \mathbf{e}_{i,u}^{t_n} &= \alpha_u \mathbf{u}_u^{t_n} + \alpha_d \mathbf{x}_i^{t_n,L} + (1 - \alpha_d - \alpha_u)\mathbf{e}_i \\ \alpha_u &= \frac{e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{u}_u^{t_n})}}{e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{u}_u^{t_n})} + e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{x}_i^{t_n,L})} + e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{e}_i)}} \\ \alpha_d &= \frac{e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{x}_i^{t_n,L})}}{e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{u}_u^{t_n})} + e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{x}_i^{t_n,L})} + e^{\mathbf{z}^T \sigma(\mathbf{W}_F \mathbf{e}_i)}} \end{aligned} \tag{3.6}$$

in which $\mathbf{e}_i$ and $\mathbf{x}_i^{t_n,L}$ is the static and dynamic item embedding correspondingly, and $\mathbf{u}_u^{t_n}$ is the vector in the matrix generated by Equation 3.5 to indicate the short-term user intent at t_n. $\mathbf{W}_F$ and $\mathbf{z}$ is the transformation matrix and vector correspondingly. To avoid the overfitting problem, during training, for interactions happening at the same timestamp as what we want to predict, we feed in $\mathbf{u}_u^{t_{n-1}}$ and $\mathbf{x}_i^{t_{n-1},L}$ to the fusion layer while generating $\mathbf{e}_{i,u}^{t_n}$.

Self-attention. With the superior performance of self-attention layer (i.e., Transformer) in next-item recommendation compared with CNN, RNN and Markov Chains-based models (as shown in [23]), we adopt self-attention as the basic model to capture the dynamic pattern in interaction sequences. $\mathbf{e}_{i,u}^{t_n}$ can be treated as embedding for interaction between i and u at t_n.

Assume that we have a sequence of items user u has interacted with in chronological order $\mathbf{L}^u = ((i_1^u, t_1^u), (i_2^u, t_2^u), ..., (i_{|\mathbf{L}^u|}^u, t_{|\mathbf{L}^u|}^u))$. To represent the k^{th} interaction, we also take the position k into consideration. We use $\mathbf{o}_k^u = \mathbf{e}_{i_k^u,u}^{t_k^u} + \mathbf{p}_k$ to represent the interaction, in which $\mathbf{p}_k$ is the positional embedding of position k to characterize the order information.

Given embedding sequence $(\mathbf{o}_1^u, \mathbf{o}_2^u, ..., \mathbf{o}_{|\mathbf{L}^u|}^u)$, self-attention [22] is designed to generate the aggregation based on the similarities (attention scores) between the last element $\mathbf{o}_{|\mathbf{L}^u|}^u$ and each

element in the sequence. Then the attention score between $\mathbf{o}^u_{|\mathbf{L}^u|}$ and $\mathbf{o}^u_j$ can be calculated as:

$$att(\mathbf{o}^u_{|\mathbf{L}^u|}, \mathbf{o}^u_j) = \frac{(\mathbf{W}_Q \mathbf{o}^u_{|\mathbf{L}^u|})^T (\mathbf{W}_K \mathbf{o}^u_j)}{\sqrt{d}}$$

in which $\mathbf{W}_Q$ and $\mathbf{W}_K$ are transformation matrices and d is the dimension of the embedding. Then the attentive aggregation can be calculated as:

$$\mathbf{d}_u^{t^u_{|\mathbf{L}^u|}} = \sum_{j=1}^{|\mathbf{L}^u|} att(\mathbf{o}^u_{|\mathbf{L}^u|}, \mathbf{o}^u_j)(\mathbf{W}_V \mathbf{o}^u_j) \tag{3.7}$$

where $\mathbf{W}_V$ is a transformation matrix. Then the generated $\mathbf{d}_u^{t^u_{|\mathbf{L}^u|}}$ can represent the dynamic preference of user u after interacting with the sequence of items in $|\mathbf{L}^u|$ at $t^u_{|\mathbf{L}^u|}$.

Preference Prediction. While predicting the preference of users for items, we should take both the dynamic item embedding and the static item embedding into consideration:

$$\bar{y}_{u,i}^{t_{n+1}} = \mathbf{d}_u^{t_{<n+1}T} (\mathbf{x}_i^{t_{<n+1},L} + \mathbf{e}_i) \tag{3.8}$$

in which $\mathbf{d}_u^{t_{<n+1}}$ and $\mathbf{x}_i^{t_{<n+1},L}$ denotes the most recent dynamic user preference and dynamic item embedding generated before t_{n+1}. To train the model, we adopt Equation 3.3 defined in Section 3.3.2 as our loss function.

3.4.3 Experiment

In this section, we conduct experiments to evaluate the performance of the proposed HyperRec over datasets sampled from three online platforms (Goodreads, Amazon and Etsy). Besides its overall performance in next-item recommendation, we further investigate the design of HyperRec via ablation tests and parameter analysis. In addition, we also examine whether HyperRec can capture both the long-term and short-term patterns in the platforms based on its recommendation to users with various lifespans.

Data Similar to Section 3.3.3, we formulate the sequential recommendation problem under leave-

Dataset	# Users	# Items	# Interactions	Density	Cutting Timestamp
Amazon	74,823	64,602	1,475,092	0.0305%	Jan 1, 18
Etsy	15,357	56,969	489,189	0.0559%	Jan 1, 18
Goodreads	16,884	20,828	1730,711	0.4922%	Jan 1, 17

Table 3.4: Statistics of the datasets.

one-out setting and split the train-test data following the real-world scenario. To explore the generalization of the proposed model, we sample data from three different online platforms. Besides the datasets (e.g., Amazon and Etsy) used in Section 3.3.3, we also introduce the Goodreads dataset. It is collected from a book reading community – Goodread, in which users can tag, rate, and write reviews on books. We treat different types of interactions equally as implicit feedback on items. We keep users who interacted with more than 5 books before 2017 and at least 2 books in 2017. This dataset is denser than both Amazon and Etsy since the items (i.e., books) in such an information sharing platform are more stable and less likely to be replaced by new items as in ecommerce platforms (e.g., products can be replaced by upgraded models). Summary statistics of these datasets are in Table 3.4.

Baselines. Following are the baseline methods we adopt in our experiments.

- **TransRec**: *Translation-based recommendation* [30]. TransRec models the transitions between different items in the interaction sequences with user-specific translation operations.
- **GRU4Rec+**: *Recurrent Neural Networks with Top-k Gains* [19]. As an improved version of GRU4Rec [6], this model adopts a GRU to model sequential user behaviors with a new class of loss functions designed for improving Top-K gains.
- **TCN**: *A Simple Convolutional Generative Network for Next Item Recommendation* [21]. This baseline improves the typical CNN-based next-item recommendation models with masked filters and stacked 1D dilated convolutional layers for modeling long-range dependencies.
- **HPMN**: *Lifelong Sequential Modeling with Personalized Memorization* [29]. HPMN is pow-

ered by a hierarchical periodic memory network to capture multi-scale sequential patterns of users simultaneously, and thus can combine recent user behaviors with long-term patterns.

- **HGN**: *Hierarchical Gating Networks for Sequential Recommendation* [65]. This method contains a feature gating and an instance gating to hierarchically select the features and instance of items for user modeling while making next-item recommendation.

- **SASRec**: *Self-attentive Sequential Recommendation* [23]. It adopts the self-attention layer to capture the dynamic patterns in user interaction sequences. It can be treated as a simplified version of the dynamic user modeling component in HyperRec to use the static item embeddings to represent each interaction.

- **BERT4Rec**: *Sequential Recommendation with Bidirectional Encoder Representations from Transformer* [24]. This baseline utilizes a bi-directional self-attention module to capture the context information in user historical behavior sequences from both left and right sides.

Overall Model Comparison. We compare HyperRec with the baselines and the results are reported in Table 3.5. Under all the evaluation metrics, HyperRec can significantly outperform all the baselines in each of the datasets, which demonstrates its effectiveness in improving next-item recommendation in realistic settings where items evolve over time.

As a pioneer for personalized next-item recommendation, TransRec can provide promising improvement compared with simply recommending the most popular items. However, TransRec treats users as a linear translation between consecutive items they purchase, which limits the model in dealing with the realistic problems that both users and items are changing. With the development of neural networks in capturing dynamic patterns in sequential data, there are lots of recent efforts in adopting these neural structure for next-item recommendation. HPMN consists of hierarchical memory networks to create lifelong profiles for users, in which each layer of the memory network is designed to capture periodic user preferences with a specific period. We find that HPMN outperforms TransRec by more than 30% in Amazon but appear to be weak on Etsy and Goodreads.

Metrics	Datasets	**TransRec**	**HPMN**	**TCN**	**GRU4Rec+**	**BERT4Rec**	**HGN**	**SASRec**	**HyperRec**	Improv.
NDCG@1/ HIT@1	Amazon	0.0533	0.0771	0.0783	0.0983	0.1011	0.1012	0.1051	0.1215∗	20.03%
	Etsy	0.4201	0.3746	0.3816	0.3916	0.4338	0.4379	0.4477	0.4725∗	7.90%
	Goodreads	0.2174	0.2229	0.2069	0.2360	0.2366	0.2447	0.2643	0.2878*	17.62%
NDCG@5	Amazon	0.1202	0.1663	0.1648	0.1989	0.2010	0.1981	0.2041	0.2264∗	12.60%
	Etsy	0.5495	0.5096	0.5120	0.5307	0.5553	0.5698	0.5713	0.5946∗	4.37%
	Goodreads	0.3752	0.3847	0.3593	0.4035	0.4073	0.4163	0.4326	0.4624∗	11.07%
HIT@5	Amazon	0.1867	0.2543	0.2499	0.2963	0.2972	0.2918	0.3001	0.3272∗	10.08%
	Etsy	0.6678	0.6300	0.6310	0.6566	0.6650	0.6885	0.6816	0.7047∗	2.35%
	Goodreads	0.5234	0.5358	0.5009	0.5581	0.5643	0.5747	0.5865	0.6206∗	7.98%
MRR	Amazon	0.1357	0.1780	0.1777	0.2073	0.2094	0.2070	0.2120	0.2328∗	11.19%
	Etsy	0.5328	0.4920	0.4974	0.5131	0.5411	0.5519	0.5555	0.5780∗	4.73%
	Goodreads	0.3624	0.3707	0.3495	0.3867	0.3896	0.3979	0.4146	0.4418∗	11.02%

Table 3.5: Comparison of Different Models. $*$ indicates that the improvement of the best result is statistically significant compared with the next-best result with $p < 0.01$.

Building on top of 1D dilated convolution layers, TCN shows its strength in modeling the short-term behaviors and outperforms HPMN in ecommerce for Etsy. It does not seem to be a good fit for scenario like Goodreads in which the long-term preferences are significant. As an advanced version of GRU4Rec targeting Top-K recommendation, in Amazon and Goodreads, GRU4Rec+ can improve TCN and HPMN by conducting dynamic user modeling with GRU and adopting a loss function tailored to RNN-based models for Top-K recommendation. The newly proposed HGN is equipped with a novel feature-gating and an instance gating to enhance the short-term user modeling, and thus can outperform the aforementioned baselines. Both SASRec and Bert4Rec employ a self-attention layer to model the sequential user patterns. In BERT4Rec, by randomly masking items in the user sequences, it is able to train a bidirectional model for recommendation. However, it does not bring in huge improvement as in the original BERT applications for natural language processing since the right-to-left patterns in sequences are not necessarily informative for predicting dynamic user preferences.

Compared with the state-of-the-art, HyperRec can achieve its largest improvement in Amazon than in other datasets. The reason might be that HyperRec is able to fully extract the item characteristics from extremely sparse user-item interactions with the Hypergraph topology. Meanwhile, the outstanding performance of HyperRec in both ecommerce and information sharing platforms demonstrates that it can be generalized to various online scenarios.

Evaluation of HyperRec. We report the results of our ablation tests in Table 3.6. For fair comparison, results are all achieved with granularity of time periods to be 12-months and HGCN containing 2 layers when there are hypergraphs in the models. First of all, HyperRec can achieve the best performance compared to any of its variants for all the datasets, indicating the effectiveness of its design.

To evaluate the effectiveness of the hypergraph structure, in (3), we assign each user and each item with different latent embeddings at different time periods as the dynamic item embeddings and short-term user intents. That is, instead of exploiting the short-term item correlations with a hypergraph as in HyperRec, we use these time-dependent embeddings to encode the change in

the platforms. We find that there is a huge drop in performance. In Amazon and Goodreads, the performance of (3) is even worse than that of (2) which uses static item embeddings. One reason is that the user-item interactions at each time period are too sparse to sufficiently train the time-dependent embedding directly. But hypergraph with HGCN is able to fully extract the effective correlations between items from the multi-hop connections at each time period.

Architecture	Amazon	Etsy	Goodreads
(1) *HyperRec*	0.1215	0.4712	0.2809
(2) *Static Item Embedding*	0.1051	0.4477	0.2643
(3) *Replace Hypergraph*	0.0978	0.4588	0.2576
(4) *(-) Residual*	0.1169	0.4591	0.2626
(5) *(-) Dynamic Item Embedding*	0.1131	0.4646	0.2789
(6) *(-) Short-term User Intent*	0.1147	0.4616	0.2709
(7) *(-) Dynamic in Prediction*	0.1151	0.4703	0.2746

Table 3.6: Results for Ablation Test under HIT@1/NDCG@1. (-) denotes removing the specific component.

Case Study – Change of Time Granularity. An important parameter which can control how sensitive HyperRec is to the change over time is the granularity of the time period. Thus in Figure 3.12, we show the performance of the proposed model by varying the granularity from 1 month to 18 months. When the granularity is small, we find that the model cannot achieve the best performance since the interactions are extremely sparse and not sufficient for building up a set of expressive item embeddings. While enlarging the granularity, we find that the performance of HyperRec is increasing in all the datasets. In Amazon, it reaches the best performance when the granularity is set to be 12-months. However, for Etsy, the optimized granularity is smaller since the products sold on Etsy (i.e., hand-crafted items) are in higher volatility than products on

Amazon. In Goodreads, the optimized granularity is around 6-months, which is smaller than that for the other datasets since there are more interactions for each time period in Goodreads for the dynamic item embedding. If we further enlarge the granularity, the performance will decrease since it underestimates the change of items and may introduce noise to the model.

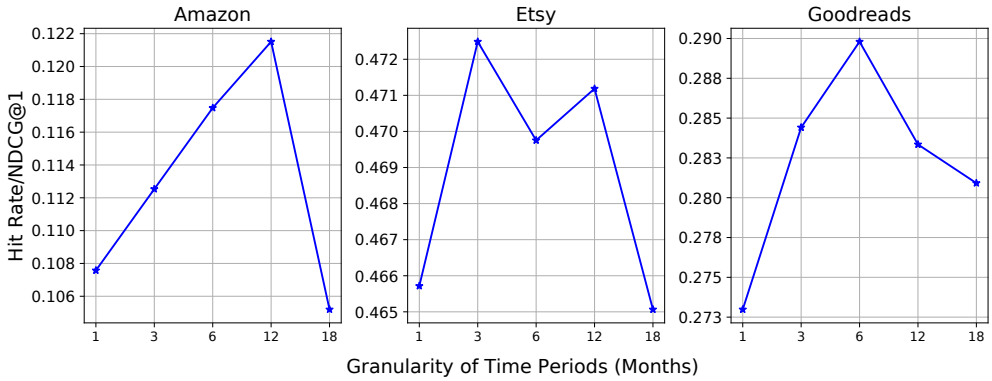

Figure 3.12: Performance comparison with various time granularity under HIT@1/NDCG@1.

Case Study – Different User Groups. To further explore the performance of the proposed model in both long-term and short-term scenarios, we compare HyperRec with the top-2 baselines, HGN and SASRec, for users with various lifespans in the platforms (in Figure 3.13). Here, we calculate the time gap between the last interaction and the first interaction for each user as his/her lifespan in the platform. We find that HGN works better than SASRec for users with a short lifespan (less than one year), while SASRec can outperform HGN in modeling the users who are active for longer time in the platforms. However, we find that HyperRec significantly outperforms the baselines for users with both short and long lifespans. And it can achieve comparatively larger improvement for users with longer lifespans, indicating that HyperRec is superior in capturing the long-term patterns while taking the short-term correlations into consideration.

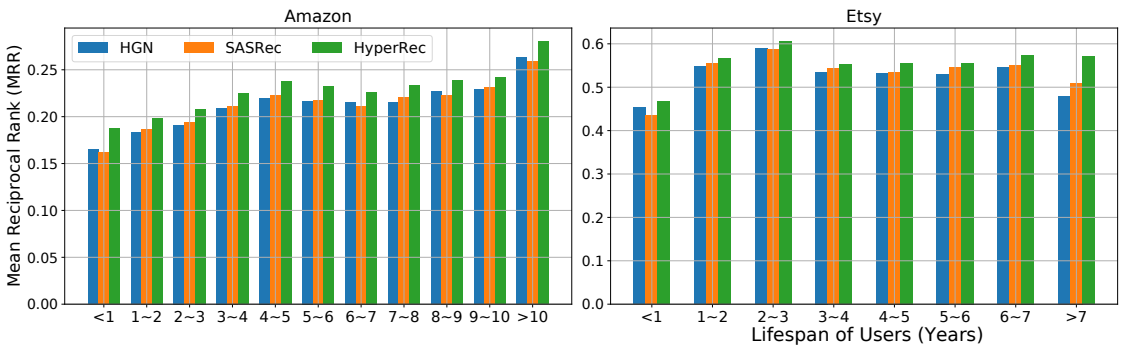

Figure 3.13: Performance comparison for users with different lifespans.

3.5 Conclusion and Future Work

In this chapter, to enable a recommendation systems to be sustainable and highly adaptive to the temporal dynamics present in these platforms, we make several contributions on balancing the instant shift in user engagement with the lifelong evolution of users and items.

On one hand, shopping decisions can be influenced by different occasions, leading to purchases that instantly deviate from a user's intrinsic preferences. Over Amazon and Etsy, we gain insights into the traceable patterns of personal and global occasion signals. We propose to utilize different attention mechanisms to elicit different occasion signals for recommendation. Through experiments, we find the proposed ***O***ccasion-***A***ware ***R***ecommender model can outperform the state-of-the-art model in two real-world e-commerce datasets.

On the other hand, we explore the dynamic meaning of items in real-world scenarios and propose a novel next-item recommendation framework empowered by sequential hypergraphs to incorporate the short-term item correlations for dynamic item embedding. With the stacking of hypergraph convolution networks, a residual gating and the fusion layer, the proposed model is able to provide more accurate modeling of user preferences, leading to improved performance compared to the state-of-the-art in predicting user's next action for both ecommerce (Amazon and Etsy) and information sharing platform (Goodreads).

In the future, we are interested in introducing more context information to characterize the

occasions explicitly and provide explainable recommendations and investigating how to transfer the dynamic patterns across platforms or across domains for an improved predictive performance.

