Temporal Graph Learning for Dynamic Recommendation

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Technology

by

Haoran Tang

to

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW, Australia

May 2025

Certificate of Original Authorship

I, *Haoran Tang*, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science, Faculty of Engineering and Information Technology* at the University of Technology Sydney. This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with The Hong Kong Polytechnic University (HKPolyU). This research was supported by an Australian Government Research Training Program (RTP) Scholarship `doi.org/10.82133/C42F-K220`.

Production Note:

SIGNATURE: Signature removed prior to publication.

Haoran Tang

DATE: 10th May, 2025

PLACE: Sydney, Australia

ABSTRACT

Dynamic recommendation, where users continuously interact with items over time, has been widely deployed in real-world applications, such as social networks, streaming media, and online shopping platforms. The burst of user-item interactions causes rapid evolution of both user representations and item representations. Compared with the conventional static recommendation that follows the one-time recommendation-evaluation paradigm, dynamic recommendation involves dynamically updating temporal representations to refresh recommendation results in a real-time manner, leading to a dynamic multi-round recommendation process. Temporal Graph Learning, which constructs temporal interaction graphs by their sequential information, has demonstrated its superiority in modeling temporal representations in the dynamic environment. Leveraging temporal graph learning makes it much easier to explore the evolving preferences of users and generate continuous recommendations for the dynamic recommendation scenario. Thus, this research focuses on fully investigating the power of temporal graph learning for dynamic recommendation.

However, designing comprehensive temporal update for user-item interaction graph still remains a challenge, since static graph learning for conventional recommendation has dominated the field of recommendation systems and also existing studies on temporal graph learning are mainly centered on some single insufficient perspectives for update. Moreover, they directly conduct representation update without scrutinizing whether the interactions would truly benefit temporal graph learning, which potentially harms the performance of recommendation. Thus, uncovering the effect of both newly-arrived interactions and historical interactions poses another challenge to temporal update. In addition to the inherent challenges above, the unfairness issue among item exposures and user experiences tends to be amplified due to the evolving context in the dynamic environment. Unfortunately, most fairness methods are designed for the conventional static recommendation.

To address the mentioned challenges, this research first proposes a novel framework that comprehensively and efficiently captures user and item dynamics over time from the perspectives of inherent interaction potential, time-decay augmentation, and symbiotic local structure learning, thus fully modeling the dynamic graph evolution for recommendation. Second, this research presents a novel temporal collaboration-aware graph network by investigating the collaborative effectiveness of the newly-arrived interactions, to guide the graph evolution learning process. Third, this research constructs a novel self-correcting interaction network by dynamically filtering out the inappropriate historical interactions, to ensure the accuracy of temporal graph aggregation. In addition, this research further studies the problem of dual-side unfairness issue for the dynamic recommendation and presents an adaptive model-agnostic post-ranking method to mitigate the dynamic unfairness in temporal graph models.

Extensive experiments over real-world datasets demonstrate both the effectiveness and superiority of this research, shedding light on the field of recommendation systems. This research successfully explores different aspects of the temporal graph learning for the dynamic recommendation, including temporal update, interaction scrutiny, and dynamic fairness. Hence, future work would like to explore the interpretability in dynamic recommendation. One feasible direction would be leveraging the large language models to incorporate textual information with temporal graph information and then generate interpretable recommendations for users. This could further strengthen the applicability of temporal graph learning in the dynamic recommendation and enhance the extensibility of this research.

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Prof. Guandong Xu (UTS) and Prof. Qing Li (HKPolyU), for their support, help, guidance, and patience throughout my whole PhD journey. The time that I spent working with them has been one of the most enriching and unforgettable experiences in my entire life. They have taught me to face the world with wisdom and courage as a researcher. I am sincerely grateful for the Collaborative Doctoral Research Degree program between UTS and HKPolyU, which has enabled me to earn dual doctoral degrees.

I am cordially thankful to all my colleagues and friends at UTS and HKPolyU. They have provided many inspirations and suggestions for my research, and I have learned a lot from them. Moreover, I am deeply grateful to the co-authors of my papers, who have helped me overcome the difficulties in my research.

I am profoundly grateful to my family, especially my parents. The unconditional love they have given to me encourages and supports me in every dark moment. They never ask for anything in return. I am the luckiest person to have my parents always by my side. No words could fully express my heartfelt gratitude for them.

Finally, I would like to say a special thanks to myself, for still moving forward when the journey grew steep, for turning obstacles into growth sources, and for never letting go of hope when progress seemed invisible. Thank you for being here after a long PhD journey, and your life journey will continue.

LIST OF PUBLICATIONS

RELATED TO THE THESIS

1. **H. Tang**, S. Wu, G. Xu, and Q. Li.
   "Dynamic Graph Evolution Learning for Recommendation," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '23, ACM, 2023, pp. 1589–1598.

2. **H. Tang**, S. Wu, X. Sun, J. Zeng, G. Xu, and Q. Li.
   "TCGC: Temporal Collaboration-Aware Graph Co-Evolution Learning for Dynamic Recommendation," in *ACM Transactions on Information Systems (TOIS)*, vol. 43, no. 1, pp. 1–27, 2024.

3. **H. Tang**, S. Wu, Z. Cui, Y. Li, G. Xu, and Q. Li.
   "Model-agnostic Dual-side Online Fairness Learning for Dynamic Recommendation," in *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 37, no. 5, pp. 2727-2742, 2025.

4. **H. Tang**, P. Wu, S. Wu, G. Xu, and Q. Li.
   "Dynamic Self-correcting Interaction Network," under review at *ACM Transactions on Information Systems (TOIS)*, 2025.

5. **H. Tang**, X. Sun, S. Wu, Z. Cui, G. Xu, and Q. Li.
   "DyBooster: Leveraging Large Language Model as Booster for Dynamic Recommendation," in *Expert Systems with Applications (ESWA)*, vol 286, p.128080, 2025.

Others

1. Y. Li, Y. Yang, J. Cao, S. Liu, **H. Tang**, and G. Xu.
   "Toward Structure Fairness in Dynamic Graph Embedding: A Trend-aware Dual Debiasing Approach," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD'24, ACM, 2024, pp. 1701–1712.

2. J. Zeng, H. Tao, **H. Tang**, J. Wen, and M. Gao.
   "Global and Local Hypergraph Learning Method with Semantic Enhancement for POI Recommendation," in *Information Processing & Management*, vol. 62, no. 1, p. 103868, 2025.

3. K. Yi, **H. Tang**, H. Bai, Y. Wang, J. Xu, and P. Li.
   "Bi-directional Interaction and Dense Aggregation Network for RGB-D Salient Object Detection," in *Proceedings of the 30th International Conference on Multimedia Modeling*, Springer, 2024, pp. 475–489.

4. X. Sun, K. Shi, **H. Tang**, G. Xu, and Q. Li.
   "Expert-Guided Toxicity Filtration for Debiased Generation," in *Proceedings of the 29th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. PAKDD '25, pp. 115–127, 2025.

5. K. Yi, Y. Li, **H. Tang**, and J. Xu.
   "Adaptive Depth Enhancement Network for RGB-D Salient Object Detection," in *IEEE Signal Processing Letters*, vol. 32, pp. 176–180, 2025.

6. Y. Pan, J. Zeng, Z. Wang, **H. Tang**, J. Wen, and M. Gao.
   "HGDRec: Next POI Recommendation Based on Hypergraph Neural Network and Diffusion Model," in *IEEE Transactions on Services Computing (TSC)*, vol. 18, no. 3, pp. 1445-1458, 2025.

7. X. Sun, K. Shi, **H. Tang**, D. Wang, G. Xu, and Q. Li.
   "Educating Language Models as Promoters: Multi-Aspect Instruction Alignment with Self-Augmentation," in *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 37, no. 8, pp. 4564-4577, 2025.

8. Y. Li, J. Zeng, **H. Tang**, J. Wen, M. Gao, and W. Zhou.
   "DMSDRec: Dynamic Structure-aware Graph Masked Autoencoder and Spatiotemporal Diffusion for Next-POI Recommendation," in *IEEE Transactions on Services Computing (TSC)*, vol. 18, no. 4, pp. 2024-2037, 2025.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **Page**

LIST OF TABLES

TABLE **Page**

Chapter 1

Introduction

This chapter presents the introduction to the thesis. The introduction contains four parts: research background, existing limitations, research challenges, contributions, and thesis structure. Specifically:

- Sec. 1.1 introduces the background of this research by converting conventional recommendation to dynamic recommendation with temporal graph learning.
- Sec. 1.2 discusses existing limitations in temporal graph learning for dynamic recommendation and derives corresponding challenges that motivate this research.
- Sec. 1.3 proposes the key research challenges to indicate how this research overcomes existing limitations and advances the dynamic recommendation.
- Sec. 1.4 briefly summarizes the contributions of this research by concluding the new proposed solutions and models from this research.
- Sec. 1.5 organizes the chapters of the thesis and shows its structure.

The introduction chapter lays the foundation for this thesis by providing a comprehensive overview of the research background, limitations, challenges, and contributions. The subsequent sections will delve into the detailed contents.

1.1 Research Background

Recommendation systems (RecSys), which help users discover their preferred items, are likely to alleviate the information overload on the web, such as e-commerce (e.g., Amazon and Taobao), video sites (e.g., TikTok and YouTube), and social networks (e.g., Facebook and Instagram) [18, 6, 69, 143]. The key to a personalized recommendation system is in learning interactions between users and items and then modeling user preferences over items. Early studies, like deep learning based Collaborative Filtering (CF) and Matrix Factorization (MF), exploit latent feature information to match users and items in the embedding space [33, 111]. More recently, the Graph Neural Network (GNN) has delivered superior performance to recommendation systems [98, 32, 105, 21], because of its powerful capability of exploiting high-order connectivity among users and items via user-item interaction graph. As a result, GNN has become one of the most state-of-the-art (SOTA) technologies for the recommendation systems.

Most GNN-based recommendation models, however, are centered on the conventional recommendation paradigm that conducts a one-time recommendation list and a one-time performance evaluation for each user. In other words, each user would get fixed recommendation results during the inferring stage, while in real-world platforms, recommendation systems should dynamically update the recommendation results and continuously generate different recommendation lists to meet with the user's evolving preference over time [88, 101, 44]. This can be described as the dynamic multi-round recommendation. For the dynamic recommendation, the burst of user-item interactions rapidly changes the interactive pattern between users and items [16, 40], leading to fast drift in the status of the recommendation system. Hence, exploiting the evolving preferences and behavior patterns hidden in the continuous user-item interactions is crucial for generating accurate recommendations.

Temporal Graph Learning (TGL), which sequentially constructs interaction graphs by time information, has demonstrated its superiority in modeling the representation evolution in dynamic environments [101, 67, 85, 132]. It endows the static GNN with the capability of perceiving temporal change among different timestamps, thus creating the Temporal Graph Neural Network (T-GNN). Leveraging the T-GNN, the dynamic recommendation can not only explore the high-order connectivity among users and items but also capture the temporal evolution of users and items [102, 96], making it possible to deliver a further improvement to recommendation performance. Despite the potential of T-GNN, there are still some limitations that remain to be solved, such as the insufficient temporal update in

T-GNN, the unverified effectiveness of newly-arrived interaction, the uncertain influence of historical interactions, and the evolving unfairness of recommendation. They pose new challenges to the temporal graph learning for the dynamic recommendation, which motivates the research of this thesis.