4. RESILIENCE: IDENTIFYING THE RESILIENT RECOMMENDER "BACKBONE"[1]

4.1 Introduction

A sustainable recommendation system should be resilient to the dramatic change in a community, e.g., users joining and leaving the community, churn in new items and frequent updates of user relationships. However, even in changing environments, users may be influenced by a small group of "backbone" individuals – Key Opinion Leaders (KOL), who can shape their views, and further impact what items they buy, what media they consume, and how they interact with online platforms [10, 11]. For example, KOLs on Instagram and Pinterest could influence shopping decisions by highlighting new fashion trends [66, 67, 68], while KOLs on Yelp and TripAdvisor could guide customer restaurant selection by providing explanatory information (like photos and reviews) for restaurants [69, 70]. Previous research has shown the effectiveness of modeling KOLs in different learning tasks, such as public sentiment analysis [71] and social event detection [72]. However, the effect of these "backbone" users in recommendation systems remains largely unexplored, which motivates us to develop a novel recommendation system by explicitly capturing the influence from the "backbone" – KOLs to the whole platform.

Despite the importance of investigating the influence of KOLs in recommendation systems, however, it is a non-trivial task due to two major challenges: (i) **Elicitation**: Compared to regular users, KOLs tend to express their opinions on items explicitly rather than leave implicit feedback. More important, such explicit interactions are inherently multi-relational: on the one hand, KOLs are able to express their opinions via different ways (e.g., review, rating or tagging); On the other hand, the opinions from KOLs could have distinct meanings (e.g., tag "fantastic" and tag "terrible" are semantically different). However, it is unclear that how to extract the elite opinions of KOLs from such multi-relational data. (ii) **Diffusion**: In online communities, KOLs are able to guide their followers' preferences and shape how users view the items. For example, users tend to

[1]Reprinted with permission from "Key Opinon Leaders in Recommendation Systems: Opinion Elicitation and Diffusion" by Jianling Wang, Kaize Ding, Ziwei Zhu, Yin Zhang and James Caverlee, 2020. Proceedings of the 13th International Conference on Web Search and Data Mining. Copyright 2020 by ACM.

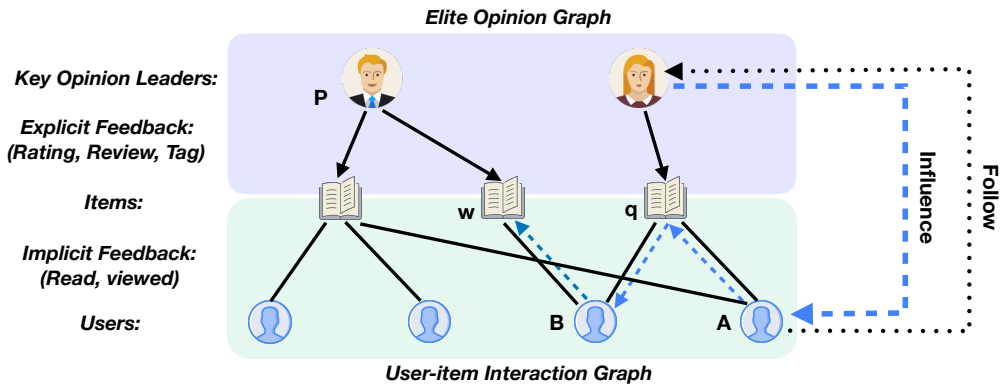

Figure 4.1: The opinions from Key Opinion Leaders (KOLs) can diffuse to their followers and items they comment. Furthermore, these opinions *diffuse* in the community via both direct and multi-hop connections between users and items.

purchase makeup products with the recommendation of Beauty-KOLs they are following; a book which was tagged as "For Teens" by KOLs could attract many teenager readers. Meanwhile, previous research [73, 7] has shown that user preferences on items could diffuse through high-order connectivity (e.g., in Figure 1, the latent preference of user A can possibly diffuse via the transitive path $A \rightarrow q \rightarrow B \rightarrow w$, to items that he/she hasn't interacted with). Therefore, the influence from KOLs will also be propagated to those non-direct followers in the community. In this regard, another challenge centers around how to model this elite opinion diffusion process for improved recommendation?

To tackle the aforementioned challenges, in this work, we propose *GoRec*: a novel *end-to-end* **G**raph-based neural model to incorporate the influence of K**O**Ls for **Rec**ommendation. Specifically, we introduce a translation-based embedding method to elicit the opinions of KOLs, in which the elite opinions are regarded as different types of translations from KOLs to items. In this way, we are able to extract the embeddings for both KOLs and items, and use them to enrich the initial user/item embeddings in the user-item interaction graph. In order to model the diffusion process of elite opinions, our model employs multiple Graph Neural Network (GNN) layers to learn the final user/item embeddings following the neighborhood aggregation strategy [48, 52]. As such, the final embeddings enriched with the elite opinions from KOLs can be decoded to accurately infer users'

preferences on all the items.

Contributions. In this chapter, with the data-driven analysis, we explore the relationships among users, items and key opinion leaders within real-world online platforms, and uncover the importance of explicitly modeling the influence of the "Backbone" KOLs in recommendation systems. We develop a novel end-to-end item recommendation framework - *GoRec*, which is able to elicit elite opinions from KOLs and model their diffusion in the community. Meanwhile, we onduct extensive experiments on two real-world scenarios including Goodreads (a book sharing community) and Epinions (an ecommerce review sharing platform). We find that the proposed *GoRec* model outperforms the state-of-the-art by 10.75% and 9.28% on average in Top-K recommendation. Meanwhile, we find that a small set of KOLs is sufficient to hint on the preferences of a huge amount of users in the community and thus benefit the recommendation system.

4.2 Motivation

To gain insight into the relationships among the recommendation "Backbone", users and items, we start our discussion with an initial exploration into the Goodreads community. Goodreads is a book-based platform with about 80 million registered users in which users can manage their own book reading habits and also connect with other readers [74]. Users on Goodreads leave lots of *implicit* feedback, which can be treated as positive signals while inferring users preferences [3, 2] (e.g. a user read both "Harry Potter" and "Twilight" probably likes paranormal fiction). However, a user who has not read a book (and hence, left no implicit feedback) may not because she doesn't like it but just not knows about the book. Goodreads are also shaped by *explicit* feedback, which conveys positive or negative opinions on items explicitly, e.g., text reviews, numerical ratings, or semantic tagging. Aside from the bidirectional "friendship", users on Goodreads can unidirectionally follow other users from whom they can receive activity updates. One of the key features of Goodreads is that some users are experts who provide detailed reviews and highlight new releases, and thus are followed by many other users. In our initial analysis, naturally, we ranked all the accounts based on their numbers of followers and treat the top accounts as KOLs [75, 76].

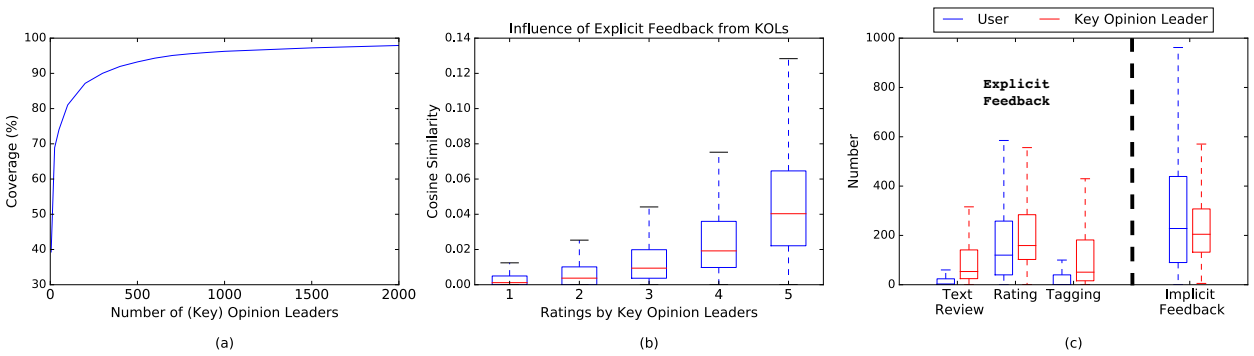

Figure 4.2: (a) Coverage: The percentage of users following at least one of the top (key) opinion leaders. More than 95% of users follow at least one of the Top-500 accounts. (b) Books read by users are more similar to books with higher ratings from key opinion leaders they are following. (c) While leaving a similar number of implicit feedback, key opinion leaders prefer to show their opinions on items via explicit interactions (reviews, ratings, self-defined tags).

A small number of key opinion leaders (KOLs) can provide sufficient coverage. Before exploiting the influence of KOLs, we want to examine whether the top accounts that we treat as KOLs can provide sufficient coverage for the regular users in Goodreads. In Figure 4.2(a), we check the percentage of users following at least one of the KOLs (what we refer to as coverage) by changing the number of top accounts that we consider as KOLs. We find that while considering only the top-500 KOLs, there are more than 95% of the users following at least one of these KOLs. In other words, the patterns and behaviors of just the top accounts (the KOLs) can potentially have wide-reaching impact on the community. We can conclude that *a small number of key opinion leaders (KOLs) can provide sufficient coverage.* Next, focusing on the top-500 KOLs, we explore whether there are patterns in their opinions and how their opinions can diffuse to the community.

Users are shifted by the KOLs they are following. To examine whether the explicit opinions from KOLs can influence what their followers read, we represent each user with a simple binary vector over all books, in which a "1" indicates that the user has left implicit feedback on this book. We use five different binary vectors for each KOL to represent books with different ratings (1 to 5) from the KOL. In Figure 4.2(b), we list the similarities between books read by users and books with different ratings from KOLs they are following. We find that the set of books a user read are more

similar to the books receiving high ratings from KOLs they follow, while having little overlapping (similarity) with books with low ratings (1 or 2) from the KOLs they follow. We conclude that the explicit opinions of KOLs could directly influence what their followers consume.

Compared to ordinary users, KOLs tend to express opinions on items explicitly. In Figure 4.2(c), we compare the numbers of different kinds of feedback from regular users and KOLs in Goodreads. We find that ordinary users and KOLs leave implicit feedback on a similar number of books (that is, they mark a book as "read" or "to read"), indicating that both are active in their use of Goodreads, presumably for managing their own book collections. However, we do find that KOLs tend to leave more reviews, ratings and tags; that is, KOLs are more engaged in explicitly sharing their opinions on books. KOLs appear to be capable of providing specialized expertise and high-quality opinions in the community. These *elite opinions* (including reviews, ratings and tags) are public on the item pages and can influence how the community views or defines the items, illustrating a possible way of how these elite opinions diffuse in the community. In this work, we will focus on explicitly modeling the influence of KOLs in recommendation system with opinion elicitation and diffusion.

4.2.1 Related Work

There is a growing interest in incorporating various auxiliary information into recommendation to deal with the challenge of sparse interactions between users and items. One direction is to extract content information of items to enrich the latent representations, like visual features from images [77, 78], topical information from text descriptions [79, 80] or audio features from multimedia [81].

On the other hand, there are also works on transferring semantic knowledge learned from the relational graph of many knowledge bases (e.g., DBpedia, YAGO, Freebase) for item recommendation [82, 83, 80]. KTUP [82] and MKR [83] combine the tasks of item recommendation and Knowledge Graph (KG) completion, and thus transfer information of entities to items. However, there are challenges in utilizing the KG in recommendation systems. Update and maintenance of the semantic databases highly depends on the contributors. It takes effort to find the correct map-

ping between items and entities in the KG. An incorrect mapping will introduce noise and hurt the recommendation. Thus, in our work, instead of relying on externally managed KGs, we focus on eliciting elite opinions from KOLs in the community itself, to characterize items and users.

4.3 Methodology

With these observations in mind, we propose *GoRec*, a novel graph-based recommendation system enhanced with the influence of KOLs in the community. Our design is structured around the challenges we are faced with: (i) **Elicitation:** How can we elicit the elite opinions of KOLs from the multi-relational data? (ii) **Diffusion:** How to model the diffusion process of elite opinions in the community for improved recommendation?

4.3.1 Problem Setting and Notation

Task. In this work, we aim to provide Top-K recommendation from a candidate set of M items $\mathbf{I} = \{i_1, i_2, ..., i_M\}$ to a set of N users $\mathbf{U} = \{u_1, u_2, ..., u_N\}$. For each user u, we use a binary vector $\mathbf{y}_u = \{y_{u1}, y_{u2}, ..., y_{uM}\}$ to indicate the implicit feedback u left on all the items. That is, if u interacted with item i, then $y_{ui} = 1$. And $y_{ui} = 0$ means u has not left any feedback on i.

User-item Interaction Graph. Based on the (implicit) interactions between users and items, we can construct a bipartite graph $\mathbf{G} = (\mathcal{V}, \mathcal{W})$ in which the set of nodes $\mathcal{V} = \mathbf{U} \cup \mathbf{I}$ consists of all the users and items. The edge $(u, i) \in \mathcal{W}$ denotes that user u has implicit feedback on item i. Similarly, we can construct an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times M}$ for graph $\mathbf{G}$ by concatenating the feedback vector of each user, that is $\mathbf{A} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N]^T$.

Elite Opinion Graph. We use $\boldsymbol{L} = \{l_1, l_2, ..., l_P\}$ to represent the set of key opinion leaders (KOLs) we investigate while constructing the elite opinion graph $\mathbf{G}_o$. The explicit opinions can be different rating levels, words mentioned in the reviews or tags. We consider Q different types of explicit opinions $\boldsymbol{O} = \{o_1, o_2, ..., o_Q\}$ for the graph. Thus based on the explicit feedback from KOLs to items, we can harvest many opinion triplets. Each triplet is denoted as (l, o, i) representing kol l left opinion o on item i. And we construct a directed graph $\mathbf{G}_o$ composed of these *kol-opinion-item* triplets.

User-KOL Following. To explore how the elite opinions from KOLs directly influence their followers, we use $\mathbf{F}_u \subset \boldsymbol{L}$ to represent the set of KOLs followed by user $u \in \boldsymbol{U}$. And we let $\boldsymbol{U} \bigcap \boldsymbol{L} = \emptyset$.

4.3.2 "Backbone" Opinion Elicitation

First, we start by eliciting the opinions from KOLs toward improving the quality of recommendation. Recall that KOLs leave explicit opinions on items via reviews, ratings and tags. These opinions constitute a large scale of multi-relations from KOLs to items. As analogous to the data structure of knowledge graph, the resulted elite opinion graph $\mathbf{G}_o$ consists of many valid opinion triplets. For example, a triplet (l_1, *Review: wizard*, *Harry Potter*) denotes that KOL l_1 mentions the word $wizard$ in a review for item *Harry Potter*. As shown in Figure 4.3, we can also construct these opinion triplets based on ratings or tags provided by KOLs, and get triplets like (l_1, *Rate: 5*, *Harry Potter*) or (l_1, *Tag: fiction*, *Harry Potter*).

Our goal is to generate effective embedding for both items and KOLs in a continuous vector space while preserving the multi-relations (opinions) between them. In the below, we will list three features of $\mathbf{G}_o$ followed by the corresponding design we propose in the opinion elicitation process:

Feature 1. Multiple relations: Opinions come with distinct meaning, e.g., tag "fantastic" and "terrible" are semantically different.

Translation from KOL to Item. Adopting the similar idea in multi-relational graph embedding [84, 85, 86, 87], we treat opinions as translations from KOLs to items. That is, given a valid opinion triplet (k_h, o_r, i_t), we want to ensure that the embedding of item i_t is close to the embedding of KOL k_h plus the embedding of opinion o_r. Let $s(k_h, o_r, i_t)$ denote the scoring function for the translation operation, with which larger value means better translation. Given all the valid (positive) and negative opinion triplets, the objective is to maximize the translation score for all the positive triplets while minimizing that for the negative triplets. We formalize this objective into a

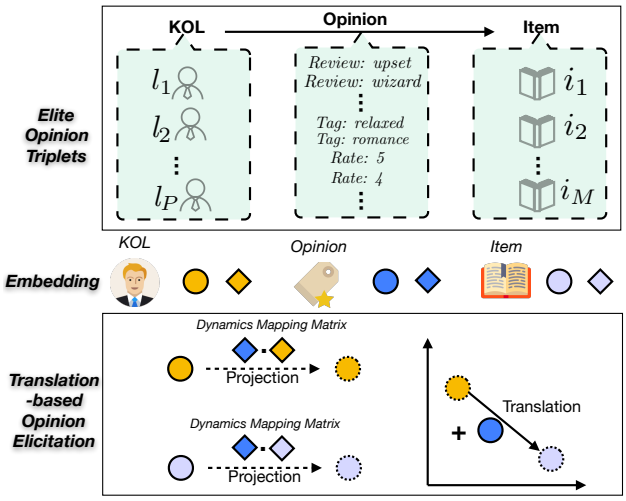

Figure 4.3: Translation-based Embedding with Elite Opinions.

task of minimizing the marginal loss below:

$$L_{op} = \sum_{(k_h, o_r, i_t) \in \mathbf{G}_o} \sum_{(k_h, o_r, i_t') \in \mathbf{G}_o^-} [\gamma + s(k_h, o_r, i_t') - s(k_h, o_r, i_t)]_+ \tag{4.1}$$

in which $[\cdot]_+ \triangleq max(0, \cdot)$ and γ denotes the margin the model used to separate the valid (positive) triplets and negative triplets. Here the negative triplet $(k_h, o_r, i_t') \in \mathbf{G}_o^-$ indicates that k_h wouldn't attach opinion o_r on i_t'. Thus while generating the negative samples, we randomly select i_t' from the subset of items that k_h has left feedback on excluding o_r.

Feature 2. Many-to-Many relations: On one hand, the connections between KOLs and items are not always one-to-one. On the other hand, KOLs can endow opinions with their personal attitudes. For example, each KOL has his/her own criteria for tagging a book with "BestOf2019".

Dynamic Mapping Matrix. To handle the *Many-to-Many* relations, a common strategy is to project KOLs and items to an opinion-specific space before the translation operation. Additionally, to cope with the various meanings of the same opinion, while doing projection, we adopt a dynamic mapping matrix [87] which is determined by both the opinion and the KOL (or item). In our case, each kol, item and opinion is represented by two vectors. One vector acts as its latent representations,

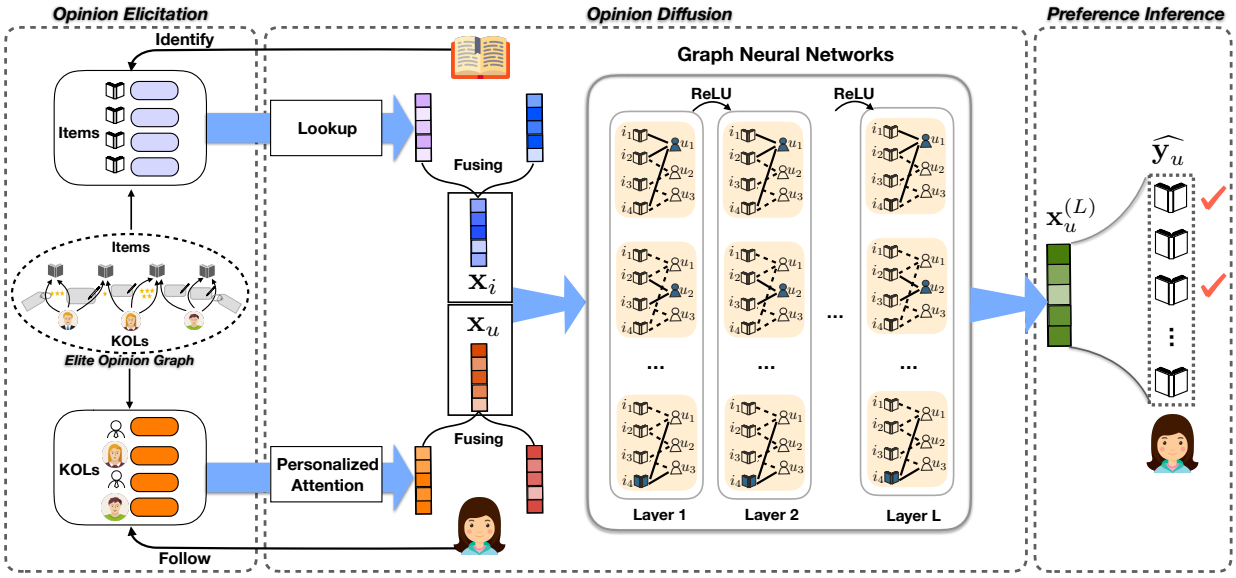

Figure 4.4: Graph Neural Recommendation Incorporating the Influence of the "Backbone".

while the other vector is used to construct the mapping matrix (as in Figure 4.3). Given a triple (k_h, o_r, i_t), we will initialize dense vectors $\mathbf{k}_h^e$, $\mathbf{k}_h^t$, $\mathbf{o}_r^e$, $\mathbf{o}_r^t$, $\mathbf{i}_t^e$, $\mathbf{i}_t^t$. First we will construct the mapping matrices for k_h and i_t on opinion o_r with vectors $\mathbf{k}_h^t$, $\mathbf{i}_t^t$ and $\mathbf{o}_r^t$:

$$\mathbf{M}_{rh} = \mathbf{o}_r^t \mathbf{k}_h^{t\,T} + \mathbf{I} \quad \mathbf{M}_{rt} = \mathbf{o}_r^t \mathbf{i}_t^{tT} + \mathbf{I}$$

in which $\mathbf{I}$ denotes the identity matrix. $\mathbf{M}_{rh}$ is used to transfer $\mathbf{k}_h^e$ to the space of o_r and $\mathbf{M}_{rt}$ is for transferring $\mathbf{i}_t^e$. Thus we get the projected representation of k_h and i_t under opinion o_r with:

$$\mathbf{k}_h^\perp = \mathbf{M}_{rh}\mathbf{k}_h^e \quad \mathbf{i}_t^\perp = \mathbf{M}_{rt}\mathbf{i}_t^e$$

The score function $s(\cdot)$ used to evaluate the translation distance for triple (k_h, o_r, i_t) is represented as:

$$s(k_h, o_r, i_t) = -\|\mathbf{k}_h^\perp + \mathbf{o}_r^e - \mathbf{i}_t^\perp\|^2 \tag{4.2}$$

in which we use L2-norm to calculate the distance empirically. Larger $s(k_h, o_r, i_t)$ means k_h and i_t are close to each other with translation o_r, i.e, it is more likely that k_h attaches opinion o_r to i_t.

Feature 3. Preference Signals: KOLs have preferences on the items they would interact with, e.g., a romantic book lover may seldom leave any feedback on horror novels.

Personalized Ranking Model A typical assumption is that the items with feedback from the user are preferred than those without. We also want to capture these (implicit) preference signals while modeling both KOLs and items. Following the basic idea in matrix factorization, we use the multiplication between $\mathbf{k}_h^e$ and $\mathbf{i}_t^e$, that is $p(k_h, i_t) = \mathbf{k}_h^{eT}\mathbf{i}_t^e$, to capture the preference of k_h on i_t. Then given the positive pair (k_h, i_t) representing k_h has left feedback on i_t and negative pair (k_h,$i_t{}'$) meaning k_h has not left feedback on $i_t{}'$, we adopt Bayesian Personalized Ranking (BPR) [4] to maximize the difference of preference scores between the positive pair and the negative pair. With $\delta(\cdot)$ denoting the Sigmoid function, the objective function to model these preference signals is:

$$L_{BPR} = \sum_{(k_h, i_t, i_t{}') \in \mathbf{S}} - \ln \delta(p(k_h, i_t) - p(k_h, i_t{}')) \tag{4.3}$$

Each element in the training data set $\mathbf{S}$ is generated by combining the ground truth interaction pair (k_h, i_t) with item $i_t{}'$ that KOL k_h hasn't left any feedback on.

Joint Tasks. During the opinion elicitation, we combine the task of modeling explicit opinions and extracting preference signals from the elite opinions graph $\mathbf{G}_o$, leading to the following loss function:

$$L_{\mathbf{G}_o} = L_{op} + \beta L_{BPR} \tag{4.4}$$

Here β is used to adjust the weight of pairwise loss in capturing the preference signals. By minimizing this joint loss $L_{\mathbf{G}_o}$, we will get the set of embeddings $\mathbf{K}_e = \{\mathbf{k}_1^e, \mathbf{k}_2^e, ..., \mathbf{k}_P^e\}$ for KOLs and $\mathbf{I}^e = \{\mathbf{i}_1^e, \mathbf{k}_2^e, ..., \mathbf{i}_M^e\}$ for items, which inherit both the explicit information and preference signals in the elite opinion graph $\mathbf{G}_o$.

4.3.3 "Backbone" Opinion Diffusion

As explained in Section 4.2, the opinions from KOLs can influence their followers and items they comment, and thus make up part of their features. Besides the implicit user-item interactions,

these elite opinions should also be exploited while modeling users preferences. In what follows, we will start from enriching the initial user/item embeddings with elite opinions. Then we will explain how to model the elite opinion diffusion process with graph neural networks.

Fusing Layer (Users). Each user is associated with an embedding $\mathbf{e}_u^U \in \mathbf{R}^d$ to represent the initial interest, which can be derived from his/her one-hot index with a fully-connected dense layer. Since users are directly influenced by whom they follow, aggregating the embeddings of KOLs whom the user is following can hint on the user preferences on items. However, we know that a particular KOL can have different levels of influence on different users. Building on the recent development of attention mechanisms [88, 22], we can model the dynamic (personalized) linkage between users and KOLs. We have the set of embeddings from Section 4.3.2 for the set of P KOLs $\mathbf{k} = \{\mathbf{k}_1^e, \mathbf{k}_2^e, ..., \mathbf{k}_P^e\}$. Given that $\mathbf{F}_u$ is the set of KOLs that u is following and $\mathbf{e}_u^U$ is a trainable dense representation for u, the weight of KOL p's influence on user u can be calculated as:

$$\alpha_{up} = \frac{e^{d_{up}}}{\sum_{j \in \mathbf{F}_u} e^{d_{uj}}}, \qquad d_{up} = \mathbf{z}^T ReLu(\mathbf{W}_A[\mathbf{k}_p^e \| \mathbf{e}_u^U] + \mathbf{b}_A)$$

Here $||$ represents the concatenation operation. $\mathbf{W}_A$ and $\mathbf{b}_A$ is the weight matrix and bias for the attention layer. $\mathbf{z}$ is a transformation vector. Then we aggregate the embeddings of all KOLs the user follows with the attentive weights:

$$\mathbf{n}_u = \sum_{j \in \mathbf{F}_u} \alpha_{uj} \mathbf{k}_j^e$$

Thus $\mathbf{n}_u$ can be used to characterize influence of elite opinions to user u from whom he/she follows. Lastly, we fuse $\mathbf{n}_u$ with the initial embedding of u with the following operation:

$$\mathbf{x}_u = ReLu(\mathbf{W}_U[\mathbf{n}_u \| \mathbf{e}_u^U])$$

where $\mathbf{W}_U$ is a transformation matrix, and the output $\mathbf{x}_u$ will be treated as cornerstone for the opinion diffusion.

Fusing Layer (Items). Similarly, each item will start with a trainable dense representation $\mathbf{e}_i^I \in \mathbf{R}^d$, which is associated with its index. Since the KOLs can influence how the whole community view an item, we want to complement $\mathbf{e}_i^I$ with the KOL-defined features $\mathbf{i}_i^e$ of item i which we elicit from the opinions of KOLs. Thus we adopt the similar fusion operation to generate the enriched representation of item i:

$$\mathbf{x}_i = ReLu(\mathbf{W}_I[\mathbf{i}_i^e \| \mathbf{e}_i^I])$$

in which $\mathbf{W}_I$ is a transformation matrix and $\mathbf{i}_i^e$ is the embedding gained from Section 4.3.2 for item i.

Opinion Diffusion with GNNs. As suggested by [73, **?**], user preferences on items could diffuse through high-order connectivity, thus the elite opinions from KOLs will also be propagated to those non-direct followers in the community. In this dissertation, we propose to model this opinion diffusion process by virtue of Graph neural networks (GNNs) [56, 48, 89, 90].

The core idea of GNNs is that each layer learns the node embeddings by aggregating the features of neighbors. At the initial GNN layer of our model, for user u and item i, given the sets of neighbors $\mathcal{N}_u$ and $\mathcal{N}_i$ which are directly connected with u and i correspondingly, we formulate the message passing on the edge (u, i) from i to u as:

$$\mathbf{c}_{i\to u}^{(1)} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\mathbf{W}_U^{(0)}\mathbf{x}_i \tag{4.5}$$

Here, $\mathbf{x}_i$ is the representation of i with influnce from KOLs and $\mathbf{W}_U^{(0)}$ denotes a trainable transforming matrix for users at layer 0. The term $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ is a normalization constant between u and i. And then we need to sum up all the message passed to u to generate its representation $\mathbf{x}_u^{(1)}$:

$$\mathbf{x}_u^{(1)} = \tau(\sum_{i\in\mathcal{N}_u}\mathbf{c}_{i\to u}^{(1)})$$

where $\tau(\cdot)$ is the activation function and we choose $ReLu$ in this work empirically. Similarly, we

can generate the representation of item i at this layer with:

$$\mathbf{x}_i^{(1)} = \tau(\sum_{u \in \mathcal{N}_i} \mathbf{c}_{u \to i}^{(1)}) = \tau(\sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{W}_I^{(0)} \mathbf{x}_u)$$

After generating the $\mathbf{x}_u^{(1)}$ and $\mathbf{x}_i^{(1)}$ from the first GNN layer, we can further capture the high-order diffusion by stacking multiple GNN layers. Specifically, at the L^{th} layer, we will have:

$$\mathbf{x}_u^{(L)} = \tau(\sum_{i \in \mathcal{N}_u} \mathbf{c}_{i \to u}^{(L)}) = \tau(\sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{W}_U^{(L-1)} \mathbf{x}_i^{(L-1)})$$

$$\mathbf{x}_i^{(L)} = \tau(\sum_{u \in \mathcal{N}_i} \mathbf{c}_{u \to i}^{(L)}) = \tau(\sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{W}_I^{(L-1)} \mathbf{x}_u^{(L-1)})$$

Note that $\mathbf{x}_u^{(L)}$ at layer L inherits embeddings of users and items from previous layers. That is how we *capture the diffusion of opinions in multiple-order user-item connectivity with GNNs*.

Preference Inference With what we have reached so far, our final step is to infer user u's preference on all the items. That is the probability that these items are connected with u. Following the similar idea as in Graph Auto-encoder [91, 51] and Autorec [5] - a basic autoencoder for recommendation, we use a fully-connected layer to recover the graph structure (user-item interaction graph) from the output of the encoder (the stack of GNN layers). That is, for u, we will decode $\mathbf{x}_u^{(L)}$ to reconstruct his/her feedback vector $\mathbf{y}_u$:

$$\widehat{\mathbf{y}_u} = \delta(\mathbf{V}\mathbf{x}_u^{(L)} + \mathbf{b}')$$

where $\mathbf{V}$ and $\mathbf{b}'$ are the weight matrix and bias term correspondingly. And $\delta(\cdot)$ represents the Sigmoid function. The objective is to minimize the reconstruction loss L_{re} between $\widehat{\mathbf{y}_u}$ and $\mathbf{y}_u$:

$$L_{re} = \sum_{u \in \mathbf{U}} ||\mathbf{y}_u - \mathbf{S}_u \cdot \widehat{\mathbf{y}_u}||^2 \tag{4.6}$$

where $\mathbf{S}_u$ is a binary masking vector with 1 indicating items that we want to consider while calcu-

lating the reconstruction loss for u. Since the feedback usually is extremely sparse, as in [92, 13], we don't consider all the 0s in $\mathbf{y}_u$ while calculating the loss. We allocate 1 on all the items that u has left implicit feedback on (positive) and also on some randomly selected items without feedback (negative) in $\mathbf{S}_u$. And we combine the tasks of opinion elicitation and diffusion jointly, then the objective function of our final model (GoRec) becomes:

$$L = L_{re} + \lambda L_{\mathbf{G}_o}$$

Thus we reach our GoRec model (in Figure 4.4) which combines both tasks end-to-end with a hyper-parameter λ to balance the tasks.

Prediction: The reconstructed vector $\widehat{\mathbf{y}_u}$ will be used to infer user's preference on all the items, in which larger value means higher probability that the user is interested in the item. We will rank those predictions to generate the Top-K recommendations to users.

4.4 Experiment

In this section, we will evaluate the performance of the proposed GoRec model on two real-world datasets:

Dataset. We test GoRec and the baselines on both Goodreads and Epinions (summarized in Table 4.1). Empirically, we select the Top-500 accounts in the communities as KOLs. There is no overlapping between ordinary users and KOLs. We split the user-item interaction data with ratio 6:1:3 for training, validation and testing:

- **Goodreads.** We randomly sample 2 million user IDs and crawl all their interactions with books and their following information until November 2018. We filter out inactive users with fewer than 5 interactions on books. While constructing the opinion triplets, we utilize the reviews, ratings and tags provided by KOLs. For each review, we handle it with pre-processing, tokenization, and stop word removal to extract the words. Each unique word, rating level (1 to 5), or tag is treated as one type of opinion, based on which we construct the opinion triplets.

- **Epinions.** This is a public dataset with user reviews and unidirectional user-user relations [93]. Epinions is a review site on which users can write and read reviews for products. In Epinions, a user can "trust" another user, which is treated as the "follow" signal in this platform. By analyzing the "trust" relationships between users, we can see similar patterns as shown in Figure 4.2 for Goodreads. We keep active users leaving no less than 5 feedback. We ranked all the accounts based on the numbers of their followers and select the top accounts as KOLs. For KOLs, we treat all the reviews and ratings as explicit opinions. And we use the same method as in Goodreads to construct the opinion triplets. For users, we treat all their interactions with items as implicit feedback.

	#User	#Item	User-item Feedback	User-KOL Feedback	Opinion Triplet
Goodreads	15,324	36,645	1.831,826	167,054	2.8M
Epinions	6,334	8,015	81,965	63,939	0.1M

Table 4.1: Dataset Statistics.

Baselines. We compare the proposed GoRec with the following baseline methods:

- **ItemPop:** This model ranks items based on their popularity and recommends the most popular items.
- **BPRMF:** *Bayesian Personalized Ranking* [4]. It estimates user's preference on an item with the multiplication between their latent factors (MF). It is optimized with the Bayesian personalized ranking (BPR) loss [4] based on user-item interactions.
- **CDAE:** *Collaborative Denoising Autoencoder* [13]. This model is a generalization of collaborative filtering and matrix factorization. It models user-item interactions with the basic Autoencoder structure and an additional user node.

- **NGCF:** *Neural Graph Collaborative Filtering* [7]. It models user-item interactions with GNNs and *concatenates* the embeddings from different GNN layers to balance the multi-order connectivity in a bipartite graph. It is optimized with BPR loss.

Somewhat similar to our idea of incorporating the influence of KOLs is exploiting semantic knowledge for recommendation. Below are methods originally proposed to enhance recommendation with knowledge graph (KG). By treating the opinion triplets in the same way as the fact triplets in a KG, they can also consider the interactions between user, KOLs and items for recommendation:

- **MKR:** *Multi-Task Feature Learning* [83]. This model proposes to utilize the cross&compress unit to combine recommendation with the task of KG embedding. It aims to optimize AUC.

- **KTUP:** *Unifying KG Learning and Recommendation* [82]. It performs item recommendation and knowledge completion simultaneously. It enhances the basic BPRMF by transferring embeddings for relations and entities learned from KG completion.

- **CKE:** *Collaborative Knowledge Base Embedding for Recommender Systems* [80]. It proposes to combine knowledge of items from multiple resources to enhance recommendation. It uses TransR [86] to construct embeddings for the structural knowledge.

Overall Model Comparison We summarize the results of Top-K recommendation at K=5 and K=10 in Table 4.2 and 4.3. And Δ represents the improvement of GoRec over the best baseline methods. GoRec achieves the best performance under different K for both communities on all the metrics (Precision, Recall, F1 and NDCG).

Starting from ItemPop of recommending the most popular items, with matrix factorization, BPRMF can improve ItemPop by 56.61%. Then, NGCF extends BPRMF by concatenating the embedding generated from multiple GNN layers and achieves 8.13% and 4.82% improvement on Goodreads and Epinions, which shows the significance of paying attention to the high-order connectivity between users and items in recommendation.

Model	Goodreads							
	k=5				k=10			
	Precision	Recall	F1	NDCG	Precision	Recall	F1	NDCG
ItemPop	16.95	2.31	4.07	17.78	13.82	3.54	5.64	15.61
BPRMF	27.58	4.27	6.13	28.50	24.99	7.27	8.99	27.03
NGCF	29.02	4.53	6.41	30.23	26.02	7.57	9.28	28.41
CDAE	30.52	4.67	6.69	31.85	27.58	7.94	9.82	30.04
MKR	21.80	2.70	4.29	21.43	19.50	4.32	6.28	20.11
KTUP	28.70	4.22	6.14	29.76	26.08	7.23	9.14	28.16
CKE	30.99	4.43	6.50	32.38	27.82	7.46	9.58	30.28
GoRec	**34.61***	**5.06***	**7.50***	**35.88***	**31.09***	**8.58***	**11.02***	**33.61***
$\Delta(\%)$	11.68	8.35	12.11	10.81	11.75	8.06	12.21	10.99

Table 4.2: Comparing Models on top-K Recommendation in Goodreads. *All the results are in percentage.* $*$ indicates that the improvement of the best result is statistically significant compared with other methods for $p < 0.05$.