Therefore, this thesis focuses on addressing existing limitations of TGL and developing more advanced T-GNN models and solutions, to conduct better dynamic recommendation systems that are aligned with the real-world scenarios. It is believed that the research on temporal graph learning for dynamic recommendation would have a significant impact on the field of recommendation systems and would connect closely to the practical and reliable information systems in the real world. Following the research background, the next section comprehensively discusses the existing limitations of TGL.

1.2 Existing Limitations

As introduced in the research background, temporal graph learning can learn the evolving representations of users and items to improve dynamic recommendation. However, most temporal GNN-based recommendation models mainly utilize insufficient perspectives for representation update, without comprehensively capturing different dynamics from different perspectives. In other words, they prefer designing single and restricted graph aggregation methods to derive temporal representations, which cannot fully reflect the dynamics of users and items. The dynamic change among different timestamps naturally exists in interactions, historical neighbors, and structures. They build and affect each other over time, thus causing multiple perspectives for representation update. Unfortunately, most studies cannot efficiently utilize them in one unified framework. This is the first limitation (**EL 1**) that should be solved in the first place. Unless EL 1 is solved, research into other aspects, such as interaction effectiveness and model extensibility, would be unproductive, since the accurate update cannot be promised.

Second, in the TGL, the incoming interaction will usually trigger the T-GNN in a real-time manner. Thus, the collaboration between the new interaction and historical interactions will jointly generate the temporal representations for the currently triggered nodes [96]. Existing studies, however, tend to directly model the temporal evolution when the new interaction is injected into the graph, without carefully verifying whether the collaborative effect learned from the new interaction would truly benefit modeling the evolution, or how much useful information could be learned at the current timestamp. This is the second limitation (**EL 2**), which also exists in the static GNN studies [9, 21]. This limitation brings the risk of learning uninformative or even harmful information due to the ambiguous understanding of the new interaction. Moreover, dynamic recommendation systems are more sensitive to the stream of interactions. Blindly learning the evolution of users and items cannot reflect actual behavior patterns and may destroy the user experience during the recommendation procedure.

Third, except for the newly-arrived interaction, the historical interactions, which carry the long-term preferences of users, are also the key factor in updating the temporal representations. Most previous studies aggregate the historical neighbors in the graph to pass the high-order information. However, they explicitly assume that all historical interactions would reflect the actual representation evolution of users and items. This means they do not verify whether those historical interactions could benefit temporal representation learning. As a matter of fact, some user-item interactions may be imperfect since they cannot indicate the users' actual preferences or items' intrinsic attributes [52, 92]. This is the third limitation (**EL 3**), which is a little similar to EL 2 but focuses on the historical interactions. In some cases, the historical interactions could be the occasional results, like the popular item recommendations from the homepage. It is inappropriate to leverage them representing the temporal evolution of user preferences. Unfortunately, most T-GNN models are not able to distinguish them, leaving room for improvement.

Fourth, there has been a growing awareness that fairness is important to build healthy relationships between users and between items because it significantly affects how items are exposed to users and how users access items [115, 60]. For the dynamic recommendation systems, the popular items would get more exposure and the active users would get better recommendation services, due to the continuous interactions over time. Most temporal graph-based recommendation methods lack the consideration of mitigating the unfairness, causing bias to the unpopular items and the inactive users. This is the fourth limitation (**EL 4**). One of the core reasons is that existing fairness-aware solutions still follow the conventional one-time recommendation-evaluation paradigm. They cannot be smoothly applied to the TGL for the dynamic recommendation systems. This is, however, the limitation of the conventional recommendation systems, which is relevant to but obviously more than the EL 4.

The above-mentioned **EL 1**, **EL 2**, **EL 3** and **EL 4** limit the improvement and development of temporal graph learning for the dynamic recommendation, providing this thesis the opportunity to conduct further study. EL 1 is oriented at the temporal update problem of T-GNN. EL 2 is about the effectiveness of newly-arrived interaction while EL 3 is about the influence of historical interactions. Solving them would improve the T-GNN on the basis of EL 1. Thus, EL 1 should be overcome first. Last, EL 4 is oriented at the extensibility of dynamic recommendation. A brief summarization that involves the limitations, challenges, and contributions, is shown in Figure 1.1. Based on existing limitations, the next section proposes the key research challenges.

1.3 Research Challenges

To solve existing limitations, this research should identify the core challenges that provide a clear roadmap for research. For EL 1, the key point is compared with the static GNN, the temporal GNN can naturally capture the changes among timestamps of different perspectives, such as interaction, historical neighbors, and structures. By continuous graph convolution on them, the dynamic representation evolution of both users and items can be extracted through the temporal user-item interaction graph. Hence, the first research challenge (**RC 1**) would be:

RC 1: ***How to comprehensively leverage different update perspectives in one unified framework and efficiently convert static graph learning to temporal graph learning for recommendation?***

For EL 2, the key to measuring the effectiveness of newly-arrived interaction is to investigate the collaboration between the incoming interaction and historical interactions since the incoming interaction would trigger the update at the current timestamp. Such investigation could further rebuild the design of T-GNN to understand the correlation between newly-arrived interactions and historical interactions and then conduct a more accurate update of temporal representations. Therefore, the second research challenge (**RC 2**) would be:

RC 2: ***How well the newly-arrived interaction could match the temporal history and how much information should be learned according to the current context?***

For EL 3, the key point lies in the relevance of the historical interactions over user actual preferences. In dynamic environment, the growth of historical interactions would derive the long-term preferences of users, reshaping their temporal representations. Compared with newly-arrived interaction, historical interactions tend to have more impact due to the large number of them. It is very necessary to select useful historical interactions for update and prevent the model from learning incorrect representations. Hence, the third research challenge (**RC 3**) would be:

RC 3: ***How to efficiently discover valuable interactions from long-term history and adaptively filter out those inappropriate interactions for improving update?***

For EL 4, the key to mitigating the unfairness for dynamic recommendation is to endow fairness with the capability of perceiving the temporal information at different timestamps. There are different T-GNN models, but the fairness is a high-dimension problem more than model design. Constructing different exclusive fairness methods for every T-GNN model is time-consuming and also non-generalizable. Moreover, the item side exhibits the exposure disparity among items while the user side exhibits performance discrepancy among users. Thus, the fourth research challenge (**RC 4**) would be:

RC 4: ***How to dynamically trace the dual-side unfairness over time and efficiently mitigate unfairness in a generalizable way without being restricted by models?***

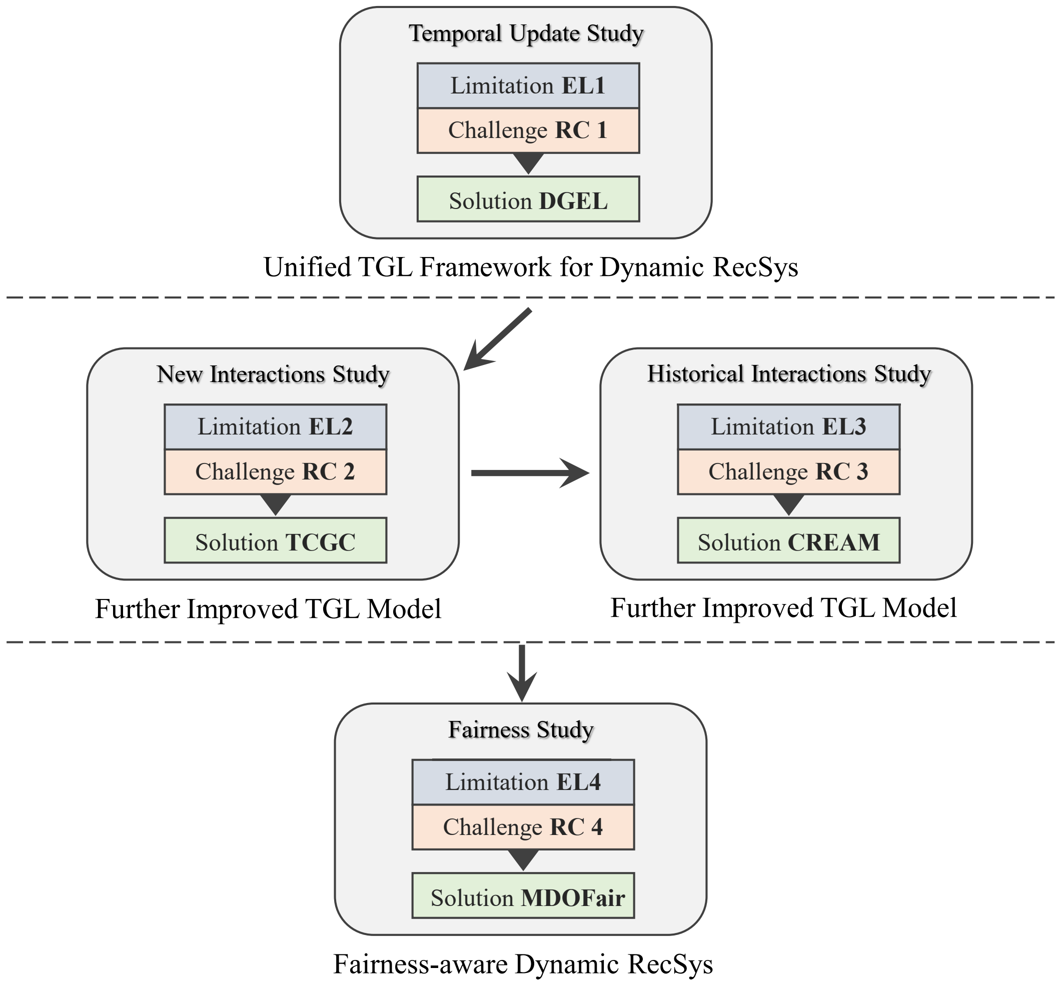


Figure 1.1: The roadmap of this research on TGL for dynamic RecSys.

The research challenges RC 1, RC 2, RC 3, and RC 4, build the roadmap of the research in this thesis. The thesis will follow the roadmap to conduct the research and present the outcome. The roadmap is illustrated in Figure. 1.1, with the limitations and contributions.

This research aims to address RC 1 and obtain a unified T-GNN framework that fully explores different perspectives for temporal update. Next, on the basis of the outcome of RC 1, this research aims to address RC 2 and get a collaboration-aware T-GNN model that efficiently investigates the effectiveness of newly-arrived interaction. Meanwhile, this research also aims to address RC 3 and get a dynamic self-correcting graph network that adaptively filters out the historical interactions. After RC 1, RC 2, and RC 3, this research further aims to address RC 4 and propose a generalizable fairness learning method that successfully mitigates the unfairness for different models. The next section summarizes the contributions of this research.

1.4 Contributions

Following the research challenges outlined in the previous section, this research presents four studies that aim to address them and deliver improvement to the dynamic recommendation systems. The four studies are presented in Chapter 3, Chapter 4, Chapter 5, and Chapter 6, and their contributions can be summarized:

- To address RC 1, Chapter 3 proposes a novel framework named Dynamic Graph Evolution for Recommendation (**DGEL**) The DGEL could comprehensively and efficiently capture user and item dynamics over time from the perspectives of inherent interaction potential, time-decay augmentation, and symbiotic local structure learning. Meanwhile, DGEL performs the initial attempts to adaptively bridge the normalization of dynamic embeddings with the model learning process for the dynamic recommendation. Moreover, DGEL is designed to update the user-item interaction graph in a real-time manner, thus successfully refreshing the recommendation results over time and converting static one-time recommendation into dynamic multi-round recommendation.
- To address RC 2, Chapter 4 investigates the temporal collaborative effect existing in the evolution learning of temporal graphs for dynamic recommendation. It first proposes the temporal collaboration-aware indicators to measure the newly-arrived interactions and guide the learning process for the temporal graph. Then, to capture the inherent correlation between event-level and history-level dynamics, it presents a novel graph network named Temporal Collaboration-Aware Graph Co-Evolution Learning (**TCGC**), which leverages the proposed collaboration-aware indicators to generate accurate temporal representations.

- To address RC 3, Chapter 5 proposes a novel dynamic self-correcting interaction network (**CREAM**), which aims to accurately and continuously generate predictions over the temporal user-item interaction graph. Along this line, CREAM builds the adaptive dual-side corrector to dynamically correct the inappropriate information from historical interactions, and the temporal excitement learning to capture the effects of the newly-arrived interactions. Meanwhile, CREAM presents a warm initialization that could flexibly initialize both temporal and static representations. Last, CREAM utilizes both temporal and static representations to precisely conduct predictions on future interactions.

- To address RC 4, Chapter 6 studies the problem of dual-side unfairness for the dynamic recommendation where there exists a rapid evolution of unfairness in the dynamic online environment. It proposes an adaptive model-agnostic post-ranking method with dual-side online fairness learning (**MDOFair**), which involves dynamic item-side exposure-oriented unfairness and user-side quality-oriented unfairness. MDOFair traces the online fairness for both users and items over time, and moreover, it leverages the online fairness task and dynamic recommendation task in one unified framework, to achieve a double-win success for both unfairness mitigation and recommendation improvement.

This research also conducts comprehensive and extensive experiments over real-world datasets to evaluate all the proposed solutions and models, assessing their performance for dynamic recommendation. The experiments vary across multiple dimensions, including overall comparison, ablation study, variation exploration, hyperparameter study, efficiency comparison, user case study, and so on. The experimental results demonstrate both the superiority and effectiveness of the proposed solutions and models and also offer a thorough understanding of the application of TGL for dynamic recommendation.

1.5 Thesis Structure

The structure of this thesis is illustrated in Figure 1.2. Specifically, Chapter 1 serves as the introduction of this thesis, providing an overview of the background, limitations, challenges, and contributions. Chapter 2 reviews the related literature of this research, revealing the strengths and weaknesses of existing methods. Next, Chapter 3 explores dynamic graph evolution from multiple perspectives for recommendation, addressing RC 1. Furthermore, Chapter 4 presents the temporal collaboration-aware graph network for recommendation, addressing RC 2, while Chapter 5 presents the temporal self-correction interaction network for recommendation, addressing RC 3. Then, Chapter 6 studies the dual-side online fairness in dynamic environment, addressing RC 4. Last, Chapter 7 concludes the whole thesis and discusses future work.

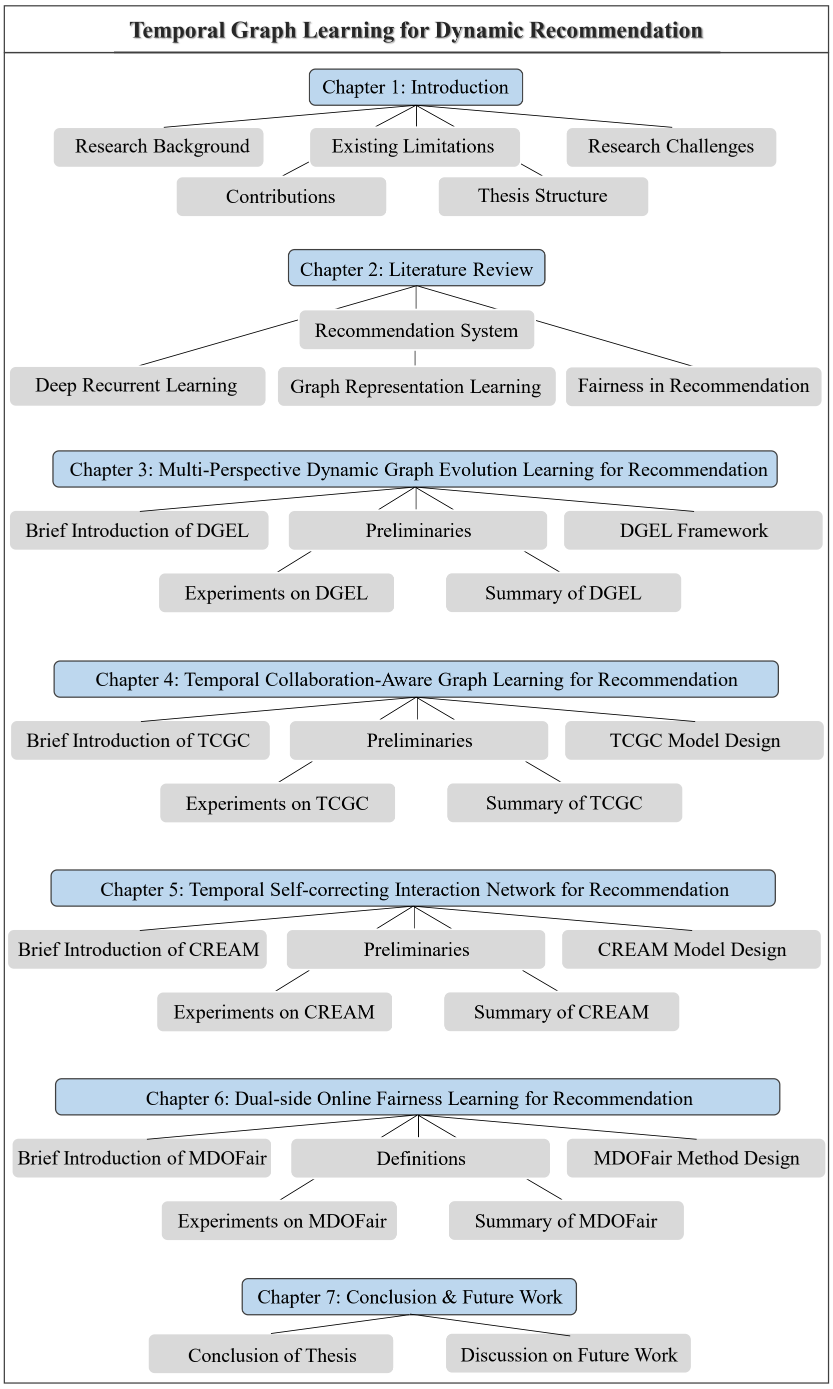

Temporal Graph Learning for Dynamic Recommendation
Chapter 1: Introduction
Research Background
Existing Limitations
Research Challenges
Contributions
Thesis Structure
Chapter 2: Literature Review
Recommendation System
Deep Recurrent Learning
Graph Representation Learning
Fairness in Recommendation
Chapter 3: Multi-Perspective Dynamic Graph Evolution Learning for Recommendation
Brief Introduction of DGEL
Preliminaries
DGEL Framework
Experiments on DGEL
Summary of DGEL
Chapter 4: Temporal Collaboration-Aware Graph Learning for Recommendation
Brief Introduction of TCGC
Preliminaries
TCGC Model Design
Experiments on TCGC
Summary of TCGC
Chapter 5: Temporal Self-correcting Interaction Network for Recommendation
Brief Introduction of CREAM
Preliminaries
CREAM Model Design
Experiments on CREAM
Summary of CREAM
Chapter 6: Dual-side Online Fairness Learning for Recommendation
Brief Introduction of MDOFair
Definitions
MDOFair Method Design
Experiments on MDOFair
Summary of MDOFair
Chapter 7: Conclusion & Future Work
Conclusion of Thesis
Discussion on Future Work

Figure 1.2: The structure of thesis.

CHAPTER 2

LITERATURE REVIEW

This chapter provides a comprehensive literature review on the recommendation systems and the relevant technologies, involving static recommendation research and dynamic recommendation research. Specifically, this chapter presents four aspects:

- Sec. 2.1 introduces two types of recommendation systems, namely Conventional Recommendation (sec. 2.1.1) and Dynamic Recommendation (sec. 2.1.2).
- Sec. 2.2 discusses the existing methods that mainly focus on deep recurrent learning to model the temporal evolving representations.
- Sec. 2.3 discusses the existing methods that mainly utilize graph representation learning for recommendation, which is categorized into Static Graph Learning (sec. 2.3.1) and Temporal Graph Learning (sec. 2.3.2).
- Sec. 2.4 introduce existing fairness studies on recommendation systems to highlight the importance of mitigating fairness for the dynamic recommendation.

This chapter presents a thorough literature review regarding recommendation systems, deep recurrent learning methods, static graph learning methods, temporal graph learning methods, and fairness studies, thereby establishing a solid foundation for the research in this thesis.

2.1 Recommendation System

2.1.1 Conventional One-time Recommendation

Recommendation systems (RecSys), which aim to help users discover their truly preferred items from massive data, play an important role in people's daily life [31]. They have been deployed in various real-world applications, such as e-commerce (e.g., Amazon and Taobao), video sites (e.g., TikTok and YouTube), and social networks (e.g., Facebook and Instagram) [18, 6, 69, 143], meeting users' personalized interests and alleviating the information overload. Learning the features/representations of users and items is the core theme to the RecSys since it determines the process where recommendation systems can match the users and the candidate items [98, 27].

A common paradigm for recommendation study is exploiting the user preferences from the historical user-item interactions and then evaluating the recommendation performance on unseen test data by predicting a recommendation list for each user [33]. Traditionally, collaborative filtering (CF) and matrix factorization (MF) are the two most extensively adopted theologies for recommendation. DMF [111] uses explicit ratings and implicit feedback to build a deep matrix factorization model that maps the users and items into a common low-dimensional space with non-linear projections. NCF [33] presents neural network based collaborative filtering to generalize traditional matrix factorization by leveraging a multi-layer perceptron on the user-item interaction function. Then, CMF [20] proposes the collaborative memory network that combines the memory component and neural attention mechanism to encode complex user-item relations under deeper architecture. Later on, graph neural networks (GNN), which are capable of learning high-order connectivity via user-item interaction graphs, spark a new technological innovation in both academia and industry. For example, LightGCN [32], which simplifies the design of GNN by only involving the neighbor aggregation and the weighted sum, exhibits substantial improvements for recommendation and thus becomes a milestone in this area. Furthermore, SGL [98] explores self-supervised learning on LightGCN by generating different views on the same nodes, yielding a better performance.

However, the critical flaw of conventional recommendation study is that, the one-time recommendation-evaluation paradigm cannot capture the evolution of users and items in dynamic environment and also cannot model the online interaction procedure where users are continuously interacting with items. Therefore, it is necessary to study the evolving preferences of users and refresh the recommendation results in real-time. This leads to dynamic multi-round recommendation.