Model	Epinions							
	k=5				k=10			
	Precision	Recall	F1	NDCG	Precision	Recall	F1	NDCG
ItemPop	2.90	2.62	2.75	3.58	2.36	4.15	3.01	3.94
BPRMF	4.33	3.57	3.48	5.04	3.64	6.02	4.01	5.64
NGCF	4.73	4.04	3.86	5.53	3.82	6.33	4.21	6.03
CDAE	5.02	4.15	4.04	5.94	4.16	6.91	4.60	6.57
MKR	4.80	3.94	3.92	4.86	3.10	5.46	3.55	4.97
KTUP	4.51	3.79	3.67	5.26	3.87	6.38	4.29	5.94
CKE	4.98	4.27	4.08	5.96	4.13	6.86	4.57	6.58
GoRec	**5.45***	**4.68***	**4.47***	**6.50***	**4.43***	**7.62***	**4.97***	**7.16***
$\Delta(\%)$	8.56	12.77	9.55	9.06	6.49	10.27	8.75	8.81

Table 4.3: Comparing Models on top-K Recommendation in Epinions. *All the results are in percentage.* $*$ indicates that the improvement of the best result is statistically significant compared with other methods for $p < 0.05$.

Comparing GoRec with the baseline models which are designed to enhance the recommendation utilizing the semantic information from knowledge graphs (MKR, CKE and KTUP), we can see GoRec outperforms all of them, which indicates the proposed model is a good fit for eliciting the elite opinions and incorporating them to infer user preferences. Among those models, we find that MKR performs worst because it is designed to optimize the AUC and thus may not be a good fit for the Top-K recommendation task. KTUP improves BPRMF by transferring embeddings learned from the relational structured data. And we can see it outperforms BPRMF by 2.14% and 5.59% in recommendation for Goodreads and Epinions, which shows the effectiveness of treating the opinion triplets similar as the fact triplets in structured knowledge graph. In CKE, it encodes the structured data with TransR [86], which can provide more flexibility and is more powerful in handling the *many-to-many* relationships between KOLs and items.

Diffusion via High-order Connectivity. In Table 4.4, Compared to the basic autoencoder for recommendation (Autorec) [5], GARec comprises multiple GNN layers to encode also the high-order connectivity of users and items. The collaborative denoising Autoencoder (CDAE) [13] is an advanced version of Autorec with an additional embedding vector to characterize user preferences. In GARec, we use GNN layers to capture multi-order connectivity in the user-item interaction graph and a fully-connected layer to reconstruct the feedback vector. We find that by taking the high-order connectivity into consideration, GARec can outperform CDAE by 4.46% and 1.47% in Goodreads and Epinions correspondingly. Thus we can conclude that it is necessary to model the high-order proximity between users and items in a sparse scenario. The proposed structure is capable in capturing the opinions and signals diffusion in the community, leading to improved item recommendation.

Fusing Elite Opinions into Diffusion Process. KOLs can directly influence their followers, starting from which their opinions diffuse in the whole community. As shown in Table 4.4, while modeling user preferences, GARec+*User* extends GARec by fusing the initial user embedding with the attentive aggregation of all the KOLs the user is following, and thus outperforms GARec in both Goodreads and Epinions. This confirms the importance of capturing and modeling the

	Goodreads			Epinions		
	F1	NDCG	Δ(%)	F1	NDCG	Δ(%)
CDAE	9.82	30.04	-	4.60	6.57	-
GARec	9.99	32.20	4.46	4.70	6.62	1.47
GARec+*User*	10.75	32.72	9.20	4.87	7.00	6.21
GARec+*Item*	10.69	32.80	9.02	4.90	6.91	5.84
GoRec	11.02	33.61	12.05	4.97	7.16	8.51

Table 4.4: Ablation Analysis of GoRec (K=10). *All the results are in percentage.*

influence of KOLs to their followers in such a direct process. Additionally, KOLs can change how the community views an item by publishing reviews, ratings and tags. In this way, they add community-specific/KOL-defined features to items. In GoRec, we model this process with a fusing layer to enrich the initial embeddings of items. Thus extending from GARec, we add in the item embeddings generated from elite opinion elicitation, and then end up with GARec+*Item*. We find that in both Goodreads and Epinions, by modeling the influence of KOLs on how users view the items, we can further improve the quality of our recommendations. Also we find that the improvement from GARec+*User* is larger than that of GARec+*Item*, which indicates that the influence from whom the user is following is more direct and more significant. And the combination of both diffusion processes leads to even better improvement in both communities.

Visualization of Elite Opinions. Finally, we want to gain insight into the elite opinions we obtain with the translation-based embedding model in Section 4.3.2. Based on the opinions embedding $\mathbf{o}_r^e$ from Equation 4.2, how do the KOLs in Goodreads view or define items in several ambiguous book genres? A tag left by the KOL indicates the genre to which the KOL believes the book should belong. Thus in Figure 4.5, after deducing their dimensions with t-SNE [94], we plot the embeddings for 5 ambiguous tags ("Tag: nonfiction", "Tag: mystery", "Tag: horror", "Tag: paranormal", "Tag: historical") and the neighboring elite opinion words (in black). This figure indicates how the Goodreads community defines these genres. We find that in Goodreads, "Sherlock", "detective" or "hacker" related contents are likely to be categorized as Mystery. Books talking about "diet", "biography" or life of "youtubers" are nonfiction. KOLs describe horror contents and paranormal

contents closely which cover topics like “zombie”, “ghost”, “werewolf” and “reaper”. And the Goodreads community pays attention to “historic” books for various countries like Scotland and Germany.

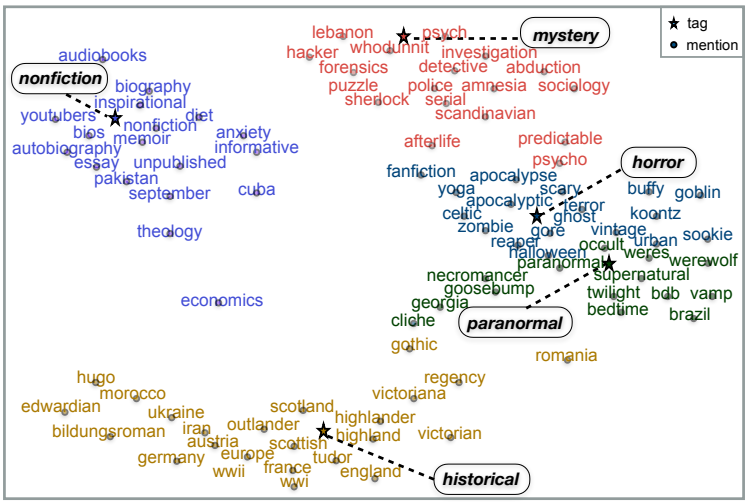

Figure 4.5: Visualization of Elite Opinions in Goodreads.

4.5 Conclusion and Future Work

Toward a sustainable recommendation system which can be resilient to dramatic changes in online communities, we investigate the influence of the “Backbone” – a small group of well-known individuals (KOLs), who can shape our views, and further impact what items we buy, what media we consume, and how we interact with online platforms We propose a novel recommendation system to provide improved item recommendation by taking the influence of the “Backbone” into consideration while exploiting user preferences. It is able to elicit the elite opinions from key opinion leaders with a translation-based embedding method. Meanwhile, building upon multiple GNN layers, the proposed framework can efficiently model the opinion diffusion process. Through experiments on Goodreads and Epinions, the proposed model outperforms state-of-the-art approaches in Top-K recommendation.

In the future, we are interested in further exploring how the influence of the “Backbone” can

be transferred cross-platform. We also want to develop a flexible model to support some newly-emerging types of opinions (like video blogs).

5. ROBUSTNESS: LEARNING FROM IMPERFECT ENVIRONMENTS[1]

5.1 Introduction

A sustainable system should be robust to the imperfection (i.e., cold-start, low-activity level users) in real-world environments, providing reliable recommendation to meet users' actual needs and aspirations. Previous research usually assumes that recommender systems can be trained on an abundant amount of clean user-item interactions. Such an assumption may not hold in the real-world environment which is imperfect, leading to an unstable performance in the recommendation systems. For example, the emergence of cold-start users with only limited interactions will degrade the predictive power of traditional recommender systems trained on users with abundant interactions. It is essential for a sustainable recommendation system to learn from imperfect situations and rapidly adapt to the real-world environment.

In this section, we will discuss our efforts on learning from two type of imperfect situations: (i) *cold-start users*: we explore the challenging problem of sequential recommendation for cold-start users with only minimal logged interactions without relying on auxiliary information. We present a novel "learning-to-learn" paradigm to model the transition patterns of users, which can make fast adaption for cold-start users in inferring their sequential interactions; (ii) *casual users*: we center around the research problem of distilling informative transition patterns from users and efficiently adapt to casual users with low-activity level in the platforms. We propose a model-agnostic framework to automatically learn a data augmentation policy using REINFORCE and improve the recommendation system using generated augmented data.

5.2 Learning-to-adapt for Cold-start Users

In many real-world scenarios, sequential recommenders may face difficulty in dealing with new users who have only limited interactions with the system, leading to inherently long-tailed

[1]Reprinted with permission from "Sequential Recommendation for Cold-start Users with Meta Transitional Learning" by Jianling Wang, Kaize Ding and James Caverlee, 2021. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. Copyright 2021 by ACM.

interaction data [95]. Ideally, an effective recommendation system should be able to recommend items to new users who have only a few interactions with items. However, most existing sequential recommenders are not designed to handle such cold-start users due to the difficulty of characterizing user preferences with limited data. Since new users may migrate away from the platform if they receive poor recommendations initially, *how to capture the preference of these cold-start users* becomes a critical question for building a satisfactory recommendation service.

Though quite a few cold-start recommendation methods have been proposed, most require side information [96, 97, 98] or knowledge from other domains [99, 100, 101] during training, and commonly treat the user-item interactions in a static way. In contrast, cold-start sequential recommendation targets a setting where no additional auxiliary knowledge can be accessed due to privacy issues, and more importantly, the user-item interactions are sequentially dependent. A user's preferences and tastes may change over time and such dynamics are of great significance in sequential recommendation. Hence, it is necessary to develop a new sequential recommendation framework that can distill short-range item transitional dynamics, and make fast adaptation to those cold-start users with limited user-item interactions.

In this work, we propose a new meta-learning framework called *MetaTL* for tackling the problem of cold-start sequential recommendation. In order to improve the model generalization capability with only a few user-item interactions, we reformulate the task of cold-start sequential recommendation as a few-shot learning problem. Different from existing methods that directly learn on the data-rich users, MetaTL constructs a pool of few-shot user preference transition tasks that mimic the targeted cold-start scenarios, and progressively learns the user preferences in a meta-learning fashion. Moreover, we build the proposed framework on top of a translation-based architecture, which allows the recommendation model to effectively capture the short-range transitional dynamics. This way the meta-learned sequential recommendation model can extract valuable transitional knowledge from those data-rich users and make fast adaptation to cold-start users to provide high-quality recommendations.

Contributions. In this section, we explore the challenging problem of sequential recommenda-

tion for cold-start users without relying on auxiliary information and propose to formulate it as a few-shot learning problem. A novel meta-learning paradigm – *MetaTL* is proposed to model the transition patterns of users, which can make fast adaption for cold-start users in inferring their sequential interactions. With extensive experiments on three real-world datasets, we verify the effectiveness of the proposed MetaTL in cold-start sequential recommendation and shows that it can bring in 11.7% and 5.5% improvement compared with the state-of-the-art in sequential recommendation and cold-start user recommendation.

5.2.1 Related Work

Meta-learning represents a line of research aims to learn a model which can adapt and generalize to new tasks and new environments with a few training samples. To achieve the goal of "learning-to-learn", there are three types of different approaches. Metric-based methods are based on a similar idea to the nearest neighbors algorithm with a well-designed metric or distance function [102], prototypical networks [103, 104] or Siamese Neural Network [105]. Model-based methods usually perform a rapid parameter update with an internal architecture or are controlled by another meta-learner model [106]. As for the optimization-based approaches, by adjusting the optimization algorithm, the models can be efficiently updated with a few examples [107, 108, 109]. In this work, we explore how to design an effective approach to handle the cold-start sequential recommendation for short sequences.

Under the complete cold-start setting with no historic interaction available for new users or items, previous works usually learn a transformation between the auxiliary information with the well-trained latent factors [81, 110, 97]. Under the incomplete cold-start setting, Dropoutnet utilizes Dropout layer to simulate the data missing problem [111]. Meanwhile, meta-learning has been applied to train a model tailored for cold-start cases. To solve the user cold-start problem, MetaRec [112] proposes a meta-learning strategy to learn user-specific logistic regression. There are also methods including MetaCF [113], Warm-up [114] and MeLU [96], adopting Model-Agnostic Meta-Learning (MAML) methods to learn a model to achieve fast adaptation for cold-start users. However, none of them are designed with the dynamics of user preferences in mind (as

is the case in sequential recommendation).

5.2.2 Methodology

In this section, we introduce the details of the proposed MetaTL. In essence, the design of MetaTL aims to answer the following research questions:

- **RQ1**: How to enable the model to transfer knowledge from data-rich users to cold-start users?
- **RQ2**: How do we capture the short-range transition dynamics in user-item interaction sequences? and
- **RQ3**: How to optimize the meta-learner for making accurate recommendations for cold-start users?

Problem Setup. We follow the sequential recommendation setup as explained in Section 2.3. Given the sequence of items $[i_1^u, i_2^u, \ldots, i_p^u, \ldots, i_n^u]$ that user u has interacted with in chronological order, the model aims to infer the next interesting item i_{n+1}^u. Specifically, in the cold-start recommendation task, we train the model on $\mathcal{U}_{train}$, which contains users with various numbers of logged interactions. Then given u in a separate test set $\mathcal{U}_{test}$, $\mathcal{U}_{train} \cap \mathcal{U}_{test} = \emptyset$, the model can quickly learn user transition patterns according to the K initial interactions and thus infer the sequential interactions. Note that the size of a user's initial interactions (i.e., K) is assumed to be a small number (e.g., 2, 3 or 4) considering the cold-start scenario.

Few-shot Sequential Recommendation (RQ1). Meta-learning aims to learn a model which can adapt to new tasks (i.e., new users) with a few training samples. To enable meta-learning in sequential recommendation for cold-start users, we formulate training a sequential recommender as solving a new few-shot learning problem (i.e., meta-testing task) by training on many sampled similar tasks (i.e., the meta-training tasks). Each task includes a $support$ set $\mathcal{S}$ and a $query$ set $\mathcal{Q}$, which can be regarded as the "training" set and "testing" set of the task. For example, while constructing a task $\mathcal{T}_n$, given user u_j with initial interactions in sequence (e.g., $i_A \xrightarrow{u_j} i_B \xrightarrow{u_j} i_C$),

we will have the a set of transition pairs $\{i_A \xrightarrow{u_j} i_B, i_B \xrightarrow{u_j} i_C\}$ as support and predict for the query $i_C \xrightarrow{u_j} ?$.

How can we generate a pool of meta-training tasks from data-rich users to mimic the targeted cold-start scenarios? Assume that we are focusing on predicting for cold-start users with K initial interactions and want to predict for their $K+1^{th}$ interactions. To construct a meta-training task, firstly, we will randomly select a user u_j from $\mathcal{U}_{train}$ and also randomly sample $K+1$ interactions from the user's logged interactions. With the $K+1$ interactions ordered chronologically $(i_1, i_2, ..., i_K, i_{K+1})$, we will have $i_1 \xrightarrow{u_j} i_2, i_2 \xrightarrow{u_j} i_3, ..., i_{K-1} \xrightarrow{u_j} i_K$ in the support set and $i_K \xrightarrow{u_j} i_{K+1}$ for the query.

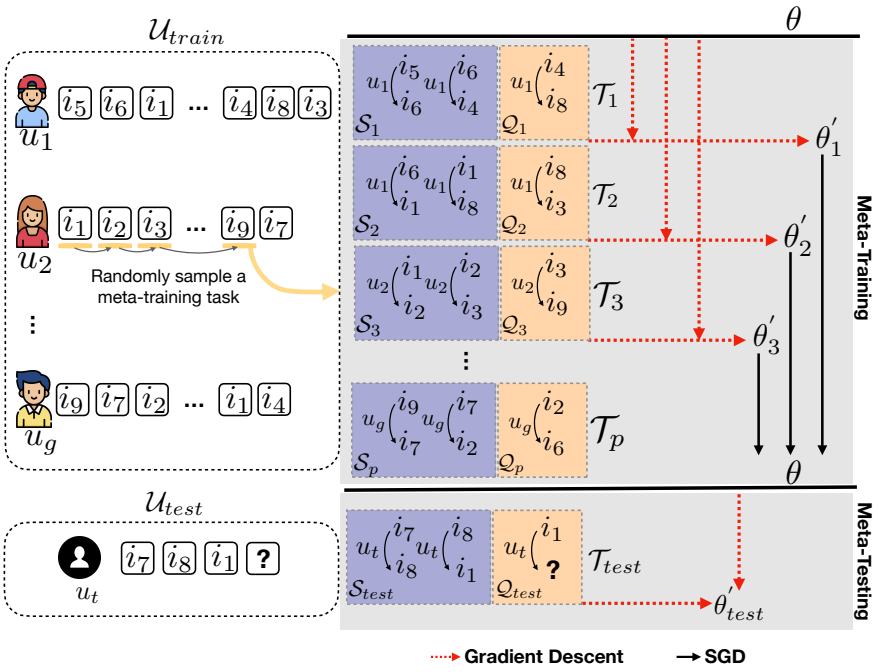

Figure 5.1: Overview of the proposed Meta Transitional Learning (MetaTL) Model.

Meta transitional Learner (RQ2). Our goal is to gain the capability of learning the transition patterns to help predict the next interactions for a new user with only a few initial interactions.

Firstly, for a task $\mathcal{T}_n$ sampled from user u_j, we want to retrieve the transitional dynamics with

the transition pairs in support set $\mathcal{S}_n$. Let $i_h \xrightarrow{u_j} i_t$ denote a transition pair and $\mathbf{i}_h$ and $\mathbf{i}_n$ as the item embedding. Following [115], then we can use a simple MLP network to represent the transitional information with item i_h and item i_t,

$$\mathbf{t}_{h,t} = \sigma\big(\mathbf{W}(\mathbf{i}_h||\mathbf{i}_t) + \mathbf{b}\big) \tag{5.1}$$

in which $\mathbf{W}$ and $\mathbf{b}$ is the trainable transform matrix and bias vector correspondingly. $\sigma(\cdot)$ denotes the Sigmoid activation function.