2.1.2 Dynamic Multi-round Recommendation

In most real-world applications, users tend to interact with items continuously, making it necessary to update temporal representations of both users and items and refresh the recommendation results in a real-time manner [30, 150, 40]. By temporal representations, exploring the evolution of users and items and conducting time-aware predictions becomes possible. Different from conventional one-time recommendation, the dynamic recommendation is a multi-round process where the system would dynamically refresh the recommendation list for every user with the growth of its interactions [44, 103]. In other words, each user could get multiple different recommendation lists during the inferring stage to fulfill its evolving preferences. Hence, dynamic recommendation is more practical and much closer to people's actual online experiences on the web.

How to capture the dynamics with time-aware information and how to update model and also the temporal representations are the major concerns to the dynamic RecSys. Jodie [40] firstly learns the users' embedding trajectories to predict the future interactions by the project and update operations, which is followed by dynamic graph-enhanced DGCF [44]. Meanwhile, some studies propose continual learning to convert dynamic recommendation into continuous data streaming, such as SPMF [88], ContinualGNN [84], and DEGC [30]. They aim to consolidate the information learned from data streaming. Similarly, ReLoop [5] introduces a continuous self-correction framework that dynamically utilizes the errors in previous results to update model for more accurate recommendations. On the other hand, IncCTR [90] and IncMSR [131] focus on maintaining efficiency and performance when incrementally updating the model with the new collected data. In addition, CoPE [140] builds an ordinary differential equation based GNN to learn information propagation for temporal embeddings. iRec[71] proposes an interactive recommendation framework that continuously learns and recommends to simulate the interaction process. The more detailed literature review can be found in Sec. 2.2 and Sec. 2.3.

Sequential recommendation, which explores the sequential dependencies between different items [43], tends to be similar to the dynamic recommendation. They both model the preference change for future item prediction. Most sequential recommendation methods [38, 65, 86, 108, 123], however, still follow the one-time evaluation paradigm. Specifically, in terms of the experimental setting, they only generate one recommendation list to verify the user's test data during the inferring stage. Dynamic recommendation, which aims to conduct the multi-round recommendation process, would continuously and dynamically refresh the recommendation list for each user during the inferring stage [30, 103, 44]. This significantly distinguishes dynamic recommendation from the sequential recommendation. Dynamic RecSys leverages user-item interaction as feedback to update the evolving preferences and follows the continuous recommendation-evaluation paradigm. Therefore, for the dynamic recommendation, not only the preference evolution is dynamic, but also the recommendation process is dynamic.

2.2 Deep Recurrent Learning

Due to the inherent continuity in dynamic environment, deep recurrent learning can be naturally applied to update temporal representations over the stream of user-item interactions. This category explores the information from newly-arrived interactions and inject them into old representations.

In the early days, DeepCoevolve [13] employs recurrent co-evolving networks to model the mutual influence between users and items, while Time-LSTM [151] equips the classical LSTM (long short-term memory network) with time-aware gates to investigate the effects of time intervals. Similarly, LatentCross [3] arms conventional RNN (recurrent neural network) with context encoding to make full use of contextual features for the recurrent recommendations. Then, Jodie [40] designs a novel project function and an update function to dynamically predict the future embedding trajectories of users, achieving a milestone in this category. Meanwhile, SML [139] proposes a novel training method that could efficiently transfer the old model into the new model by meta-learning only on the newly-arrived interactions. Later on, HILI [8] presents a sequential model that aims to prevent the information asymmetry and generate highly liquid embeddings for temporal interactions, and SE-GRU [120] leverages structure-embedded gated recurrent networks to strengthen the temporal interaction prediction. More recently, NeuFilter [103] constructs a robust temporal learning method based on the Kalman filtering to continuously deal with noise interactions. Reinforcement learning, which focuses on the newly-arrived feedback to update its reward-based policy, has been also employed in recommendation systems. [128] Decor-RCFN builds two policy networks to learn the interactive strategy that explores the user preferences in a coarse-to-fine fashion. RecipeRL [55] presents an effective reinforcement learning approach to recurrently express user states over interactive feedback in real-time.

However, most methods in this category, fail to effectively explore the complex high-order connectivity among users and items, thus much likely leading to suboptimal performance in predicting interactions. Graph representation learning, which mainly focuses on node-oriented neighbor aggregation by walking over the interaction graph, can address this issue and help exploit the graph-based collaborative information and the neighbor-based embedding information, strengthening the expressiveness of user and item representations.

2.3 Graph Representation Learning

2.3.1 Static Graph Learning

In recent years, graph neural networks have been widely deployed in various fields such as RecSys, owing to their theoretical elegance and excellent performance [10]. Specifically, GNN performs a message-passing mechanism over user-item interaction graphs to stack representation layers in high-order propagation and aggregate neighborhood information across different hops [126]. A large number of GNN-based methods are devoted to the conventional one-time recommendation study (Sec. 2.1.1). They are called static graph learning since they follow the static recommendation-evaluation setting.

Earlier NGCF [89] injects the collaborative signal into user and item embeddings by propagating them over the user-item graph structure, effectively achieving expressive modeling of high-order connectivity. LightGCN [32], however, argues that the feature transformation and nonlinear activation in GNN contribute little to the performance of graph-based CF. Then, it simplifies the design by removing them and delivers significant improvements over NGCF, making it the milestone in this area. Later on, SCG-SPRe [137] encodes the high-order item correlations in the substitutable graph and the complementary graph into the relation-specific item representations, while GCARM [62] leverages the graph attention mechanism to encode the dynamic correlations between local and global neighbors. Furthermore, SGL [98] proposes a self-supervised graph learning method for recommendation by utilizing contrastive learning to create multiple views of nodes and reinforce node representation learning via self-discrimination. Then, HCCF [105] integrates a more advanced hypergraph encoding with the self-supervised contrastive learning framework, thus winning a further improvement over the SGL. In addition, the hierarchical structure of knowledge graph has been utilized as useful side information to derive knowledge-aware representations and improve the graph network for recommendation, such as KHGT [104] and KGCL [117]. Moreover, CAGCN [93] verifies how message-passing in GNN captures the collaborative signals and proposes a recommendation-oriented metric to guide the neighbor-based aggregation.

Despite their remarkable success, they are still centered on the static recommendation that only adopts one-time recommendation-evaluation paradigm (sec. 2.1.1). However, a dynamic RecSys involves continuous temporal change in dynamic environment, making it necessary to evolve static graph learning into temporal graph learning. Without losing the powerful capability of GNN, temporal graph learning could endow static graph learning with the awareness of temporal evolution in dynamic RecSys.

2.3.2 Temporal Graph Learning

In real-world applications, the user-item interaction graphs usually present complex temporal dynamics that continuously evolve in time and rapidly change the user/item representations [96, 101]. Thus, the temporal graph learning for users and items has significantly drawn attention from both academia and industry since it is more practical to real-world scenarios, such as dynamic RecSys (Sec. 2.1.2). A common solution for temporal graph learning is to operate the temporal graph neural network (T-GNN) on continuous-time dynamic graphs represented as a stream of interactions [67]. Compared with the static GNN, temporal GNN would treat the neighbors as the temporal evolving set and encode the fine-grained time information [146].

TGN [67], as a representative temporal graph network method, first proposes a generic inductive framework to model the dynamic graphs and verifies many previous methods are specific instances of TGNs. Then, DGCF [44] exhibits multiple structural orders to dynamically explore the relationship between users and items, effectively modeling the interactions for dynamic RecSys. IGCN [102], which incrementally employs a temporal convolutional network on time periods, successfully fuses temporal information of time periods and achieves a more accurate temporal-aware feature learning for graph-based CF. Moreover, TGSRec [22] unifies the sequential patterns in recommendation and temporal collaborative signals in graph by a novel temporal collaborative Transformer layer with an advanced attention mechanism.

Later on, TREND [96] presents a temporal Hawkes process-based GNN driven by event-level and node-level dynamics, to capture both individual and collective characteristics for temporal representations learning. Next, ESG [119] develops a temporal convolution component and a structure-aware learner to learn the representations at different time scales for multivariate time series forecasting. GNG-ODE [29] integrates the temporal session graph modeling with ordinary differential equation network, fully preserving the temporal information of item transitions. In addition, FIRE [101] focuses on the efficiency problem of updating/training recommendation model and proposes a fast incremental recommendation method by non-parametric graph signal processing.

More recently, TIGER [141] devises a temporal dual-memory graph network that merges the memory update and temporal embeddings to alleviate the staleness problem in update and also designs a restarter module that aims to help model restart flexibly. Furthermore, AMCNet [127] studies the complex correlations among different dynamics of structural scales, while DynShare [146] models the symmetric interactions and fine-grained time information for share recommendation by building bi-directional temporal GNN. HOPE [57] incorporates both second-order graph convolution and second-order ODE to capture non-neighborhood information and temporal dependencies in graph for better modeling interacting dynamics. DEGC [30], which emphasizes the importance of preference shifts in streaming recommendation, designs a model isolation-based continual graph learning method that operates refining, pruning, and expanding on graphs.

Next, SEAN [135] proposes a selective neighborhood encoding mechanism for automated temporal graph modeling, which can be employed into existing T-GNN methods to achieve further improvement. Similarly, CNE-N [11] presents a new light-cost co-neighbor encoding mechanism by compressing the adjacency matrix and calculating co-neighbors in parallel, which accelerates the computational process. Meanwhile, to improve the scalability, SimpleDyG [100] maps the dynamic graph into a set of sequences and develops a simple yet effectiv Transformer-based architecture to model the dynamic graph without complex modifications. In addition, the knowledge information regarding the interactions, could also be utilized in dynamic environment to uncover the knowledge evolution and strengthen the temporal graph modeling. For example, StreamE [130] continuously accrues new knowledge to temporal knowledge graphs as streams and incrementally update and read temporal representations to conduct real-time prediction. Moreover, HGLS [133] constructs the sub-graph and the global-graph over the raw temporal knowledge graph, then successfully deriving both the short-term and long-term representations based on them.

Despite the remarkable achievements in TGL, it is still challenging to leverage TGL for dynamic recommendation due to the scenario characteristics. The first issue is how to efficiently convert static RecSys into dynamic RecSys with the support of various TGL methods. This would further pose another urgent question about how to comprehensively explore the dynamic evolution from multiple perspectives for both users and items. The second issue is that the newly-arrived interactions are directly learned/injected to update representations. However, not all the newly-arrived interactions would benefit the temporal representation learning. Measuring the effectiveness of them could improve the application of TGL. The third issue is similar to the second one. The historical interactions, which record user long-term behaviours, are not always appropriate to derive user long-term preferences. Some historical interactions may be occasional results. Considering the number of historical interactions is very large, dynamically verifying the influence of them would deliver significant and substantial improvement to the TGL for dynamic RecSys. The last issue is that most existing TGL methods cannot handle the unfairness issue that could destroy the reliability of RecSys. However, the study on fairness is more than the representation learning itself. Thus, it requires another section to review related literature.

2.4 Fairness in Recommendation

The issue of fairness in recommendation has received growing concerns because recommendation systems touch people's lives deeply and profoundly [25, 26]. It affects how people access information and how information is exposed to users [115]. Therefore, numerous fairness-aware methods have been proposed based on different dimensions, including item-side fairness for producers, user-side fairness for consumers, and dual-side fairness for both of them.

On the item side, fairness could be quantified as the exposure each item gets in expectation during inference [34]. Hence, to alleviate the popularity bias, many studies, such as FELIX [34] and FaiRIR [14], re-allocate the exposure by estimating their collective merit to users. On the other hand, the unfairness issue should be considered from not only the item side but also the user side [46]. FairRec [97] and UGF [46] eliminate the recommendation disparity from individual and group perspectives, respectively, to improve the recommendation quality for disadvantaged users. Furthermore, OCEF [17] proposes to audit for envy-freeness aligned with individual preferences, which assumes every user should prefer their recommendations to those of other users. Nevertheless, framing the fairness problem for only items or users may appear oversimplified and cause more unfairness to the neglected side [60]. Thus, some methods leverage dual-side fairness for both users and items by conducting multi-task optimization, such as TFROM [99], CPFair [60] and TSFD [87].

Although the above-mentioned methods have achieved progress in fairness for RecSys, there is still one obvious shortcoming. They focus on static fairness which ignores the rapid temporal evolution of fairness. In other words, most of them derive the global fairness from offline data, without investigating it in the dynamic environment. To this end, FCPO [24] leverages the constrained Markov decision process to model long-term influence of item exposure. Then, a novel method DM [58], adapts the existing discrepancy minimization algorithm into the dynamic settings to mitigate exposure bias in the long run. Meanwhile, from the view of user-side fairness, FADE [121] presents differentiable recommendation metrics to introduce the fairness loss into model fine-tuning. However, they are limited to a single perspective (user side or item side). Thus, studying the dual-side dynamic fairness in one unified framework is still a challenge to dynamic RecSys, which further encourages the research in this thesis. Mitigating the dynamic dual-side unfairness over time for both users and items can build a more reliable and fairer dynamic RecSys.

CHAPTER 3

MULTI-PERSPECTIVE DYNAMIC GRAPH EVOLUTION LEANING FOR RECOMMENDATION

Graph neural network (GNN) based algorithms have achieved superior performance in recommendation tasks due to their advanced capability of exploiting high-order connectivity between users and items. However, most existing GNN-based recommendation models ignore the dynamic evolution of nodes, where users will continuously interact with items over time, resulting in rapid changes in the environment (e.g., neighbor and structure). Moreover, the heuristic normalization of embeddings in dynamic recommendation is decoupled with the model learning process, making the whole system suboptimal. In this chapter, we propose a novel framework for generating satisfying recommendations in dynamic environments, called **D**ynamic **G**raph **E**volution **L**earning (DGEL). First, we design three efficient real-time update learning methods for nodes from the perspectives of inherent interaction potential, time-decay neighbor augmentation, and symbiotic local structure learning. Second, we construct the re-scaling enhancement networks for dynamic embeddings to adaptively and automatically bridge the normalization process with model learning. Third, we leverage the interaction matching task and the future prediction task together for joint training. Extensive experiments on three real-world datasets demonstrate the improvements of our proposed DGEL.

The content in this chapter mainly focuses on addressing the research challenge **RC 1** and overcoming the limitation **EL 1** by proposing a novel framework DGEL.

3.1 Brief Introduction of DGEL

Personalized recommendation systems that help users discover items they are likely to consume alleviate the information overload in many applications, such as e-commerce (e.g., Amazon and Taobao) and social network platforms (e.g., Facebook and Instagram) [6, 143, 69, 18]. In recent years, graph neural networks (GNN), which model user-item interactions as a graph, have achieved phenomenal success in recommendation tasks [142]. The domination of GNN in various fields profits from its advanced ability to preserve graph topological structure information [110, 66] and node intrinsic features aggregated from neighbors and itself [75, 89]. A significant superiority of such GNN-based recommendation models is that they exploit the possibility of linking users and items in the interaction graph and the high-order connectivity therein by explicitly propagating the node embeddings over the graph [21, 53]. Hence, from LightGCN [32] to HCCF [105], we can observe that the improvements for recommendation lie at the heart of the neighboring aggregation function for deep representation learning, which determines the recommendation performance.

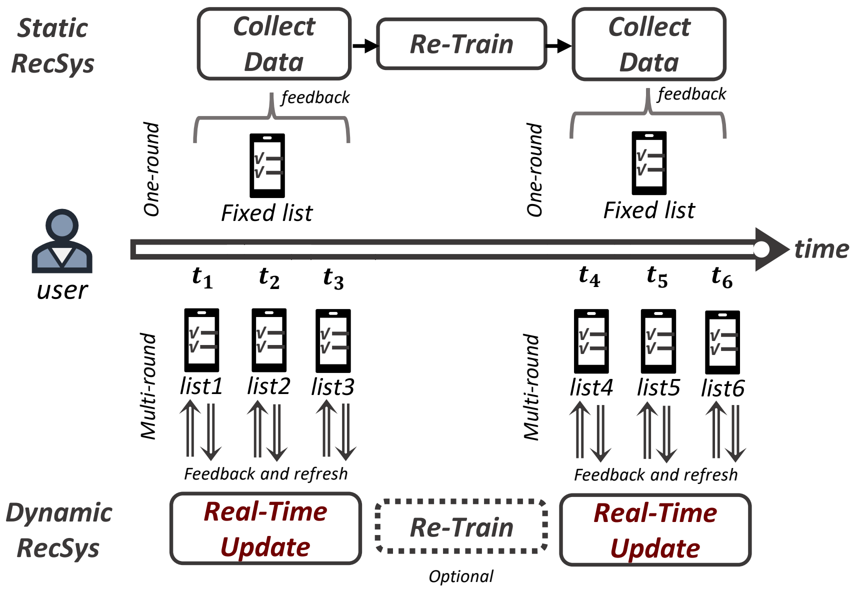


Figure 3.1: Dynamic multi-round recommendation vs. traditional one-round recommendation. For dynamic recommendation, the core is to precisely capture user and item dynamics and update the recommendation list in real time.

However, most existing GNN-based recommendation models are consistent with traditional one-round recommendation patterns [105, 19, 21, 41], which generate a fixed recommendation list for the target user during the prediction stage and lack the real-time update strategy when the user preferences change [44, 101]. In other words, it significantly ignores the dynamic evolution of nodes in the interaction graph over time. For example, in real-world recommendation scenarios, users tend to interact with items continuously at distinct times, resulting in rapid evolution and changes in the environment, such as neighbors and structure [102]. The discrepancy between multi-round and one-round recommendations is illustrated in Figure 3.1. Although some models have considered temporal context to correct representation from the graph [73, 79, 50, 104], the ultimate return lists are still designed under traditional one-time recommendation strategies. In addition, the user preferences are dynamic and diverse over time [22, 35], making it necessary to update and modify the real-time recommendation list to satisfy the user requirements by capturing the current user dynamic preferences. Another tough challenge is that the heuristic and intuitive normalization methods in GNN (e.g., L2 normalization) decoupled from model learning can lead to the inflexible limitation to the value range of each node embedding and cause undesirable effects, such as scale distortion issues between different embeddings [74]. Namely, the performance of the whole system is likely to become sub-optimal when applying such static pre-defined normalization. Unfortunately, despite the urgent desire for adaptive and automatic normalization design in the dynamic recommendation, it is frustrating that such an issue does not receive enough attention.

To tackle the above-mentioned challenges, we propose a novel framework for the multi-round dynamic recommendation, which is named **D**ynamic **G**raph **E**volution **L**earning (DGEL). Specifically, to learn user and item dynamics effectively and efficiently, we design three learning methods from different perspectives to accurately represent dynamic sub-embeddings in DGEL, i.e., inherent interaction potential (IIP), time-decay neighbor augmentation (TNA), and symbiotic local structure learning (SLS). Furthermore, user-item interactions follow a sequential order and bring temporal evolution [96], forcing our DGEL to update and modify recommendations at each distinct time when a new interaction is formed. Meanwhile, the heuristic and intuitive normalization methods, which are independent of dynamic update and model learning, impose limitations on these dynamic embeddings, restricting the performance of the whole dynamic recommendation system. Considering we design individual dynamic embeddings from three perspectives, it is essential to bridge normalization with the learning process of our framework to adaptively and automatically modify these dynamic embeddings. Hence, we construct the re-scaling enhancement networks (RENet) to make embeddings comparable and to fit differences between inputs. Moreover, inspired by the idea of multi-tasks to compensate for the absence of rich information in the sparse graph [98], we leverage joint training for the interaction matching task and the future prediction task to ensure that the rapid evolution of nodes is consistent with the temporal-linear graph pattern. We summarize the contributions of this work as follows:

- To investigate user and item dynamics precisely, we carefully devise inherent interaction potential, time-decay neighbor augmentation, and symbiotic local learning to update node dynamic embeddings with rich graph information comprehensively.
- We propose the re-scaling enhancement networks to bridge the normalization with the dynamic embedding learning process adaptively, which can keep the consistency and comparability of dynamic embeddings and avoid the model becoming suboptimal.
- To further improve the performance and ensure the temporal-linear evolution of nodes, we leverage two tasks, the interaction matching task and the future prediction task together for joint training.
- We utilize real-world datasets to evaluate the effectiveness and superiority of our proposed model. Furthermore, we conduct ablation studies and variant research to confirm the importance of each proposed component in our model.

3.2 Preliminaries

In this section, we first briefly introduce the graph neural network for recommendation and then define the dynamic recommendation with the evolving graph over time.

Recap Graph-Based Paradigm for Recommendation. Suppose the recommendation scenario involves m users and n items with the user set $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ and item set $\mathcal{V} = \{v_1, v_2, ..., v_n\}$. Edges in the user-item interaction graph $\mathcal{G}$ are built if user u_i has interacted with item v_j (e.g., click action in e-commerce). To begin with, we initialize users and items via a d-dimensional latent space to encode their representation under interaction patterns. More precisely, $\mathbf{h}_i \in \mathbb{R}^d$ and $\mathbf{h}_j \in \mathbb{R}^d$ are generated for user u_i and item v_j, respectively. In addition, all the users and items compose the embedding matrices $\mathbf{H}^{\mathcal{U}} \in \mathbb{R}^{m\times d}$ and $\mathbf{H}^{\mathcal{V}} \in \mathbb{R}^{n\times d}$. Upon the constructed interaction graph, the core idea of the graph-based paradigm lies in the information aggregation functio (e.g., average aggregation) [106]. Inspired by LightGCN [32], an efficient aggregation for both users and items could be indicated as follows:

$$\mathbf{h}_i^{(k+1)} = \sum_{j\in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_j|}} \mathbf{h}_i^{(k)}, \mathbf{h}_j^{(k+1)} = \sum_{i\in N_j} \frac{1}{\sqrt{|N_i|}\sqrt{|N_j|}} \mathbf{h}_j^{(k)}, \tag{3.1}$$

where $\mathbf{h}_i^{(k+1)}$ and $\mathbf{h}_j^{(k+1)}$ denote the refined embedding of user u_i and item v_j after k convolutional layers, respectively . N_i denotes the set of items that are interacted with user u_i while N_j denotes the set of users that interact with item v_j. With the support of such aggregations, capturing local graph structure information to learn better node representation for making link prediction becomes a reality.