Since there are multiple transition pairs in a support set $\mathcal{S}_n$ for the current task $\mathcal{T}_n$, we need to aggregate the transitional information from all the pairs to generate the final representation $\mathbf{tr}_n$. As a straightforward solution, we take the average of all the transitional information from each transition pair in the support set $\mathcal{S}_n$:

$$\mathbf{tr}_n = \frac{1}{|\mathcal{S}_n|} \sum_{(i_h \xrightarrow{u_j} i_t) \in \mathcal{S}_n} \mathbf{t}_{h,t}, \tag{5.2}$$

in which $|\mathcal{S}_n|$ denotes the size of $\mathcal{S}_n$. After generating $\mathbf{tr}_n$, we need a scoring function to evaluate its effectiveness in characterizing the transition pattern of the user. Following the idea in translation-based recommendation [30], a user can be viewed as the translation (or transition) between two consecutive items in the interaction sequence. Thus if $\mathbf{tr}_n$ is effective in characterizing the transition pattern of a transition pair in $\mathcal{S}_n$ (i.e., $i_h \xrightarrow{u_j} i_t$), the translation $\mathbf{i}_h + \mathbf{tr}_n$ should be close to $\mathbf{i}_h$. That is to say, the score $s(i_h \xrightarrow{u_j} i_t) = \|\mathbf{i}_h + \mathbf{tr}_n - \mathbf{i}_h\|^2$ will have a small value. Then we calculate the marginal loss based on all the transition pairs in support $\mathcal{S}_n$ as:

$$\mathcal{L}_{\mathcal{S}_n} = \sum_{(i_h \xrightarrow{u_j} i_t) \in \mathcal{S}_n} [\gamma + s(i_h \xrightarrow{u_j} i_t) - s(i_h \xrightarrow{u_j} i'_t)]_+ \tag{5.3}$$

in which $[\cdot]_+ \triangleq max(0, \cdot)$ and γ denotes the margin. Here i'_t is a negative items without interaction from u_j.

Optimization and Fast Adaptation (RQ3). Next, we explain the procedure to optimize the model so that it can learn the transition pattern for new users with just a few interactions. We denote the meta model as a parameterized function $f_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$. Since we hope that the model can obtain a small value in $\mathcal{L}_{\mathcal{S}_n}$, we update the model $f_{\boldsymbol{\theta}}$ by minimizing the $\mathcal{L}_{\mathcal{S}_n}$ with one gradient step:

$$\boldsymbol{\theta}'_n = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{S}_n}(f_{\boldsymbol{\theta}}), \tag{5.4}$$

in which α is the task-learning rate. Note that we can extend this to perform multiple-step gradient updates based on Equation (5.4).

After updating $\boldsymbol{\theta}$ to be $\boldsymbol{\theta}'_n$ with the support set $\mathcal{S}_n$, we can generate the updated $\mathbf{tr}_n$ with $f_{\boldsymbol{\theta}'_n}$. We evaluate its performance on the query set $\mathcal{Q}_n$ with loss $\mathcal{L}_{\mathcal{Q}_n}$, which can be calculated following Eq. (5.3) with transition pair in $\mathcal{Q}_n$. Our goal is to determine the $\boldsymbol{\theta}$ that can work as a good initialization for each task. That is, it can minimize the loss on query sets across multiple meta-training tasks,

$$\boldsymbol{\theta} = \min_{\boldsymbol{\theta}} \sum_{\mathcal{T}_n \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{Q}_n}(f_{\boldsymbol{\theta}'_n}), \tag{5.5}$$

in which $p(\mathcal{T})$ is the distribution of meta-training tasks and can be obtained by randomly sampling meta-training tasks. To solve this equation, we can perform optimization via stochastic gradient descent (SGD), such that:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_n \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{Q}_n}(f_{\boldsymbol{\theta}'_n}), \tag{5.6}$$

where β is the meta step size.

When **testing** on a new user u_{test}, we will firstly construct the support set $\mathcal{S}_{test}$ based on the user's initial interactions. With Eq. (5.4), the model $f_{\boldsymbol{\theta}}$ is fine-tuned with all the transition pairs in $\mathcal{S}_{test}$ and updated to $f_{\boldsymbol{\theta}'_{test}}$, which can be used to generate the updated $\mathbf{tr}_{test}$. Given the test query $i_o \xrightarrow{u_{test}} ?$, the preference score for item i_p (as the next interaction) is calculated as $-\|\mathbf{i}_o + \mathbf{tr}_{test} - \mathbf{i}_p\|^2$.

	# Users	# Items	Avg. Length of Sequences	Splitting Timestamp
Electronics	22,685	20,712	15.26	1/1/2014
Movie	26,933	18,855	28.97	1/1/2014
Book	90,892	58,250	27.81	1/1/2017

Table 5.1: Dataset Statistics.

5.2.3 Experiment

In this section, we report our experiments over multiple datasets to evaluate the performance of the proposed model in cold-start sequential recommendation.

Datasets. We adopt three public real-world datasets: ***Electronics*** is adopted from the public Amazon review dataset [31], which includes reviews ranging from May 1996 to July 2014 on Amazon products belonging to the "Electronics" category. ***Movie*** is similarly drawn from the "Movie" category of the same Amazon review dataset. For both, we treat a user review as an interaction. ***Book*** is scraped from Goodreads, a book platform for users to tag, rate and review books. We treat all these interactions on items equally.

For all of the datasets, we filter out items with fewer than 10 interactions. We split each dataset with a corresponding cutting timestamp T, such that we construct $\mathcal{U}_{train}$ with users who have interactions before T and construct $\mathcal{U}_{test}$ with users who start their first interactions after T. Summary statistics of the datasets can be found in Table 5.1. When evaluating few-shot sequential recommendation for a choice of K (i.e., the number of initial interactions), we keep K interactions as initialization for each user in $\mathcal{U}_{test}$ and predict for the user's next interactions.

Baselines. Aside from a standard matrix factorization method (**BPR-MF**) [4], we compare the proposed MetaTL model with two categories of widely used recommendation models for Top-N recommendation: (i) *Sequential recommendation baselines* utilize different methods to capture the sequential patterns in the interaction sequences of users. **TransRec** [30] embeds items into a "transition space" and learns a translation vector for each user. **GRU4Rec** [6], **TCN** [21] and **SASRec** [23] rely on Gated Recurrent Units, the simple convolutional generative network, and the

self-attention layers to learn sequential user behaviors, respectively. **BERT4Rec** [24] adopts the bi-directional transformer to extract the sequential patterns and it is the state-of-the-art for sequential recommendation; (ii) *Cold-start baselines* include methods focusing on providing accurate recommendations for cold-start users with limited information. **MeLU** [96] learns a user preference estimator model based on Model-Agnostic Meta-Learning (MAML), which can be rapidly adapted for cold-start users. We modify MeLU as in [113] to fit for the case without auxiliary information. **MetaCF** [113] learns a collaborative filtering (CF) model which can quickly adapt to new users. We adopt the version on top of NGCF [7] for better performance.

Overall Model Comparison. We compare the performance of MetaTL and the baseline models under $K = 3$ and report the results in Table 5.2. The best performing method in each column is boldfaced, and the second best method is marked with †. MetaTL achieves the best performance under different evaluation metrics in all of the datasets, illustrating the effectiveness of MetaTL in providing accurate sequential recommendation for cold-start users with limited interactions.

	Electronics		**Movie**		**Book**	
	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR
BPR-MF	0.066	0.123	0.025	0.083	0.043	0.098
TransRec	0.183	0.296	0.208	0.321	0.335	0.454
GRU4Rec	0.185	0.301	0.189	0.309	0.330	0.466
TCN	0.182	0.303	0.186	0.314	0.349	0.489
SASRec	0.193	0.318	0.211	0.345	0.347	0.488
BERT4Rec	0.200	0.323	0.220	0.351	0.369	0.513
MeLU	0.172	0.265	0.168	0.289	0.318	0.423
MetaCF	0.210†	0.330†	0.234†	0.365†	0.398†	0.528†
MetaTL	**0.224**	**0.352**	**0.258**	**0.380**	**0.420**	**0.555**

Table 5.2: Comparison of Different Models under $K = 3$. The improvement of MetaTL is statistically significant compared with the next-best model with $p < 0.05$

Starting from the simplest collaborative filtering (BPR-MF), we find that it performs weakly since it ignores the dynamic patterns in the user interaction sequences and fails to learn effective embeddings for cold-start users. TransRec also becomes ineffective in learning translation embeddings for cold-start users since there is insufficient data to update the embeddings. Then we compare the performance of various neural models for sequential recommendation. We can see that GRU4Rec and TCN perform the worst. SASRec and BERT4Rec (utilizing transformers to extract the sequential patterns) work better since they are able to aggregate the items with attention scores and thus obtain more informative representations for users with limited interactions.

MeLU and MetaCF are both meta-learning based methods for providing cold-start recommendations. Since MeLU requires side information for both users and items, we treat their historic interactions as the side information as in [113]. Alas, MeLU is unable to obtain satisfying results since it designed for scenarios with abundant auxiliary user/item information which is absent in this case. Meanwhile we find that MetaCF can achieve the second-best performance for sequential recommendation, illustrating the importance of fast adaption in cold-start scenario. It still falls behind the proposed MetaTL since it is unable to learn the transition patterns for cold-start sequential recommendation.

Evaluation of MetaTL. To further evaluate the effectiveness of the proposed MetaTL model, we compare it with its variants and some of the baselines under different K values (i.e., the number of initial interactions) in Figure 5.2 and 5.3. Note that *MetaTL–* is the simplified version of MetaTL by removing the MAML optimization step. BERT4Rec is the state-of-the-art sequential recommendation method and MetaCF is the strongest cold-start baseline from our original experiments (and illustrates the performance of CF with meta-learning).

In both datasets, BERT4Rec loses the prediction power on sequences consisting of only a few items and thus performs weakly in the cold-start sequential recommendation task. Even without the fast adaptation learning module, MetaTL– can outperform BERT4Rec since it is carefully designed to learn the transition patterns on extremely short sequences. However, it still falls behind MetaCF, which demonstrates the necessity of training a model with fast adaptation in cold-start

scenarios. With the well-designed optimization steps and meta transitional learner, the proposed MetaTL can further improve MetaTL– and outperform both the state-of-the-art methods in sequential recommendation and cold-start user recommendation with different numbers of initial interactions.

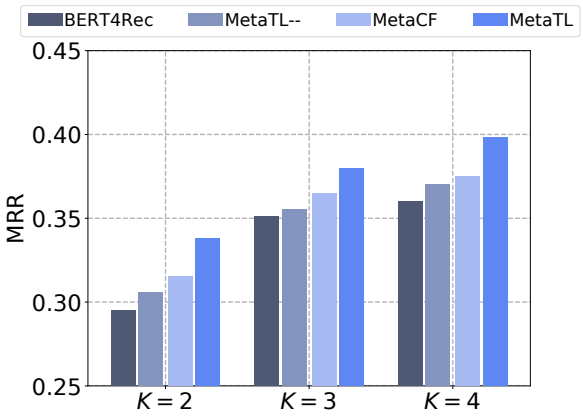

Figure 5.2: Comparison for different model variants *w.r.t.* K in **Movie**.

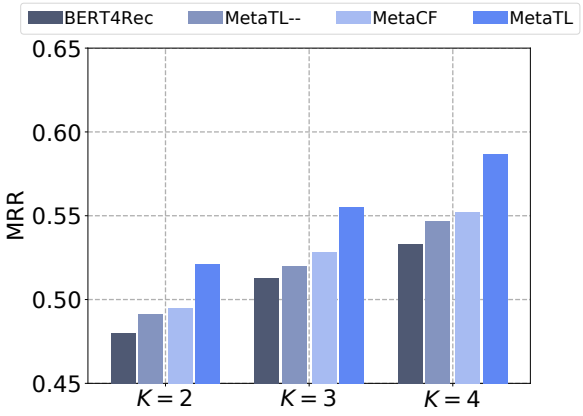

Figure 5.3: Comparison for different model variants *w.r.t.* K in **Book**.

5.3 Learning-to-augment for Casual Users

Besides the cold-start users, a recommendation systems would also come into trouble when faced with users with low-activity level, e.g., the users visit the platforms occasionally but with a few interactions. It is also essential for a sustainable recommendation system to generate robust recommendation to these imperfect test cases (i.e., low-activity level users).

Users coming to the online platform are often heterogeneous in activity levels. There usually exists a set of ***core users*** who visit the platform regularly and consistently, while others are ***casual users*** who tend to visit the platform occasionally. The heterogeneity in activity levels can lead to distinct transitional patterns between these two groups of users [116, 117]. As shown in Figure 4.5(a), consecutively interacted items are less concentrated, and of lower similarity in casual users than core as they come to the platform less frequently.

Sequential recommenders trained predominantly on interaction data from core users often fail to capture the activity patterns of casual users and, as a result, provide less satisfactory recommendations for casual users. As shown in Figure 5.4, the self-attention based recommender (SASRec [23]) performs significantly worse on casual users than on core users in all sequence lengths. *How to improve the recommendation for casual users without sacrificing the performance on core users* is a critical challenge for building satisfactory recommendation services for all.

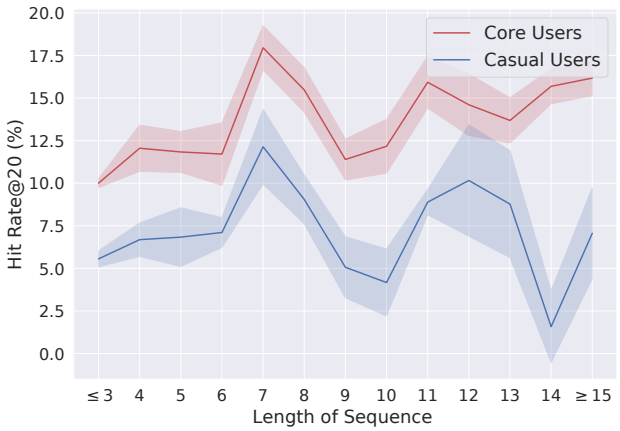

Figure 5.4: A recommendation system always performs worse on casual users than on core users, indicating the difficulty of making recommendations for casual users.

Existing methods such as cold-start recommendation [110, 97, 111, 118] or cross-domain approaches [119, 120, 121, 122, 123], mainly focus on addressing the data scarcity, but fail to handle the activity disparity between different types of users in the system. Although there are often more casual users than core users on the platforms, they left much fewer interactions in total compared to the core users. How to distill informative transition patterns from core users and efficiently adapt to casual users is the main research question we aim to tackle here. Inspired by the recent advances in data augmentation techniques [124, 125, 126, 127], we set out to generate augmented data sequences from core users to mimic the behavior patterns of casual users. While many augmentation techniques have been studied for continuous inputs such as images [128, 129], augmentation on user activity data consisting of sequential discrete item IDs is still under-explored. Meanwhile, compared with core users that tend to leave consistent and informative interaction sequences, casual users usually have noisier and more diverse behavior sequences as shown in Figure 5.4. It is an open question to find an effective data augmentation method to generate augmented sequences that inherit informative transition patterns from core users and improve casual user recommendations.

To tackle the aforementioned challenges, we propose a model-agnostic "Learning to Augment" framework – *L2Aug*, which bridges the gap between casual and core users for sequential recommendation systems. Specifically, we develop a data augmentor which decides on a series of augmentation actions on the input sequence to generate augmented data sequences. From the "Learning to Augment" perspective, this data augmentor is trained to conduct effective data augmentation to maximize the performance of the *target model* (i.e., recommender). Framing this as learning a data augmentation policy, the data augmentor (agent) generates context/state and chooses the augmentation action. Meanwhile, the target model is updated with the augmented data sequences, and its performance improvement on a meta validation set is used as the reward to guide the learning of the augmentation policy. Through alternating between the data augmentation step using the recommender performance as the reward, and improving the recommender with the augmented data, the two modules reinforce each other and progressively improve both the data augmentation and recommendation quality. As a result, this builds an adaptive recommendation system, which can

distill informative transition patterns from core users and adapt to casual users with dramatically different interaction patterns.

Contribution. In this section, through the data analysis, we investigate the disparity between core and casual users in their transitional patterns within the interaction sequences, and study the feasibility of bridging the gap between sequential recommendation for core and casual users from the data augmentation perspective. Then we propose a model-agnostic framework *L2Aug* to learn a data augmentation policy using REINFORCE and improve the recommendation system using generated augmented data. We evaluate *L2Aug*, on top of various SOTA sequential recommendation models, on four real-world datasets, and show that it outperforms other treatment methods and achieves the best recommendation performance on both core and casual users.

5.3.1 Motivation

In this section, we conduct an initial investigation with data sampled from the public Amazon review dataset [31], to explore the distinct behavior patterns between casual and core users, and then examine the feasibility of applying data augmentation in bridging the gap between them.

Firstly, to investigate the interest continuity in item consumption history of different users, we compute the *correlation between consecutive items* in their interaction sequences. We apply the bag-of-words model on item descriptions to obtain the item embeddings and then calculate the cosine similarity between the embeddings of consecutive items in the interaction sequences. In Figure 5.5, we can observe that the consecutive items consumed by core users are more similar. It confirms our hypothesis that core and casual users behave differently and the interests of casual users are less concentrated compared with core users.

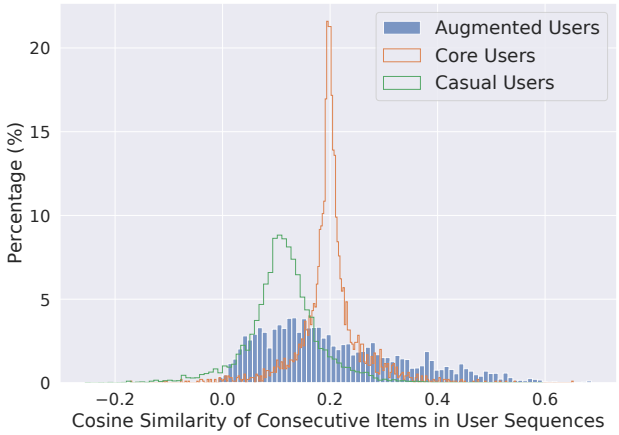

Figure 5.5: Comparison of interest continuity (i.e., the similarity of items consumed consecutively) for different user groups.

Next, as interaction sequences of core users tend to be longer and denser than those of casual users, the most straightforward approach for data augmentation is to *randomly drop* part of the interactions from core users. We adopt this approach in this initial investigation and train a SASRec model [23] with the augmented data. In Figure 5.6, we visualize the performance for casual user recommendation by varying the percentage of dropped interactions. When the dropping percentage is equal to 0, none of the interactions from core users is dropped, thus the recommender is trained on the original data from both core and casual users. On the contrary, when the dropping percentage is equal to 1.0, all interactions from the core users are dropped, meaning that the recommender is trained only on the original data from casual users. It can be observed that the recommender system achieves improved performance on casual users when we start to drop interactions from the core users, which suggests that the synthetic data can help improve casual user recommendation. However, as the dropping percentage increases, discarding too much information negatively impacts casual user recommendation. These observations motivate us to search for more fine-grained and controlled augmentation policies.