Evolving Graph for Dynamic Recommendation. In real-world scenarios, users tend to interact with items continuously, leading to the rapid evolution of the graph. The evolution includes new interaction edges, new users and item nodes, and new graph structure. More importantly, the node embeddings would be updated due to the change of their neighbors. It is worth noting that we do not consider the removal of interaction edges. Now we define evolving graph for dynamic recommendation according to work [40, 44, 102], that is $\mathcal{G}^t$, the snapshot at time t. It is formed by $\mathcal{S}$ which represents an ordered sequence of temporal user-item interactions in the graph. The r-th interaction happened between user u_i and item v_j is denoted as $s_r^{i,j} = \left(u_i, v_j, t, f\right)$, where t and f represent the timestamp and feature vector of the interaction, respectively (note that not all the interactions have features). Hence, as interactions are observed in temporal order, the graph will evolve over time. As a consequence, the recommendation system needs to modify its recommendation results. That is also consistent with real-world applications where users will not only receive a fixed recommendation list.

The core component of dynamic recommendation based on the interaction graph is how to update dynamic embedding efficiently since the graph evolves all the time. Here, we devise two embeddings for each node in the graph, i.e., static and dynamic embeddings, to encode both the long-term stationary properties of the entities and their dynamic properties. Static embeddings (e.g., one-hot vectors or side-information vector) of user u_i and item v_j, denoted as $\bar{\mathbf{h}}_i \in \mathbb{R}^{\bar{d}}$ and $\bar{\mathbf{h}}_j \in \mathbb{R}^{\bar{d}}$, will keep unchanged. While dynamic embeddings, denoted as $\tilde{\mathbf{h}}_i \in \mathbb{R}^{\tilde{d}}$ and $\tilde{\mathbf{h}}_j \in \mathbb{R}^{\tilde{d}}$, evolve over time to model their time-varying representations. Therefore, our goal is to update node dynamic embeddings accurately after observing current interaction at time t and generate a modified recommendation list for future $t+1$ based on that:

$$\tilde{\mathbf{h}}_i^t, \tilde{\mathbf{h}}_j^t = F_\Theta\left(\tilde{\mathbf{h}}_i^{t-1}, \tilde{\mathbf{h}}_j^{t-1}, f\right), \tag{3.2}$$

$$Rec_i^{t+1} = \{v_j \mid v_j \in \mathcal{V}, [\tilde{\mathbf{h}}_j^t \parallel \bar{\mathbf{h}}_j] \simeq [\tilde{\mathbf{h}}_i^{t+1} \parallel \bar{\mathbf{h}}_i]\}_k, \tag{3.3}$$

where F_Θ is our framework with parameter set Θ that investigates node dynamics from various generative perspectives and f is the feature of current interaction. Rec_i^{t+1} is the new recommendation list for the user u_i at future time $t+1$ when he/she uses our recommendation service again and k is the length of list. $[x \parallel y]$ represents the concatenation of vector x and y. Moreover, $\simeq$ indicates that we aim to discover the target items nearest to user embeddings at time $t+1$.

3.3 DGEL Framework

In this section, we present the overall architecture of DGEL in Figure 3.2. First, we devise three dynamic learning methods, i.e., inherent interaction potential, time-decay neighbor augmentation, and symbiotic local structure learning, to comprehensively capture node dynamics over time. Second, we construct the re-scaling enhancement network to bridge the normalization with the dynamic embedding learning process to seek consistency and comparability of dynamic embeddings and avoid scale distortion issues. Third, we leverage the future prediction task and interaction matching task together for joint training to improve the overall performance of the proposed DGEL framework. Technical details are discussed in the following sub-sections.

3.3.1 Dynamic Representation Learning

3.3.1.1 Inherent Interaction Potential

Inspired by [40, 22], we assume that the current interaction between user u_i and item v_j at time t plays a key role in reflecting the user and item current states, leading to more meaningful dynamic embeddings. Also, the interaction itself indicates the explicit attraction for user u_i and item v_j to each other, revealing the inherent potential of linking user u_i with item v_j under dynamic circumstances in the graph. We call this Inherent Interaction Potential (IIP).

Since our scenario belongs to interaction-based multi-round dynamic recommendation, the first and foremost objective is to investigate IIP for user u_i and item v_j with the following form:

$$\mathbf{h}_i^t(IIP) = \sigma(\mathbf{W}_{11}^u \tilde{\mathbf{h}}_i^{t-1} + \mathbf{W}_{12}^u \tilde{\mathbf{h}}_j^{t-1} + \mathbf{W}_{13}^u f + \mathbf{W}_{14}^u \Delta t_i), \tag{3.4}$$

$$\mathbf{h}_j^t(IIP) = \sigma(\mathbf{W}_{11}^v \tilde{\mathbf{h}}_j^{t-1} + \mathbf{W}_{12}^v \tilde{\mathbf{h}}_i^{t-1} + \mathbf{W}_{13}^v f + \mathbf{W}_{14}^v \Delta t_j), \tag{3.5}$$

where $\mathbf{h}_i^t(IIP), \mathbf{h}_j^t(IIP) \in \mathbb{R}^{\tilde{d}}$ represent the updated embeddings for user u_i and item v_j at time t from the perspective of inherent interaction potential. $\tilde{\mathbf{h}}_i^{t-1}$ and $\tilde{\mathbf{h}}_j^{t-1}$ are the dynamic

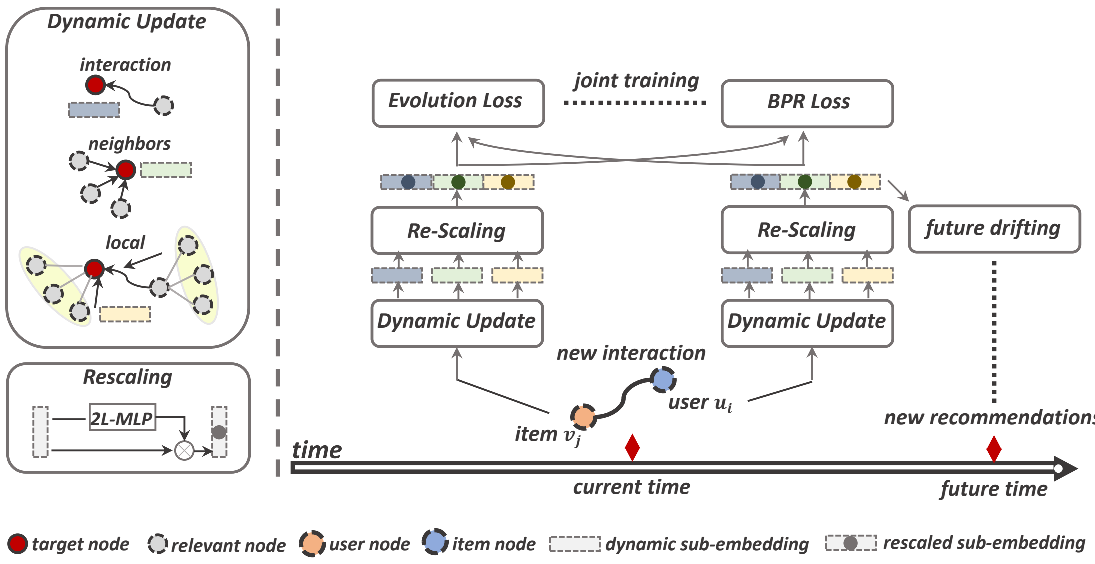


Figure 3.2: The illustration of DGEL's overall architecture. We capture user and item dynamics using three learning methods from the perspectives of interaction, neighbors, and local structure. Then we construct the re-scaling enhancement network to bridge the normalization of dynamic embeddings with model learning. Last, we leverage two tasks, the interaction matching task and the future prediction task together for joint training.

embeddings at previous time. In addition to nodes, we also consider the current interaction feature f. Δt_i denotes the time interval between the current interaction and the previous interaction of user u_i (with any item) while Δt_j is the time interval between current interaction and previous interaction of item v_j (with any user). $\mathbf{W}^u_{11-13} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}, \mathbf{W}^u_{14} \in \mathbb{R}^{\tilde{d}}$, and $\mathbf{W}^v_{11-13} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}, \mathbf{W}^v_{14} \in \mathbb{R}^{\tilde{d}}$ are learnable weight matrices for users and items, respectively. σ is the activation function throughout this chapter (e.g., Tanh and LeakyReLU). The reason that we devise two channels $\mathbf{W}^u_{11-14}$ and $\mathbf{W}^v_{11-14}$ is the explicit updating trend from the current interaction exhibits different dynamic levels for users and items. In general, user dynamics are more sensitive to interaction than item dynamics.

3.3.1.2 Time-Decay Neighbor Augmentation

Having investigated user-item interaction behaviors, we seek to explore the influence information from historical records of nodes (both users and items) that reveals the long-term dynamic attributes. In section 3.1, we recap the graph convolutional paradigm by neighbors. However, adopting static attributes (e.g., degrees of nodes) makes such methods (e.g., LightGCN [32]) impotent to freeze the snapshots at distinct times.

Moreover, simple mean or graph attention [44] in dynamic neighbor aggregation will lose the temporal information severely. Hence, following the work in [102, 96], to construct efficient neighbor aggregation that maintains the influence derived by time interval, we propose the single-convolution time-decay neighbor augmentation (TNA) aggregation as formulated below:

$$\mathbf{h}_i^t(TNA) = \sigma(\mathbf{W}_{21}^u \tilde{\mathbf{h}}_i^{t-1} + \sum_{j \in N_i} \mathbf{W}_{22}^u \kappa^u \tilde{\mathbf{h}}_j^{t-1}), \tag{3.6}$$

$$\mathbf{h}_j^t(TNA) = \sigma(\mathbf{W}_{21}^v \tilde{\mathbf{h}}_j^{t-1} + \sum_{i \in N_j} \mathbf{W}_{22}^v \kappa^v \tilde{\mathbf{h}}_i^{t-1}), \tag{3.7}$$

where $\mathbf{h}_i^t(TNA), \mathbf{h}_j^t(TNA) \in \mathbb{R}^{\tilde{d}}$ represent the updated embeddings for user u_i and item v_j at time t from the view of time-decay neighbor augmentation. N_i denotes the set of items that are interacted with user u_i, and N_j denotes the set of users that interact with item v_j. $\mathbf{W}_{21,22}^u \in \mathbb{R}^{\tilde{d} \times \tilde{d}}$ and $\mathbf{W}_{21,22}^v \in \mathbb{R}^{\tilde{d} \times \tilde{d}}$ are learnable weight matrices for users and items, respectively. Our function κ outputs the time-weight of each neighbor that is based on Softmax: $\kappa^u(t_i - t_j) = exp(\tau_{ij}) / \sum_{j^- \in N(i)} exp(\tau_{ij^-}), \tau_{ij} = -(t_i - t_j) / max(t_i - t_{j^- | j^- \in N(i)})$. Instead of the complex time-aware attention score that requires additional training overhead, we adopt the negative time-interval normalization for Softmax, which is fast to compute without other hyperparameters. Note that, the neighbor sets for both user u_i and item v_j are only the snapshots at current time t and will change over time as new interactions between users and items will form. Consequently, our time-decay neighbor augmentation keeps consistent with real-world scenarios.

3.3.1.3 Symbiotic Local Structure Learning

Although the aforementioned inherent interaction potential and time-decay neighbor augmentation endow our DGEL with the capability of exploiting the node evolution from interactions and relevant neighbors, they are still decoupled from each other and work in their own way. Hence, we further supercharge our DGEL with Symbiotic Local Structure Learning (SLS), which absorbs the impact of interaction and the excitements from the neighbors. The excitements measure the tendency of current interaction happening under the environment (or context) composed of neighbors. In our assumption, forming a new interaction between user u_i and item v_j depends on their inherent features and mutual neighbors simultaneously. It is reasonable to encode the interaction and neighbors into a local structure. Towards this end, we apply pooling operation [110] to realize our SLS for conserving computational cost. Formally, the SLS for user u_i and item v_j is derived as follows:

$$\begin{aligned}\mathbf{h}_i^t(SLS) = \sigma(\mathbf{W}_{31}^u(\tilde{\mathbf{h}}_i^{t-1} - \tilde{\mathbf{h}}_j^{t-1})^p + \mathbf{W}_{32}^u \text{avg}(\{\tilde{\mathbf{h}}_z^{t-1}, \forall z \in N_i\}) \\ + \mathbf{W}_{33}^u \text{avg}(\{\tilde{\mathbf{h}}_x^{t-1}, \forall x \in N_j\})),\end{aligned} \tag{3.8}$$

$$\begin{aligned}\mathbf{h}_j^t(SLS) = \sigma(\mathbf{W}_{31}^v(\tilde{\mathbf{h}}_j^{t-1} - \tilde{\mathbf{h}}_i^{t-1})^p + \mathbf{W}_{32}^v \text{avg}(\{\tilde{\mathbf{h}}_x^{t-1}, \forall x \in N_j\}) \\ + \mathbf{W}_{33}^v \text{avg}(\{\tilde{\mathbf{h}}_z^{t-1}, \forall z \in N_i\})),\end{aligned} \tag{3.9}$$

where $\mathbf{h}_i^t(SLS), \mathbf{h}_j^t(SLS) \in \mathbb{R}^{\tilde{d}}$ denote the updated embeddings for user u_i and item v_j at time t from the perspective of symbiotic local structure learning. $(x-y)^p$ represents the element-wise power function between x and y where $(x-y)^p = (y-x)^p$ when $p = 2$. Otherwise, the outputs are opposite for the user and item to focus on their own learning injection when $p = 1$. $\mathbf{W}_{31-33}^u \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ and $\mathbf{W}_{31-33}^v \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ are learnable weight matrices for users and items, respectively. To model the discrepancy between users and items on dynamic sensitivity, similar to Section 4.1.1, we construct two channels to process the same pooling operation on user neighbors and item neighbors. The SLS helps us to understand the implicit correlation between two local groups under the light paradigm of structure learning. So far, we have obtained three categories of dynamic embeddings. Next, we will introduce how we re-scale these embeddings to form the final dynamic embedding.

3.3.2 Re-scaling Enhancement Network

3.3.2.1 Bridge the Normalization with Model Learning

Even though the superiority of GNNs over shallow graph embeddings comes from various aggregations and encodings, there is still one common weakness in such aggregation and encoding mechanism of many dynamic recommendation models [102, 22, 40, 96, 44]. Heuristic and intuitive normalization methods (e.g., L2-Norm) on representations of nodes are independent of the model learning, causing inflexible limitations to the value range of node embedding and undesirable effects (e.g., the scale distortion issue between different embeddings). Since we derive dynamic sub-embeddings from three novel generative perspectives (see Section 4.1), it is essential to link the normalization coefficients to the learning process of our model to endow it with an automatic and adaptive ability to determine the appropriate scaling degree.

Motivated by [74], we propose the Re-scaling Enhancement Network (RENet) to further scale the dynamic embeddings. To the best of our knowledge, it is the first time applying the neural re-scaling strategy on dynamic recommendations. Without losing generality, here we use $\tilde{h}$ to represent dynamic embeddings derived from any perspectives (in Section 4.1) and define our RENet with a 2-layer MLP as follows:

$$G(\tilde{h}) = \sigma_2(\mathbf{w}_2\sigma_1(\mathbf{W}_1\tilde{h} + \mathbf{b}_1) + b_2), \tag{3.10}$$

$$\tilde{h}^{\circ} = G(\tilde{h}) \cdot \tilde{h}, \tag{3.11}$$

where $\mathbf{W}_1 \in \mathbb{R}^{\tilde{d}/2\times\tilde{d}}$, $\mathbf{b}_1 \in \mathbb{R}^{\tilde{d}/2}$, $\mathbf{w}_2 \in \mathbb{R}^{\tilde{d}/2}$, and $b_2 \in \mathbb{R}$ are the learnable parameters of $G(\cdot)$ that outputs a re-scaling factor. σ_1 is the activation of the first layer, which could be any option (e.g., Tanh and LeakyReLU) while σ_2 is the activation of the second layer and it determines the value of re-scaling factor. In [74], σ_2 is the $Sigmoid$ function to force the output value in $(0,1)$. However, we observe that the value range of each dimension in node embeddings is too small to bring obvious numerical variation in the second layer before $Sigmoid$, which means the output of $Sigmoid$ always approaches $Sigmoid(0) = 0.5$ (e.g., 0.48 and 0.52). Thus, we select LeakyReLU as the activation of the second layer to allow more valuable space for the factor. $G(\cdot)$ could adaptively control the scaling intensity according to the dynamic embedding. Meanwhile, we separately assign independent RENets for the distinct-perspective dynamic embeddings, such that our model can distinguish the impact of them. There are some other re-scaling variants (e.g., Feed-Forward Network), and we will compare them with our RENet.

3.3.2.2 Final Dynamic Representation

To fulfill the final dynamic embedding of user u_i and achieve evolution learning for user, we adopt concatenation of the re-scaled $\mathbf{h}_i^t(IIP)$, $\mathbf{h}_i^t(TNA)$ and $\mathbf{h}_i^t(SLS)$ (same operation on item) for two reasons: 1) maintain their own dynamic information from the various perspectives of interaction, neighbors, and local structure; 2) keep the consistency with RENet since RENet corrects these sub-embeddings individually and enhances the expression of them. Moreover, we adopt a learnable weight sum of dynamic states from the current time and previous time because remembering the essential information from the previous embedding at time $t-1$ is a common idea in the sequential recommendation. Therefore, the final dynamic embeddings for user u_i and item v_j at time t are constructed as follows:

$$\tilde{\mathbf{h}}_i^t = \sigma(\mathbf{W}_3^u[\mathbf{h}_i^t(IIP)\|\mathbf{h}_i^t(TNA)\|\mathbf{h}_i^t(SLS)] + \mathbf{W}_4^u\tilde{\mathbf{h}}_i^{t-1} + \mathbf{b}_3), \tag{3.12}$$

$$\tilde{\mathbf{h}}_j^t = \sigma(\mathbf{W}_5^v[\mathbf{h}_j^t(IIP)\|\mathbf{h}_j^t(TNA)\|\mathbf{h}_j^t(SLS)] + \mathbf{W}_6^v\tilde{\mathbf{h}}_j^{t-1} + \mathbf{b}_4), \tag{3.13}$$

where $\mathbf{W}_3^u \in \mathbb{R}^{\tilde{d}\times 3\tilde{d}}, \mathbf{W}_4^u \in \mathbb{R}^{\tilde{d}\times\tilde{d}}, \mathbf{b}_3 \in \mathbb{R}^{\tilde{d}}$ and $\mathbf{W}_5^v \in \mathbb{R}^{\tilde{d}\times 3\tilde{d}}, \mathbf{W}_6^v \in \mathbb{R}^{\tilde{d}\times\tilde{d}}, \mathbf{b}_4 \in \mathbb{R}^{\tilde{d}}$ are learnable parameters for users and items, respectively. Here, we finally obtain the user and item dynamic embeddings after dynamic evolution learning and re-scaling enhancement networks. Note that our dynamic model DGEL only updates the corresponding user u_i and item v_j of the current interaction instead of updating all nodes in the interaction graph. In the following sections, we will present the details about leveraging multi-tasks in the dynamic recommendation for joint training.

3.3.3 Multi-Task Joint Training

3.3.3.1 Match the Current User and Item

Having established the evolution of user u_i and item v_j based on the current interaction at t, we treat them as the positive pair to measure the level of matching the user and item under our DGEL framework. The auxiliary supervision of positive pairs ensures the accuracy of updated dynamic embeddings according to the fact they have linked with each other at the current time. For simplicity, we adopt the BPR loss function to calculate the matching level of all the positive pairs from the ordered interaction sequence:

$$\mathscr{L}_{bpr} = \frac{1}{|\mathscr{S}|}\sum_{t=1}^{|\mathscr{S}|} Log(\sigma(s_t^{i,j} - s_t^{i,j^-})), s_t^{i,j} \in \mathscr{S}, s_t^{i,j^-} \notin \mathscr{S}, \tag{3.14}$$

where inner-product $s_t^{i,j} = \tilde{\mathbf{h}}_i^t \times \tilde{\mathbf{h}}_j^t$ is positive pair belonging to interaction set $\mathscr{S}$ and s_t^{i,j^-} is sampled negative pair for user u_i. σ is $Sigmoid$ function. In our assumption, the positive pairs brought by real interactions will amplify the distinction from fake interactions via this BPR loss.

3.3.3.2 Future Drifting and Evolution Loss

Dynamic recommendation methods usually generate embeddings only when users take actions on items and do not explicitly model the future trajectory of the user in the embeddings space. Thus, motivated by [40, 44], we introduce a future drifting component to predict the future embeddings of user preference and target item at any future time, which encourages the flexibility and extendibility of our framework:

$$\tilde{\mathbf{h}}_i^{*(t+1)} = (1 + \triangle(t+1)\mathbf{w}_1) \cdot \tilde{\mathbf{h}}_i^t, \tag{3.15}$$

$$\tilde{\mathbf{v}}_j^{*(t+1)} = \mathbf{W}_2\tilde{\mathbf{h}}_i^{*(t+1)} + \mathbf{W}_3\bar{\mathbf{h}}_i + \mathbf{W}_4\tilde{\mathbf{h}}_j^t + \mathbf{W}_5\bar{\mathbf{h}}_j + \mathbf{b}, \tag{3.16}$$

where $\tilde{\mathbf{h}}_i^{*(t+1)}$ is the predicted future dynamic representation of u_i at time $t+1$ and $\triangle$ calculates the interval between time t and time $t+1$. $\bar{\mathbf{h}}_i$ and $\bar{\mathbf{h}}_j$ are static embeddings of user u_i and item v_j. $\mathbf{W}_1 \in \mathbb{R}^{\tilde{d}}, \mathbf{W}_{2,4} \in \mathbb{R}^{(\tilde{d}+\bar{d})\times\tilde{d}}, \mathbf{W}_{3,5} \in \mathbb{R}^{(\tilde{d}+\bar{d})\times\bar{d}}, \mathbf{b} \in \mathbb{R}^{\tilde{d}+\bar{d}}$ are learnable parameters to obtain the predicted item embedding $\tilde{\mathbf{v}}_j^{*(t+1)}$ that user will interact with in future. Note that, in Equation 3.3 (see Section 3.2), we update recommendation results and generate a new list according to the combination of two parts: dynamic embedding and static embedding. Therefore, $\tilde{\mathbf{v}}_j^{*(t+1)}$ is the $(\tilde{d}+\bar{d})$-dimensional embedding and once user u_i access our system at future time $t+1$, we will return top-k items whose embeddings are nearest to $\tilde{\mathbf{v}}_j^{*(t+1)}$. The $\tilde{\mathbf{h}}_i^t$ encodes the user preference learned from continuous interactions and captures the inherent need of the user in the long run, while $\tilde{\mathbf{v}}_j^{*(t+1)}$ indicates the future preference drifting from the current timestamp and enables the discovery of more potential items in the future.