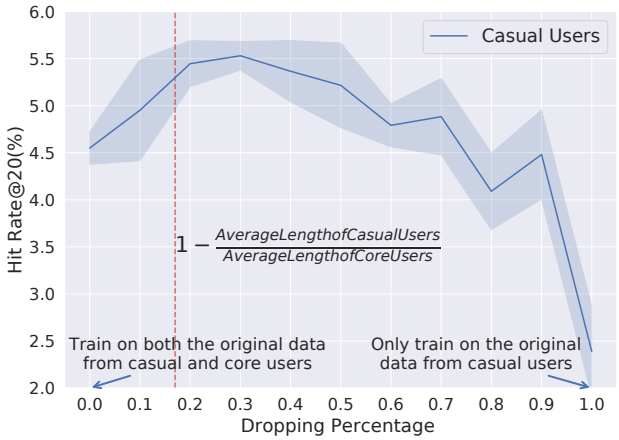

Figure 5.6: Data from core users is helpful for training a model for casual users. Furthermore, randomly dropping part of the interactions from core users can further improve the performance on casual users.

As shown in Figure 5.4, the self-attention based recommender (SASRec [23]) performs significantly worse on casual users than on core users in all sequence lengths. *How to improve the recommendation for casual users without sacrificing the performance on core users* is a critical challenge for building satisfactory recommendation services for all.

5.3.2 Related Work

Several research topics in recommender systems are related to improving casual user recommendations. [130] define the focused learning problem and propose to find different optimal solutions for different user groups through hyperparameter optimization and a customized matrix factorization objective. There are other efforts trying to learn multiple recommendation models and then select the best performing one from the pool for each user [131, 132]. Another line of research focuses on domain transfer and improving the recommendation for users who have very few observed interactions [133]. Most of these works rely on side information or contextual data [96, 97, 100, 101]. [95] proposes to learn the transferable parameters between data-rich head users and data-ill tail users, leading to an improved recommendation for both groups. On the contrary, we aim to bridge the gap between users who visit more frequently and users who casually visit the

service via data augmentation.

Data augmentation is a widely adopted strategy for training deep neural models [134, 135] in computer vision [124, 125], speech [126, 136] and natural language processing [127, 137]. The idea is to generate more diverse data samples to improve model robustness. While most of the early work manually designs dataset-specific augmentation strategies or implementations, recent attention has been paid to automating the process of generating effective augmentation for the target dataset. Generative Adversarial Networks (GANs) have been commonly used to generate additional data by learning the distribution of the training data under the MinMax framework [138, 139, 140, 141]. As an alternative to GAN-based methods, AutoAugment [124] automatically searches for the best augmentation policy using reinforcement learning. However, there is still a research gap between generating additional images or textual data and augmenting sequential user interaction data; the latter is still understudied. Recently, CL4Rec [142] proposes to construct different views of an interaction sequence with simple data augmentation using the contrastive loss, resulting in a more robust recommendation system. We focus on learning to generate effective augmented interaction sequences that mimic the patterns of casual users while inheriting informative transition patterns of core users.

5.3.3 Methodology

In this section, we elaborate on the design of *L2Aug*, organized around two guiding research questions: 1) How to perform data augmentation on sequential data to support various operations (e.g., remove, keep or substitute) on items within the sequence? 2) How to learn an effective data augmentation policy to achieve the goal of improving causal user recommendation?

Problem Formulation. In this work, the users on the platform can be partitioned into two groups: casual users $\mathcal{U}_{casual}$ and core users $\mathcal{U}_{core}$ based on their activity levels (i.e., the frequency they visit the platform), s.t. $\mathcal{U}_{casual} \cap \mathcal{U}_{core} = \emptyset$ and $\mathcal{U}_{casual} \cup \mathcal{U}_{core} = \mathcal{U}$. Note that each item is mapped to a trainable embedding vector associated with its unique ID. In this work, we do not consider any auxiliary information for users and items. We follow the sequential recommendation setup as discussed in Section 2.3. Given the sequence of items that user u has interacted with

in chronological order $\mathbf{S}^u = [i_1^u, i_2^u, \ldots, i_p^u, \ldots, i_n^u]$, where i_p^u represents the pth item u interacted with, the objective of the sequential recommendation model (**target model**) $f_{\boldsymbol{\theta}}$ is to infer the next interesting item $i_{u,n+1}$ for user u.

Framework Overview. We propose *L2Aug*, illustrated in Figure 5.7, to learn the data augmentation policy for improving casual user recommendation. There are two main components in the design: a *recommender* to make sequential recommendations and a *data augmentor* to generate synthetic interaction sequences by applying the learned data augmentation policy on the input sequences. These two components are alternately trained. Specifically, for each batch of sequences sampled from $\mathbf{S}_{core}$, the *data augmentor* learns to take a series of augmentation actions (e.g., remove, keep or substitute) so the generated synthetic sequences can improve the performance of the *recommender*. Framing this as learning a data augmentation policy, the data augmentor (agent) generates context/state and chooses the augmentation action. Meanwhile, the target model is updated with the augmented data sequences and its performance improvement on a meta validation set is used as the reward to guide the training of the augmentor. Through alternating between the data augmentation step using the recommender performance as the reward, and improving the recommender with the augmented data, the two modules reinforce each other and progressively improve both the data augmentation and recommendation quality.

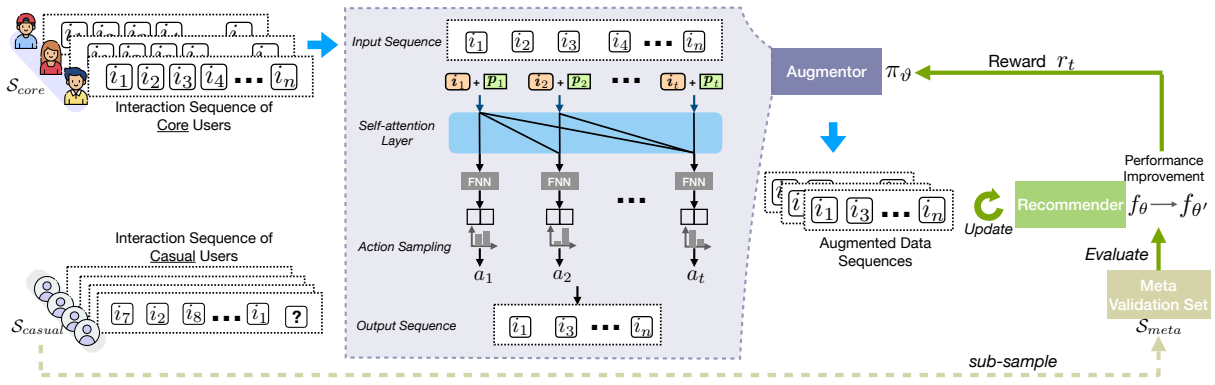

Figure 5.7: The proposed model-agnostic "Learning to Augment" framework – *L2Aug*.

Learning Augmentation Policy. Our goal is to *learn a discrete sequential data augmentation policy* for *maximizing the performance of recommendation systems* on casual users. Inspired by preliminary studies in Section 5.3.1, we set out to generate augmented (synthetic) data sequences to mimic the behavior patterns of casual users by editing the core user sequences. For each batch of interaction sequences sampled from $\mathcal{U}_{core}$, we consider as a *data augmentation task* that generates the synthetic sequences by taking a series of editing actions from {keep, drop, replace, ...} for items in the input sequences sequentially. Since the data augmentation process is non-diffentiable, we adopt the policy learning framework described below to enable the training:

- **Context/State**: Let $\mathbf{S}_u$ denote an interaction sequence from $\mathcal{U}_{core}$. When encountering item i_k^u in sequence $\mathbf{S}^u$, a vector $\mathbf{h}_k$ that encodes the subsequence $\mathbf{S}^u_{[1:k]}$ is regarded as the context or state representation. The detailed model to obtain the state vector will be discussed later.

- **Action**: For simplicity, we use two actions – "Keep" and "Drop". The model could be extended to support multiple actions (i.e., more than two). At step k, for item i_k^u in sequence $\mathbf{S}_u$, we need to decide on the action $a_k^u \in \{0, 1\}$ to keep or drop item i_k^u when generating the augmented sequence from sequence $\mathbf{S}^u$. Note that $a_k^u = 0$ indicates dropping the item and $a_k^u = 1$ means to keep it.

- **Meta Valication Set ($\mathcal{U}_{meta}$)**: To guide the training of the data augmentor, from the set of N casual users $\mathcal{U}_{casual}$, we randomly sample a small subset of M users ($M \ll N$) and construct the meta validation set $\mathcal{U}_{meta}$ with their interaction sequences. During the training process, the performance of the target model on $\mathcal{U}_{meta}$ is computed as the reward for learning the augmentation policy.

- **Reward**: The reward function aims to guide the agent to learn the augmentation policy in order to maximize the performance of the target model. Each batch of the augmented sequences is used to train the target model and leads to a performance change of the target model, which can be regarded as the reward to the data augmentor. In offline settings, it can be the *update of recommendation performance* on $\mathcal{U}_{meta}$ measured by offline metrics

(e.g., NDCG, Hit rate and Mean Reciprocal Ranking). In online settings where we can simulate user response on counterfactual recommendations, the reward can be the user response returned by the environment (e.g., engagement, rating or conversion). More details about the metrics can be found in Section 2.2. In our experiments, we use the performance gain on both NDCG and Hit Rate as the reward for offline experiment, and use the change of simulated rating as the reward for online experiment. Following [143, 144, 145], the policy network for data augmentation is updated on a delayed reward received after feeding the generated augmented data to the recommender.

Data Augmentor. Since most of the existing data augmentation methods are designed for continuous feature spaces [128, 129], they are not fit for handling sequence data consisting of discrete item IDs in our case. We propose the *Data Augmentor*, which generates a synthetic sequence by encoding the input sequence as the **context/state** representations, and deciding on the editing **actions** on the input sequence.

Given a sequence of interacted items $\mathbf{S}^u = [i_1^u, \ldots, i_k^u, \ldots, i_t^u]$ for a core user $u \in \mathcal{U}_{core}$, the Data Augmentor needs to decide on the editing action on each item. To make the decision on each item i_k^u, the agent needs to encode the subsequence $\mathrm{S}_{[1:k]}^u$, which in turn requires a representation of each item. We encode two pieces of information in the individual item representation: the item content itself and its position. In other words, for each item i_k^u in $\mathbf{S}^u$, we have $\mathbf{e}_k = \mathbf{i}_k + \mathbf{p}_k$. Here, $\mathbf{i}_k$ is the embedding for item i_k^u and $\mathbf{p}_k$ is the positional embedding of position k, which is used to retain the order information.

With the individual item representation, any sequential embedding model including RNN, Bidirectional-RNN or Transformers [22, 146] can be used to encode the subsequence $\mathrm{S}_{[1:k]}^u$ to produce a context/state representation. In this work, we adopt the self-attention model [22] to produce the state $\mathbf{h}_k$. Self-attention [22] is designed to match a sequence against itself and thus uses the same objects as the queries, keys, and values. We will transform the position-aware embeddings with $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ to generate the query vectors, key vectors, and value vectors, correspondingly. Then we can generate the state representation for each item i_k^u by encoding its correlation

with other items in the sequence via:

$$\mathbf{h}_k = \sum_{j \leq k} \sigma_{kj} \mathbf{W}_V \mathbf{e}_j \quad \text{and} \quad \sigma_{kj} = \frac{S(\mathbf{W}_Q \mathbf{e}_k, \mathbf{W}_K \mathbf{e}_j)}{\sum_{j \leq k} S(\mathbf{W}_Q \mathbf{e}_k, \mathbf{W}_K \mathbf{e}_j)}, \tag{5.7}$$

in which the function $S(\cdot, \cdot)$ is used to denote the similarity score between two elements. In particular, we use the *Scaled Dot-Product Attention* [23, 22] to calculate the scores: $S(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} / \sqrt{D}$, where D is the dimension size and used for normalization when calculating the similarity scores. Once obtaining each individual ***context/state*** representation $\mathbf{h}_k$, the agent produces a policy π_k over the action space $\mathcal{A}$ through

$$\pi_{\boldsymbol{\vartheta}}(\cdot | \mathbf{h}_k) = \text{Softmax}(\mathbf{W}_A \cdot \mathbf{h}_k). \tag{5.8}$$

Here $\mathbf{W}_A \in \mathcal{R}^{|\mathcal{A}| \times |\mathbf{h}|}$ is a trainable weight matrix. We use $\boldsymbol{\vartheta}$ to denote all the parameters involved in building the augmentation policy network. Each dimension in $\pi_{\boldsymbol{\vartheta},k}$ represents the probability of a specific action at step k. The agent then decides on the editing ***action*** for the item by sampling with the probabilities.

Augmentation Policy Optimization. To optimize for the data augmentation policy, we aim to maximize expected rewards on improved recommendation quality, which can be defined as:

$$J(\boldsymbol{\vartheta}) = \max_{\boldsymbol{\vartheta}} \mathbb{E}_{\pi_{\boldsymbol{\vartheta}}} [r_t], \tag{5.9}$$

where r_t is the reward computed as the improvement in recommendation performance when feeding the augmented data generated from input batch t. In Algorithm 1 line 12, we use the Hit Rate HT to evaluate the recommendation performance and the difference of the two as the reward. Note that Hit Rate can be replaced with any performance metrics or their combination. We update the parameters $\boldsymbol{\vartheta}$ via policy gradient:

$$\boldsymbol{\vartheta} \leftarrow \boldsymbol{\vartheta} + \alpha \nabla_{\boldsymbol{\vartheta}} \tilde{J}(\boldsymbol{\vartheta}), \tag{5.10}$$

Algorithm 1: *L2Aug* Training Process

Input: Training sequences $\mathcal{S}_{core}$, $\mathcal{S}_{meta}$, $\mathcal{S}_{casual}$ and the pre-trained recommendation system $f_{\boldsymbol{\theta}}$, update frequency F

Output: A fine-tuned recommendation system $f_{\boldsymbol{\theta}}$

Random initialize the data augmentor $\pi_{\boldsymbol{\vartheta}}$ and $i = 0$.
while *not converge* **do**
  Sample B samples $\mathcal{S}_B$ from $\mathcal{S}_{core}$
  // `Augmentation by Data Augmentor`
  for $u = 1 \rightarrow B$ **do**
    for $k = 1 \rightarrow |s_u|$ **do**
      Compute the state representation for the k_{th} item
      Sample the augmentation action
    Obtain the augmented sequence
  // `Augmentation Policy Optimization`
  Fine-tune $f_{\boldsymbol{\theta}}$ with the batch of augmented sequences to get $f_{\boldsymbol{\theta}'}$
  Calculate the reward $r_t = \text{HT}(f_{\boldsymbol{\theta}'}, \mathcal{S}_{meta}) - \text{HT}(f_{\boldsymbol{\theta}}, \mathcal{S}_{meta})$
  Update $\pi_{\boldsymbol{\vartheta}}$ according to Eq. (5.10) and (5.11)
  // `Replay and Update the Recommender`
  if $i \bmod F == 0$ **then**
    Sample B samples $\mathcal{S}_{train}$ from $\mathcal{S}_{core}$
    Generate synthetic samples $\mathcal{D}_{syn}$ from $\mathcal{S}_{train}$ with $\pi_{\boldsymbol{\vartheta}}$
    Sample B samples $\mathcal{S}'_{train}$ from $\mathcal{S}_{casual}$ and $\mathcal{S}_{core}$
    Update the recommender $f_{\boldsymbol{\theta}}$ with S_{syn} and $\mathcal{S}'_{train}$
  $i \leftarrow i + 1$
return The fine-tuned recommendation system $f_{\boldsymbol{\theta}}$

in which α is the learning rate. With the obtained rewards, the gradient can be computed by:

$$\nabla_{\boldsymbol{\vartheta}} \tilde{J}(\boldsymbol{\vartheta}) = r_t \sum_{u \in \mathcal{S}_B} \sum_{k=1}^{|\mathbf{S}_u|} \nabla_{\boldsymbol{\vartheta}} \log \pi_{\boldsymbol{\vartheta}}(a_{u,k} | \mathbf{h}_k), \tag{5.11}$$

Details of the training process are shown in Algorithm 1. Note that the same meta reward over the batch is assigned to all the augmentation decisions taken within the batch. In essence, the obtained reward based on recommendation improvement computed on the meta validation set is used to guide the learning of the data augmentation policy. The augmented sequences in return are used to further improve the performance of the recommender. During the training process, the data augmentor and recommender system can reinforce each other, and progressively improve the data

augmentation and recommendation quality.

Replay. To ensure that the model still achieves satisfying performance on core users, we adopt the replay strategy [147] to avoid forgetting. Besides the synthetic sequences, the recommendation system is also updated with the original data sequences from core users, leading to a recommender that has improved performance on casual users without sacrificing the performance on core users.

5.3.4 Experiment

Datasets. To examine the performance of the proposed method, we conduct experiments on four real-world datasets. Table 5.3 shows the summary statistics for each dataset. ***Amazon_CDs*** is adopted from the public Amazon review dataset [31], which includes reviews spanning May 1996 – July 2014 on products belonging to the "CDs and Vinyl" category on Amazon. Similarly, ***Amazon_Books*** and ***Amazon_Movies*** are from two of the largest categories – "Books" and "Movies" of the same Amazon review dataset. To further investigate the performance in other application scenarios, we include another public dataset from an idea-sharing community – ***Goodreads***, on which users can leave their reviews and ratings on books [148]. For all the datasets, all the items the user has *interacted* with (reviewed) form the user interaction sequence $\mathbf{S}_u$. We use the average time gap between their consecutive interactions to differentiate between casual and core users: core users are those with average time gaps of less than 30 days; others are labeled as casual users.

Table 5.3: Summary statistics for the datasets.

	# Items	# Casual Users	# Core Users	Avg. # Interactions
Amazon_CDs	22,685	2,176	1,022	23.75
Amazon_Books	17,443	11,083	3,457	31.71
Amazon_Movies	11,079	10,020	2,808	14.06
Goodreads	65,864	11,836	5,017	130.03

Baselines. As the proposed method is model agnostic, we apply it to various sequential recommendation models and compare its performance with other model-agnostic treatment methods to examine its effectiveness. We select three major *sequential recommendation models* – **GRU4Rec** [6], **SASRec** [23] and **NextItNet** [21], which are built on top of Gated Recurrent Units (GRU), self-attention layers and stacked 1D dilated convolutional layers to capture the sequential patterns in user interaction sequences respectively. They are widely used in many applications and serve as the foundation for many advanced recommender systems.

Since there are no previous works focusing on improving casual user recommendations, for each sequential recommendation model, we compare the proposed *L2Aug* with the following *treatment methods* which are proved to alleviate the performance gap between different user groups.

- **Random**: It randomly drops the interactions of core users to obtain the synthetic data, which are combined with the original data (both core & casual users) for training the recommender.
- **Focused Learning** [130]: It treats the casual users $\mathcal{U}_{casual}$ as the focused set and performs a grid search for the best performing hyperparameters (i.e., the regularization) for improving recommendation accuracy on the focused set.
- **Adversarial Reweighting** [149]: It plays a minimax game between a recommender and an adversary. The adversary would adversarially assign higher weights to regions where the recommender makes significant errors, in order to improve the recommender's performance in these regions.

Overall Model Comparison for Casual User Recommendation. We summarize the average performance of different baseline methods on all of the datasets in Table 5.4 and 5.5. When combined with various sequential models, the proposed L2Aug outperforms all of the other treatment methods and achieves the best recommendation for casual users. In the following, we present more in-depth observations and analyses:

- The simplest treatment of *randomly* dropping part of the interactions from core users helps to improve casual user recommendations compared to the model trained on original data. This

observation verifies the hypothesis that data augmentation can help bridge the gap between core and casual users.

- By learning a recommendation model with a special focus on casual users, the *focused learning* treatment can help improve model performance on casual users. Meanwhile, since recommenders tend to make inaccurate predictions on casual users, *adversarial reweighting* can guide the recommender to improve its performance on casual users, leading to more accurate recommendations for them.

- In general, the proposed L2Aug significantly outperforms all the baseline treatments on improving casual user recommendations for various widely-used sequential recommendation models. Take the *Amazon_CDs* dataset as an example, we find that L2Aug achieves 9.59%, 6.58%, and 9.90% improvements for NDCG@5 with GRU, NextItNet and SASRec, respectively, compared against the best performing baseline treatment. We can conclude that L2Aug is effective in solving the challenging problem of improving casual user recommendations.

Overall Model Comparison for Core User Recommendation. Besides the performance on casual users, we also report the recommendation performance on core users in Amazon_CD in Figure 5.8 and 5.9, measured by NDCG@5 and HT@5. Although *focused learning* improves the recommendation on casual users, it loses its prediction power on core users. The *adversarial reweighting* treatment, aiming at improve challenging data samples rather than specific user groups, improves core user recommendations in some cases, but not always. In contrast, the proposed L2Aug improves core user recommendations with various sequential recommendation models and outperforms all the other baseline treatments. Importantly, similar patterns are also observed in other datasets. These results showcase its effectiveness in bridging the gap between recommendation for casual users and core users, leading to overall recommendation improvement.

Table 5.4: Performance on casual user recommendation of various models on different datasets with K=5.

Method	Amazon_CDs		Amazon_Books		Amazon_Movies		Goodreads	
	NDCG@5 (%)	HT@5 (%)	NDCG@5 (%)	HT@5 (%)	NDCG@5 (%)	HT@5 (%)	NDCG@5 (%)	HT@5 (%)
GRU	0.64 ± 0.04	1.05 ± 0.09	0.66 ± 0.04	1.10 ± 0.04	1.11 ± 0.07	1.81 ± 0.08	1.32 ± 0.06	2.09 ± 0.08
w/ Random	0.66 ± 0.06	1.13 ± 0.11	0.71 ± 0.03	1.15 ± 0.05	1.22 ± 0.01	1.95 ± 0.12	1.44 ± 0.13	2.29 ± 0.14
w/ Focused	0.69 ± 0.03	1.17 ± 0.08	0.74 ± 0.02	1.19 ± 0.04	1.28 ± 0.06	2.08 ± 0.07	1.43 ± 0.05	2.37 ± 0.11
w/ Adversarial	0.73 ± 0.02	1.24 ± 0.06	0.73 ± 0.03	1.20 ± 0.05	1.34 ± 0.05	2.23 ± 0.06	1.46 ± 0.04	2.21 ± 0.08
w/ L2Aug	$\mathbf{0.80 \pm 0.03}$	$\mathbf{1.37 \pm 0.07}$	$\mathbf{0.75 \pm 0.02}$	$\mathbf{1.25 \pm 0.03}$	$\mathbf{1.43 \pm 0.04}$	$\mathbf{2.35 \pm 0.05}$	$\mathbf{1.53 \pm 0.03}$	$\mathbf{2.45 \pm 0.05}$
NextItNet	1.12 ± 0.16	1.69 ± 0.18	0.97 ± 0.03	1.56 ± 0.06	1.30 ± 0.05	2.13 ± 0.09	2.05 ± 0.11	2.97 ± 0.13
w/ Random	1.17 ± 0.18	1.88 ± 0.20	1.08 ± 0.05	1.71 ± 0.07	1.45 ± 0.08	2.26 ± 0.09	2.13 ± 0.10	3.11 ± 0.16
w/ Focused	1.22 ± 0.15	2.07 ± 0.16	1.13 ± 0.04	1.68 ± 0.03	1.47 ± 0.06	2.42 ± 0.08	2.25 ± 0.09	3.32 ± 0.13
w/ Adversarial	1.52 ± 0.11	2.39 ± 0.13	1.15 ± 0.04	1.75 ± 0.05	1.58 ± 0.06	2.56 ± 0.08	2.42 ± 0.06	3.43 ± 0.11
w/ L2Aug	$\mathbf{1.62 \pm 0.09}$	$\mathbf{2.44 \pm 0.10}$	$\mathbf{1.24 \pm 0.05}$	$\mathbf{1.87 \pm 0.04}$	$\mathbf{1.71 \pm 0.05}$	$\mathbf{2.74 \pm 0.07}$	$\mathbf{2.51 \pm 0.07}$	$\mathbf{3.62 \pm 0.10}$
SASRec	1.83 ± 0.10	2.77 ± 0.16	1.13 ± 0.05	1.78 ± 0.06	1.72 ± 0.05	2.71 ± 0.08	2.29 ± 0.07	3.49 ± 0.10
w/ Random	1.85 ± 0.15	2.81 ± 0.18	1.21 ± 0.05	1.86 ± 0.07	1.76 ± 0.11	2.73 ± 0.12	2.36 ± 0.11	3.54 ± 0.19
w/ Focused	1.88 ± 0.14	2.90 ± 0.13	1.24 ± 0.03	1.94 ± 0.05	1.81 ± 0.09	2.83 ± 0.11	2.42 ± 0.10	3.60 ± 0.14
w/ Adversarial	1.92 ± 0.13	3.03 ± 0.14	1.23 ± 0.02	1.93 ± 0.03	1.88 ± 0.06	2.86 ± 0.10	2.45 ± 0.08	3.72 ± 0.11
w/ L2Aug	$\mathbf{2.11 \pm 0.11}$	$\mathbf{3.26 \pm 0.13}$	$\mathbf{1.31 \pm 0.04}$	$\mathbf{1.99 \pm 0.04}$	$\mathbf{1.95 \pm 0.05}$	$\mathbf{3.00 \pm 0.08}$	$\mathbf{2.71 \pm 0.09}$	$\mathbf{3.93 \pm 0.10}$

Table 5.5: Performance on casual user recommendation of various models on different datasets with K=10.

Method	Amazon_CDs		Amazon_Books		Amazon_Movies		Goodreads	
	NDCG@10 (%)	HT@10 (%)	NDCG@10 (%)	HT@10 (%)	NDCG@10 (%)	HT@10 (%)	NDCG@10 (%)	HT@10 (%)
GRU	0.84 ± 0.07	1.74 ± 0.13	0.92 ± 0.03	1.87 ± 0.05	1.56 ± 0.06	3.22 ± 0.11	1.81 ± 0.07	3.61 ± 0.12
w/ Random	0.91 ± 0.12	1.92 ± 0.16	0.99 ± 0.04	2.02 ± 0.03	1.67 ± 0.08	3.34 ± 0.15	1.94 ± 0.10	3.84 ± 0.20
w/ Focused	0.93 ± 0.06	1.91 ± 0.14	1.03 ± 0.03	2.06 ± 0.05	1.77 ± 0.06	3.59 ± 0.13	1.91 ± 0.09	3.89 ± 0.15
w/ Adversarial	0.96 ± 0.04	1.97 ± 0.13	0.99 ± 0.04	2.02 ± 0.04	1.75 ± 0.08	3.51 ± 0.14	1.93 ± 0.08	3.68 ± 0.13
w/ L2Aug	$\mathbf{1.03 \pm 0.03}$	$\mathbf{2.11 \pm 0.10}$	$\mathbf{1.05 \pm 0.04}$	$\mathbf{2.16 \pm 0.03}$	$\mathbf{1.86 \pm 0.05}$	$\mathbf{3.65 \pm 0.10}$	$\mathbf{1.99 \pm 0.07}$	$\mathbf{3.96 \pm 0.11}$
NextItNet	1.59 ± 0.17	2.87 ± 0.20	1.29 ± 0.05	2.55 ± 0.08	1.79 ± 0.07	3.66 ± 0.16	2.52 ± 0.12	4.44 ± 0.18
w/ Random	1.57 ± 0.18	3.10 ± 0.24	1.34 ± 0.07	2.53 ± 0.09	2.05 ± 0.13	4.10 ± 0.18	2.66 ± 0.16	4.74 ± 0.19
w/ Focused	1.45 ± 0.12	2.80 ± 0.19	1.47 ± 0.05	2.74 ± 0.06	1.92 ± 0.11	3.80 ± 0.14	2.69 ± 0.15	4.73 ± 0.17
w/ Adversarial	1.85 ± 0.11	3.44 ± 0.15	1.48 ± 0.04	2.77 ± 0.05	2.11 ± 0.09	4.20 ± 0.11	2.89 ± 0.07	4.90 ± 0.13
w/ L2Aug	$\mathbf{2.11 \pm 0.12}$	$\mathbf{3.95 \pm 0.14}$	$\mathbf{1.55 \pm 0.03}$	$\mathbf{2.86 \pm 0.05}$	$\mathbf{2.22 \pm 0.08}$	$\mathbf{4.32 \pm 0.09}$	$\mathbf{2.98 \pm 0.08}$	$\mathbf{5.10 \pm 0.15}$
SASRec	2.38 ± 0.14	4.49 ± 0.21	1.60 ± 0.07	3.24 ± 0.12	2.26 ± 0.11	4.41 ± 0.19	3.02 ± 0.15	5.77 ± 0.19
w/ Random	2.25 ± 0.15	4.03 ± 0.26	1.61 ± 0.06	3.13 ± 0.14	2.34 ± 0.13	4.48 ± 0.18	3.05 ± 0.15	5.68 ± 0.21
w/ Focused	2.47 ± 0.11	4.73 ± 0.18	1.66 ± 0.04	3.32 ± 0.09	2.45 ± 0.10	4.83 ± 0.14	3.02 ± 0.12	5.47 ± 0.17
w/ Adversarial	2.35 ± 0.16	4.37 ± 0.21	1.62 ± 0.03	3.12 ± 0.10	2.43 ± 0.11	4.54 ± 0.16	3.15 ± 0.10	5.93 ± 0.21
w/ L2Aug	$\mathbf{2.61 \pm 0.12}$	$\mathbf{4.87 \pm 0.19}$	$\mathbf{1.69 \pm 0.03}$	$\mathbf{3.38 \pm 0.08}$	$\mathbf{2.44 \pm 0.08}$	$\mathbf{4.53 \pm 0.11}$	$\mathbf{3.32 \pm 0.11}$	$\mathbf{5.86 \pm 0.15}$

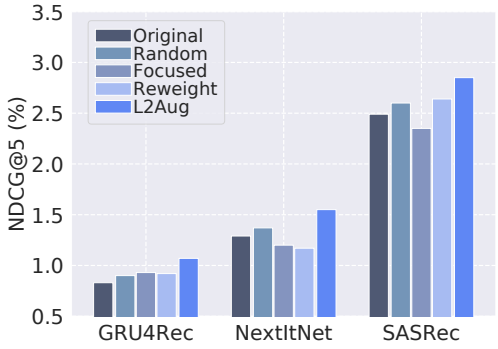

Figure 5.8: Performance on core user recommendation on Amazon_CDs with NDCG.

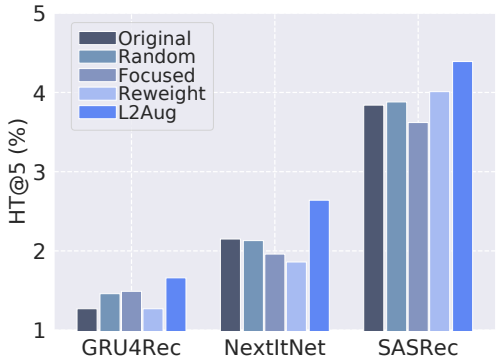

Figure 5.9: Performance on core user recommendation on Amazon_CDs with HT.

Flexibility in Expanding the Action Space. Recall that, so far, the augmentation policy can take two actions: "keep" and "drop". In this section, we examine the feasibility of expanding the action space of L2Aug, which makes the augmentor more versatile. As an initial study, we consider the "substitute" action, which replaces an item with its most correlated item. We adopt the inverse user frequency (i.e., $|N(i) \cap N(j)|/\sqrt{|N(i)||N(j)|}$) [150, 151] to define the correlation between two items (i.e., i and j), in which $N(i)$ is the set of users interacted with item i. Figure 5.10 shows that adding the "substitute" action leads to higher recommendation performance on casual users; it also takes more epochs for the model to converge. Similarly, the proposed L2Aug can

also be extended to support other actions like "reorder" and "insert". Based on the observation, it can be conjectured that the proposed framework is capable of handling various editing actions for sequential data augmentation.

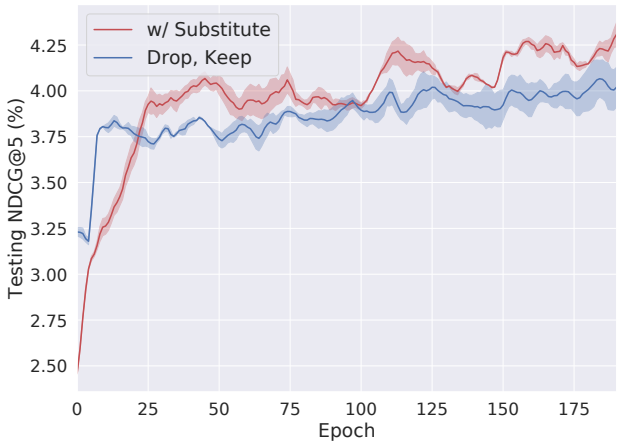

Figure 5.10: L2Aug is easily extended to support more actions (i.e., substitute) with improved casual user recommendation.

Flexibility in Online Setup. The experiments so far are in offline settings, taking the observed user response on recommendations provided by the system as the ground-truth. Offline experiments have the drawback that we are not able to observe user response on counterfactual recommendations, i.e., items that were not shown to the users. To further evaluate the capability of the proposed model for real-world applications, we also conduct online experiments with simulation. We follow [152] to set up the online simulation environment. Given the user's historical interactions and any recommendation candidate, it simulates the user response based on memory matching. This allows us to evaluate models on real-time responses (i.e., ratings) obtained from the simulator instead of relying on offline metrics. In the online experiment, we adopt the public MovieLen 100K dataset and split it into $7:3$ for training and testing, respectively. We treat users of the top 30% visiting frequency as core users and the rest as casual users. In Figure 5.11 and 5.12, DQN [153] is the deep Q-learning method and LIRD [152] is the state-of-the-art for list-wise recommendation. We use

them as the recommenders (i.e., target model) in the L2Aug framework. Combined with L2Aug, both achieve improved performance on casual and core user recommendation under different list sizes, which further corroborates the efficacy of L2Aug under different recommendation scenarios.

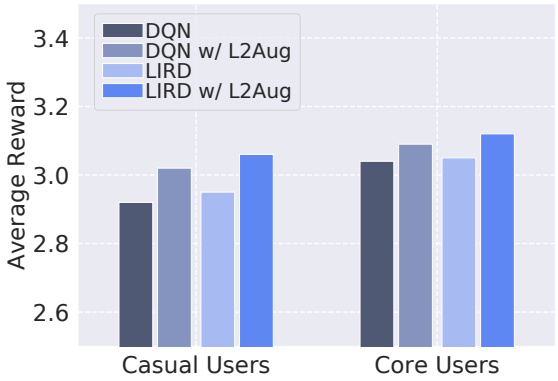

Figure 5.11: *Online* test for List-wise Recommendation with List Size=1.

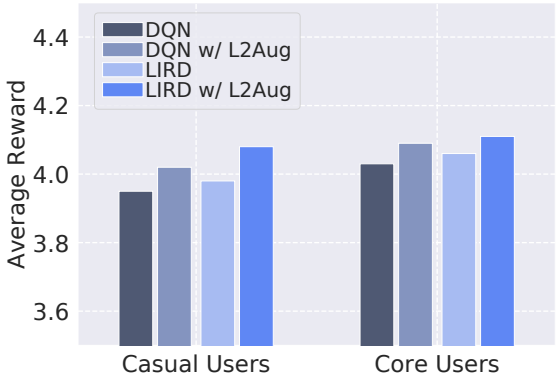

Figure 5.12: *Online* test for List-wise Recommendation with List Size=4.

5.4 Conclusion and Future Work

To enable a recommendation systems to be sustainable and robust to the imperfection in real-world environments, in this section, we firstly propose a novel framework MetaTL to improve

sequential recommendation for cold-start users. To enable the fast adaptation to cold-start users, we reformulate our task as a few-shot learning problem and adopt meta-learning for solving the problem. Powered by a translation-based architecture, the model is able to capture the transition patterns from the transition pairs. Meanwhile, given a pool of few-shot user preference transition tasks that mimic the targeted cold-start scenarios, MetaTL learns a model which can be adapted to new users with just a few interactions in a meta-learning fashion. With experiments on three real-world datasets, the proposed MetaTL can bring in significant improvement compared with the state-of-the-art methods.

Secondly, we focus on the imperfection introduced by casual users, who visit the service occasionally but may leave a few interactions. To bridge the gap between recommendation for core users and casual users, we propose a model-agnostic framework *L2Aug* to learn the data augmentation policy and improve the recommendation system with the generated data. With experiments on four real-world public datasets, the proposed *L2Aug* can outperform other treatment methods and improve casual user recommendation without sacrificing the recommendation for core users. Furthermore, *L2Aug* is flexible in supporting multiple augmentation actions and different experimental (i.e., offline and online) setups.

In the future, we are interested in extending the model to learn the order of multiple transitional pairs. We are also interested in extending the "learning to augment" concept to other application scenarios (e.g., cross-domain, cold-start) for improving the robustness and adaptivity of different recommendation systems. Moreover, we are interested in extending our augmentation policy from the bandit setup studied here to the reinforcement learning setup, where the agent chooses editing actions depending on its previous decisions.

6. CONCLUSION AND FUTURE RESEARCH OPPORTUNITIES

Recommendation systems have become indispensable in everyone's daily life and alleviate the severe information overload issue by helping users find items of interest (e.g. products, services or social content). In this dissertation, we aim to lay the foundations for a new class of *sustainable recommendation systems*. By sustainable, we mean a recommendation system should be fundamentally long-lived, while enhancing both current and future potential to connect users with interesting content. Concretely, we focus on *sustainability* from three perspectives: *adaptivity*, *resilience* and *robustness*. Inspired by efforts to balance static latent factors with temporal dynamics, with efforts to model users over time, and with recent success in Graph Neural Networks and Meta-learning, we make three unique contributions toward sustainable recommendation systems: (i) we develop recommendation models which are highly *adaptive* to the temporal dynamics present in platforms via balancing both the instant shift in user engagement with the lifelong evolution of users and items; (ii) by capturing the influence from the "backbone" – key opinion leaders to the communities, we enable recommendation systems to be *resilient* to dramatic changes including churn in new items and users, and frequently updated connections between users in online communities; and (iii) we explore the problem of training a *robust* recommendation system under imperfect environments and propose a learning-to-learn cold-start user recommendation system and a learning-to-augment framework for improving casual user recommendation.

Though we have seen the great success in recommendation systems for improving the user experience and provide a long-lived foundation for ongoing engagement, there still many challenges that have not been appropriately addressed. With respect to future work, we are quite interesting in the following directions:

- **Privacy-preserving Recommendation Systems.** The social and ethical concerns raised by recommendation systems are increasingly attracting attention. In particular, *privacy leakage* is a critical problem since it can lead to damaging outcomes, e.g., a malicious attacker

can infer the victim's political affiliation or health condition via recommendations on TV shows or medical items, and further abuse the private information for financial benefits [154, 155, 156]. This leakage could be due to factors like: (i) the output of a collaborative filtering-based recommendation systems usually encodes other users' sensitive interactions and profile information; and (ii) the recommendation systems themselves may be untrustworthy and engage in malicious behaviors. To protect users against private-attribute inference attacks, we can potentially frame the problem as an attribute isolation and decoupling problem. In order to prevent the sensitive information from being leaked by the malicious recommendation systems, *federated learning* can be a privacy-guarantee solution since it facilitates distributed collaborative learning while keeping all the training data on (personal) devices.

- **Anomaly Detection Meets Recommendation Systems.** Recommendation systems are highly vulnerable to noise or outliers. For example, there may be fake feedback left by fraudulent users and mistaken interactions made by normal users, and such information may hurt the training process and then result in unreliable recommendations. There are lots of recent efforts on detecting anomalies or outliers from graph-structured data or time-series data [157]. While recommendation systems aim to predict the desired interactions, the deviation of the observed situation from the desired pattern is what many anomaly detection methods rely on to measure the degree of abnormality. Therefore, it would be reasonable to combine the task of training a recommendation system and detecting anomalies (e.g., irregular behaviors or abnormal users), leading to a framework which can accurately detect the anomalies in the platform and generate robust recommendations jointly.

- **Lifelong and Continual Learning Recommendation Systems.** Learning continuously by accumulating the knowledge learned in the past and using the knowledge to help learn more and learn better, remains a key obstacle to achieving human-level intelligence for many ML-based models including recommendation systems. To achieve this goal, lifelong and con-

tinual learning, referred to as the continuous learning ability of an AI algorithm throughout its lifespan, attracts lots of attention recently [158, 159, 160]. With the continuous streams of information along time and across platforms existing in many real-world applications, we are interested in endowing recommendation systems with such a "lifelong" and "continual" learning capacity. Specifically, it can lead to solutions to these research problems: (i) how to apply real time updates (i.e., retrain) on complex neural recommendation models in an online fashion? (ii) how to support transfer learning in recommendation systems without catastrophic forgetting, but also foreknowledge of task similarity? and (iii) how to fine-tune the recommendation models across sessions to emphasize the need in the current session while also remembering the shareable goal?

REFERENCES

[1] D. W. Oard, J. Kim, *et al.*, "Implicit feedback for recommender systems," in *Proceedings of the AAAI workshop on recommender systems*, 1998.

[2] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *CIKM*, 2008.

[3] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets.," in *ICDM*, 2008.

[4] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009.

[5] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *WWW*, 2015.

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

[7] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *SIGIR*, 2019.

[8] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *KDD*, 2018.

[9] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *WSDM*, 2017.

[10] L. R. Flynn, R. E. Goldsmith, and J. K. Eastman, "Opinion leaders and opinion seekers: Two new measurement scales," *Journal of the academy of marketing science*, 1996.

[11] A. Shoham and A. Ruvio, "Opinion leaders and followers: A replication and extension," *Psychology & Marketing*, vol. 25, no. 3, pp. 280–297, 2008.

[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017.

[13] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *WSDM*, 2016.

[14] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, 2010.

[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, 2009.

[16] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010.

[17] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*, 2016.

[18] J. Wang and J. Caverlee, "Recurrent recommendation with local coherence," in *WSDM*, 2019.

[19] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *CIKM*, 2018.

[20] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*, 2018.

[21] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *WSDM*, 2019.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[23] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018.

[24] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," 2019.

[25] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *AAAI*, 2019.

[26] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, "Next-item recommendation with sequential hypergraphs," in *SIGIR*, 2020.

[27] C. Ma, L. Ma, Y. Zhang, J. Sun, X. Liu, and M. Coates, "Memory augmented graph neural networks for sequential recommendation," in *AAAI*, 2020.

[28] P. Jiang, Y. Zhu, Y. Zhang, and Q. Yuan, "Life-stage prediction for product recommendation in e-commerce," in *KDD*, 2015.

[29] K. Ren, J. Qin, Y. Fang, W. Zhang, L. Zheng, W. Bian, G. Zhou, J. Xu, Y. Yu, X. Zhu, *et al.*, "Lifelong sequential modeling with personalized memorization for user response prediction," in *SIGIR*, 2019.

[30] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *RecSys*, 2017.

[31] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *SIGIR*, 2015.

[32] Z. Cheng, J. Shen, L. Zhu, M. S. Kankanhalli, and L. Nie, "Exploiting music play sequence for music recommendation.," in *IJCAI*, 2017.

[33] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *KDD*, 2015.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013.

[35] Y. Koren, "Collaborative filtering with temporal dynamics," in *KDD*, 2009.

[36] A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitskii, "The dynamics of repeat consumption," in *TheWebConf*, 2014.

[37] R. Bhagat, S. Muralidharan, A. Lobzhanidze, and S. Vishwanath, "Buy it again: Modeling repeat purchase recommendations," in *KDD*, 2018.

[38] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke, "Repeatnet: A repeat aware neural recommendation machine for session-based recommendation," in *AAAI*, 2019.

[39] C. Wang, M. Zhang, W. Ma, Y. Liu, and S. Ma, "Modeling item-specific temporal dynamics of repeat consumption for recommender systems," in *TheWebConf*, 2019.

[40] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *RecSys*, 2017.

[41] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention networks," in *IJCAI*, 2018.

[42] H. Li, Y. Ge, D. Lian, and H. Liu, "Learning user's intrinsic and extrinsic interests for point-of-interest recommendation: A unified approach.," in *IJCAI*, 2017.

[43] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, *et al.*, "Item-based collaborative filtering recommendation algorithms.," *WWW*, 2001.

[44] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," *arXiv preprint arXiv:1606.03126*, 2016.

[45] S. Agarwal, K. Branson, and S. Belongie, "Higher order learning with graphs," in *ICML*, 2006.

[46] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *AAAI*, 2019.

[47] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Recurrent coevolutionary latent feature processes for continuous-time recommendation," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2016.

[48] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[49] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[51] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *SDM*, 2019.

[52] K. Ding, Y. Li, J. Li, C. Liu, and H. Liu, "Feature interaction-aware graph neural networks," *arXiv preprint arXiv:1908.07110*, 2019.

[53] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *WWW*, 2019.

[54] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," *arXiv preprint arXiv:1904.10322*, 2019.

[55] J. Wang, K. Ding, Z. Zhu, Y. Zhang, and J. Caverlee, "Key opinion leaders in recommendation systems: Opinion elicitation and diffusion," in *WSDM*, 2020.

[56] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[57] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, "Temporal recommendation on graphs via long-and short-term preference fusion," in *KDD*, 2010.

[58] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *WSDM*, 2019.

[59] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He, "Music recommendation by unified hypergraph: combining social media information and music content," in *MM*, 2010.

[60] L. Li and T. Li, "News recommendation via hypergraph learning: encapsulation of user behavior and news content," in *WSDM*, 2013.

[61] Y. Zhu, Z. Guan, S. Tan, H. Liu, D. Cai, and X. He, "Heterogeneous hypergraph embedding for document recommendation," *Neurocomputing*, vol. 216, pp. 150–162, 2016.

[62] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach," in *WWW*, 2019.

[63] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *arXiv preprint arXiv:1901.08150*, 2019.

[64] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *NeurIPS*, 2016.

[65] C. Ma, P. Kang, and X. Liu, "Hierarchical gating networks for sequential recommendation," in *KDD*, 2019.

[66] M. De Veirman, V. Cauberghe, and L. Hudders, "Marketing through instagram influencers: the impact of number of followers and product divergence on brand attitude," *International Journal of Advertising*, 2017.

[67] L. V. Casaló, C. Flavián, and S. Ibáñez-Sánchez, "Influencers on instagram: Antecedents and consequences of opinion leadership," *Journal of Business Research*, 2018.

[68] Y. Zhang and J. Caverlee, "Instagrammers, fashionistas, and me: Recurrent fashion recommendation with implicit visual influence," in *CIKM*, 2019.

[69] T. Tucker, "Online word of mouth: characteristics of yelp. com reviews," *Elon Journal of Undergraduate Research in Communications*, vol. 2, no. 1, pp. 37–42, 2011.

[70] L. Zhu, G. Yin, and W. He, "Is this opinion leader's review useful? peripheral cues for online review helpfulness," *Journal of Electronic Commerce Research*, vol. 15, no. 4, p. 267, 2014.

[71] R. Neves-Silva, M. Gamito, P. Pina, and A. R. Campos, "Modelling influence and reach in sentiment analysis," *Procedia CIRP*, vol. 47, pp. 48–53, 2016.

[72] W.-T. Hsieh, T. Ku, C.-M. Wu, and S.-c. T. Chou, "Social event radar: a bilingual context mining and sentiment analysis summarization system," in *Proceedings of the ACL 2012 System Demonstrations*, pp. 163–168, Association for Computational Linguistics, 2012.

[73] J.-H. Yang, C.-M. Chen, C.-J. Wang, and M.-F. Tsai, "Hop-rec: high-order proximity for implicit recommendation," in *RecSys*, 2018.

[74] M. Thelwall and K. Kousha, "Goodreads: A social network site for book readers," *JASIST*, no. 4, pp. 972–983, 2017.

[75] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts, "Who says what to whom on twitter," in *WWW*, 2011.

[76] Y. Yang, J. Tang, C. W.-k. Leung, Y. Sun, Q. Chen, J. Li, and Q. Yang, "Rain: Social role-aware information diffusion.," in *AAAI*, 2015.

[77] R. He and J. McAuley, "Vbpr: visual bayesian personalized ranking from implicit feedback," in *AAAI*, 2016.

[78] W. Niu, J. Caverlee, and H. Lu, "Neural personalized ranking for image recommendation," in *WSDM*, 2018.

[79] G. Karamanolakis, K. R. Cherian, A. R. Narayan, J. Yuan, D. Tang, and T. Jebara, "Item recommendation with variational autoencoders and heterogeneous priors," in *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, pp. 10–14, 2018.

[80] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *KDD*, 2016.

[81] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *NeurIPS*, 2013.

[82] Y. Cao, X. Wang, X. He, Z. Hu, and C. Tat-seng, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preference," in *WWW*, 2019.

[83] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *WWW*, 2019.

[84] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NeurIPS*, 2013.

[85] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, 2014.

[86] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015.

[87] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL*, 2015.

[88] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *SIGIR*, 2017.

[89] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang, "Robust graph neural network against poisoning attacks via transfer learning," in *arXiv preprint arXiv:1908.07558*, 2019.

[90] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Node injection attacks on graphs via reinforcement learning," in *arXiv preprint arXiv:1909.06543*, 2019.

[91] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[92] Z. Zhu, J. Wang, and J. Caverlee, "Improving top-k recommendation via joint collaborative autoencoders," in *WWW*, 2019.

[93] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," in *WSDM*, 2012.

[94] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[95] J. Yin, C. Liu, W. Wang, J. Sun, and S. C. Hoi, "Learning transferrable parameters for long-tailed sequential user behavior modeling," in *KDD*, 2020.

[96] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *KDD*, 2019.

[97] J. Li, M. Jing, K. Lu, L. Zhu, Y. Yang, and Z. Huang, "From zero-shot learning to cold-start recommendation," in *AAAI*, 2019.

[98] Z. Zhu, S. Sefati, P. Saadatpanah, and J. Caverlee, "Recommendation for new users and new items via randomized training and mixture-of-experts transformation," in *SIGIR*, 2020.

[99] N. Mirbakhsh and C. X. Ling, "Improving top-n recommendation for cold-start users via cross-domain information," in *TKDD*, 2015.

[100] S. Kang, J. Hwang, D. Lee, and H. Yu, "Semi-supervised learning for cross-domain recommendation to cold-start users," in *CIKM*, 2019.

[101] Y. Bi, L. Song, M. Yao, Z. Wu, J. Wang, and J. Xiao, "Dcdir: A deep cross-domain recommendation system for cold start users in insurance domain," in *SIGIR*, 2020.

[102] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *NeurIPS*, 2016.

[103] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017.

[104] K. Ding, J. Wang, J. Li, K. Shu, C. Liu, and H. Liu, "Graph prototypical networks for few-shot learning on attributed networks," in *CIKM*, 2020.

[105] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, 2015.

[106] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016.

[107] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," in *arXiv preprint arXiv:1803.02999*, 2018.

[108] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[109] K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot network anomaly detection via cross-network meta-learning," in *The Web Conference*, 2021.

[110] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in *ICDM*, 2010.

[111] M. Volkovs, G. Yu, and T. Poutanen, "Dropoutnet: Addressing cold start in recommender systems," in *NeurIPS*, 2017.

[112] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *NeurIPS*, 2017.

[113] T. Wei, Z. Wu, R. Li, Z. Hu, F. Feng, X. He, Y. Sun, and W. Wang, "Fast adaptation for cold-start collaborative filtering with meta-learning," ICDM, 2020.

[114] F. Pan, S. Li, X. Ao, P. Tang, and Q. He, "Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings," in *SIGIR*, 2019.

[115] M. Chen, W. Zhang, W. Zhang, Q. Chen, and H. Chen, "Meta relational learning for few-shot link prediction in knowledge graphs," in *EMNLP*, 2019.

[116] F. Buttle, *Customer relationship management*. Routledge, 2004.

[117] M. Chen, Y. Wang, C. Xu, Y. Le, M. Sharma, L. Richardson, S.-L. Wu, and E. Chi, "Values of user exploration in recommender systems," in *RecSys*, 2021.

[118] Y. Zheng, S. Liu, Z. Li, and S. Wu, "Cold-start sequential recommendation via meta learner," in *AAAI*, 2020.

[119] F. Zhu, Y. Wang, C. Chen, J. Zhou, L. Li, and G. Liu, "Cross-domain recommendation: challenges, progress, and prospects," in *IJCAI*, 2021.

[120] M. M. Khan, R. Ibrahim, and I. Ghani, "Cross domain recommender systems: a systematic literature review," in *CSUR*, 2017.

[121] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," in *JMLR*, 2016.

[122] C. Zhao, C. Li, R. Xiao, H. Deng, and A. Sun, "Catn: Cross-domain recommendation for cold-start users via aspect transfer network," in *SIGIR*, 2020.

[123] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, 2017.

[124] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *CVPR*, 2019.

[125] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[126] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Interspeech*, 2015.

[127] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," *EMNLP*, 2019.

[128] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, PMLR, 2020.

[129] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.

[130] A. Beutel, E. H. Chi, Z. Cheng, H. Pham, and J. Anderson, "Beyond globally optimal: Focused learning for improved recommendations," in *TheWebConf*, 2017.

[131] M. Luo, F. Chen, P. Cheng, Z. Dong, X. He, J. Feng, and Z. Li, "Metaselector: Meta-learning for recommendation with user-level adaptive model selection," in *TheWebConf*, 2020.

[132] M. Ekstrand and J. Riedl, "When recommenders fail: predicting recommender failure for algorithm selection and combination," in *RecSys*, 2012.

[133] B. Wang, M. Qiu, X. Wang, Y. Li, Y. Gong, X. Zeng, J. Huang, B. Zheng, D. Cai, and J. Zhou, "A minimax game for instance based selective transfer learning," in *KDD*, 2019.

[134] M. D. Ekstrand, A. Chaney, P. Castells, R. Burke, D. Rohde, and M. Slokom, "Simurec: Workshop on synthetic data and simulation methods for recommender systems research," in *RecSys*, 2021.

[135] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," *arXiv preprint arXiv:2002.12478*, 2020.

[136] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep neural network acoustic modeling," in *TASLP*, 2015.

[137] S. Kobayashi, "Contextual augmentation: Data augmentation by words with paradigmatic relations," in *NAACL-HLT*, 2018.

[138] M. Gao, J. Zhang, J. Yu, J. Li, J. Wen, and Q. Xiong, "Recommender systems based on generative adversarial networks: A problem-driven perspective," *Information Sciences*, 2021.

[139] D.-K. Chae, J.-S. Kang, S.-W. Kim, and J. Choi, "Rating augmentation with generative adversarial networks towards accurate collaborative filtering," in *TheWebConf*, 2019.

[140] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[141] D. Croce, G. Castellucci, and R. Basili, "Gan-bert: Generative adversarial learning for robust text classification with a bunch of labeled examples," in *ACL*, 2020.

[142] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, B. Ding, and B. Cui, "Contrastive learning for sequential recommendation," 2021.

[143] Z. Yao, J. R. Peddamail, and H. Sun, "Coacor: Code annotation for code retrieval with reinforcement learning," in *TheWebConf*, 2019.

[144] K. Zhang, C. Xiong, Z. Liu, and Z. Liu, "Selective weak supervision for neural information retrieval," in *TheWebConf*, 2020.

[145] K. Ding, D. Li, A. H. Li, X. Fan, C. Guo, Y. Liu, and H. Liu, "Learning to selectively learn for weakly-supervised paraphrase generation," in *EMNLP*, 2021.

[146] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[147] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," *arXiv preprint arXiv:1811.11682*, 2018.

[148] M. Wan and J. McAuley, "Item recommendation on monotonic behavior chains," in *RecSys*, 2018.

[149] P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. H. Chi, "Fairness without demographics through adversarially reweighted learning," in *NeurIPS*, 2020.

[150] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. mcgraw-hill, 1983.

[151] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *arXiv preprint arXiv:1301.7363*, 2013.

[152] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for list-wise recommendations," 2019.

[153] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[154] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *TOIT*, 2007.

[155] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *CCS*, 2013.

[156] X. Meng, S. Wang, K. Shu, J. Li, B. Chen, H. Liu, and Y. Zhang, "Personalized privacy-preserving social recommendation," in *AAAI*, 2018.

[157] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," in *CSUR*, 2021.

[158] B. Liu, "Learning on the job: Online lifelong and continual learning," in *AAAI*, 2020.

[159] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[160] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.