In practice, we propose the evolution loss to minimize the distance between the predicted item embeddings and ground truth item embedding at every interaction. We compute the evolution loss below:

$$\mathscr{L}_{evol} = \sum_{i,j,t\in\mathscr{S}} \|\tilde{\mathbf{v}}_j^{*(t+1)} - \mathbf{h}_j^{(t+1)}\|^2 + \|\tilde{\mathbf{h}}_i^t - \tilde{\mathbf{h}}_i^{t-1}\|^2 + \|\tilde{\mathbf{h}}_j^t - \tilde{\mathbf{h}}_j^{t-1}\|^2, \tag{3.17}$$

where the first term minimizes the predicted embeddings error and $\mathbf{h}_j^{(t+1)} = [\tilde{\mathbf{h}}_j^{(t+1)}, \bar{\mathbf{h}}_j]$ is the ground truth that user actually interact with at time $t+1$. $\|\cdot\|^2$ is the L2 regularization. The last two terms are added to regularize the evolution loss and protect the consecutive dynamic evolution of users and items in order to avoid varying too much.

3.3.3.3 Joint Training

To improve the overall performance of the proposed DGEL framework for generating recommendations in dynamic environments, we leverage a multi-task training strategy to jointly optimize the interaction matching task and dynamic evolution task as follows:

$$\mathscr{L} = \mathscr{L}_{evol} + \alpha\mathscr{L}_{bpr} + \delta\|\Theta\|^2, \tag{3.18}$$

where Θ is the set of model parameters. α controls the BPR loss value to balance it with evolution loss and δ controls the strength of L_2 regularization. We fully take the benefits of multi-task joint training in our scenario: 1) investigate the cohesion of current interaction by the corresponding updated user node and item node; 2) make accurate predictions for the future that allows user preference to drift without limitation of time.

3.3.4 Model Complexity

Since our framework only updates the corresponding user and item of current interaction to modify recommendation results, the model complexity totally depends on the number of interactions. The inherent interaction potential takes $\mathcal{O}(|\mathcal{S}| \times \tilde{d})$ complexity where $|\mathcal{S}|$ is the number of interactions and $\tilde{d}$ is the dimension of dynamic embeddings. Additionally, time-decay neighbor augmentation and symbiotic local structure learning both take $\mathcal{O}(|\mathcal{S}| \times |N| \times \tilde{d})$ where $|N|$ is the number of neighbors. The re-scaling network takes $\mathcal{O}(L \times |\mathcal{S}| \times \tilde{d})$ cost where L denotes the number of layers (which is set to 2 in our DGEL). In the future drifting module for prediction, owing to the combination of dynamic and static embeddings, the cost is $\mathcal{O}(|\mathcal{S}| \times (\tilde{d} + \bar{d}))$ where $\bar{d}$ is the dimension of static embedding. The computation cost for the multi-task joint loss function is $\mathcal{O}(|\mathcal{S}| \times (\tilde{d} + \bar{d}))$. Last, we can complete the top-k list in a small linear time based on the predicted future item embedding by data structure algorithms instead of calculating the scores of all items.

3.4 Experiments on DGEL

Extensive experiments are conducted to evaluate the performance of our proposed model DGEL by answering the following research questions:

- **RQ1**: How dose our DGEL perform when comparing with the state-of-art dynamic recommendation methods?
- **RQ2**: How do different modules (e.g., time-decay neighbor augmentation, re-scaling enhancement network, etc.) contribute to the overall performance of DGEL?
- **RQ3**: What are the effects of other optional variants (e.g., sum-pooling local learning, feed-forward network re-scaling, etc.) in our DGEL?
- **RQ4**: How do the key hyper-parameters (e.g., length of the historical neighbor set, dimension of dynamic embedding) influence the performance of our DGEL?

3.4.1 Experimental Settings

3.4.1.1 Datasets

We conduct experiments on three public datasets collected from different real-world platforms. **Reddit**[1]: this dataset consists of one month of user posts on subreddits. We select

[1]http://files.pushshift.io/reddit/

Table 3.1: Statistics of the experimental datasets.

Dataset	User	Item	Interaction	Average	Feature
Reddit	10,000	984	672,447	67	✓
Wikipedia	8,227	1,000	157,474	19	✓
Foursquare	1,000	950	209,730	209	✗

the 1,000 most active subreddits as items and the 10,000 most active users, resulting in 672,447 interactions. Besides, the text of each post is converted into a feature vector representing their LIWC categories [74]. **Wikipedia**[2]: this dataset contains one month of edits on Wikipedia. 157,474 interactions are filtered out for recommendation by the editors who made at least 5 edits and the 1,000 most edited pages. Also, the edited texts are converted into LIWC vectors as features. **Foursquare**[3]: this dataset contains check-ins in Tokyo collected for about 10 months [113]. We choose the 1,000 most active users and the 1,000 most visited locations for the recommendation task, totaling 209,730 check-ins. In this dataset, interactions do not have features. Table 3.1 presents the statistical information of our experimented datasets.

3.4.1.2 Baselines for Comparison

We compare our DGEL with various outstanding dynamic recommendation methods for performance evaluation.

- **Time-LSTM** [151]: A LSTM variant to model user sequential actions, equipping LSTM with time gates to model time intervals.
- **DeepCoevolve** [13]: A deep co-evolutionary network to define and capture complex mutual influence between users and items.
- **LatentCross** [3]: An easy-to-use technique to incorporate contextual data by embedding the context feature and performing an element-wise product of them.
- **CTDNE** [61]: A general framework that gives rise to methods for learning time-respecting embeddings from continuous-time dynamic networks.
- **DGCF** [44]: A novel framework leveraging dynamic graphs to capture collaborative and sequential relations of both items and users at the same time.

[2]https://meta.wikimedia.org/wiki/Data_dumps

[3]https://sites.google.com/site/yangdingqi/home/foursquare-dataset

- **FIRE** [101]: A fast-incremental non-parametric recommendation method from a graph signal processing perspective that does not suffer from the time-consuming back-propagation.
- **TREND** [96]: A novel framework for temporal graph representation learning driven by temporal events and node dynamics and built upon a Hawkes process-based graph neural network.

3.4.1.3 Evaluation Protocols and Metrics

For a fair comparison, we strictly follow the setting of the previous work [40] and employ the same interaction-based all-ranking strategy to conduct our experiments. Specifically, we use the first 80% data to train, the subsequent 10% to validate, and the final 10% to test. We evaluate the performance of models in terms of the Mean Reciprocal Rank (MRR) and Recall@10. Since we consider the multi-round dynamic recommendation scenarios, we measure each test interaction in chronological order, which is totally different from static GNN-based recommendation evaluation. For each interaction, the ranking of the ground truth item is computed with respect to all the items in the dataset by their L_2 distance from predicted item embedding (see Equation 3.16). We train the proposed DGEL by adopting temporal batch algorithm [40, 44], which follows three criteria: 1) it should process the interactions in each batch simultaneously; 2) the batching process should maintain the original sequential ordering; 3) each batch should not have duplicate nodes therein (it will lead to conflict when updating the same node). Note that we do not re-train all the dynamic models on the testing data to avoid touching future information and leading to unfair predictions during the test stage.

3.4.1.4 Hyperparameter Settings

We adopt AdamW [56] with the learning rate of $1e-3$ as our optimizer for training. We only use a single propagation layer for both time-decay neighbor augmentation and symbiotic local structure learning. In the re-scaling enhancement network, the dimension of the first layer is the size of input dynamic embedding and the dimension of the second layer is half of the first layer. The dimension of dynamic embedding and the length of the historical neighbor set are selected from $\{16, 32, 64, 128\}$ and $\{50, 100, 150, 200, 250, 300\}$, respectively. The coefficient α that controls the BPR loss is selected from $\{0.001, 0.0005\}$ because evolution loss is much smaller, and the coefficient δ of regularization is set as $1e-2$. Last, we implement all the experiments with Pytorch in an NVIDIA 3090 GPU.

Table 3.2: Performance comparison of all models on Reddit, Wikipedia, and Foursquare in terms of MRR and Recall@10. The bold and underlined numbers are the best and second-best performances. The last row is the improvement compared with the second-best performance.

Models	Reddit		Wikipedia		Foursquare	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
Time-LSTM	0.387	0.573	0.247	0.342	0.135	0.279
DeepCoevolve	0.171	0.275	0.515	0.563	0.217	0.326
LatentCross	0.421	0.588	0.424	0.481	0.193	0.289
CTDNE	0.165	0.257	0.035	0.056	0.018	0.041
DGCF	0.629	0.795	<u>0.646</u>	<u>0.713</u>	0.368	<u>0.672</u>
FIRE	0.332	<u>0.814</u>	0.259	0.648	0.154	0.524
TREND	<u>0.647</u>	0.809	0.588	0.691	<u>0.370</u>	0.663
Our Proposed DGEL	**0.701**	**0.849**	**0.674**	**0.756**	**0.381**	**0.697**
Improvement	**8.3%**	**4.2%**	**4.3%**	**6.0%**	**2.9%**	**3.7%**

3.4.2 Performance Comparison

Now we report the performance evaluation of all dynamic methods to demonstrate the superiority of DGEL in Table 3.2. From the results, we summarize the following observations.

- **Overall Performance**. DGEL consistently outperforms all dynamic baseline methods in all datasets, especially yielding significant improvement on Reddit. This result verifies the accuracy and effectiveness of our proposed DGEL that endows the graph with the capability of evolving from three novel and appropriate perspectives with the re-scaling network. The average of interactions in Foursquare is the largest, and this dataset lacks features of interactions. In this case, it is hard to distinguish valuable interactions for dynamic evolution, and the evaluation metrics across all methods (including DGEL) are inferior to the other two datasets. Except for Reddit, the improvements on Recall are greater than MRR in Wikipedia and Foursquare. It may result from our joint loss function design that focuses on investigating positive user preferences and discovering high-relevant future items. To sum up, DGEL gains performance improvements for multi-round dynamic recommendations.

- **Generality and Flexibility**. The diversity of evaluation datasets varying in different sparsity degrees, availabilities of features, and various recommendation scenarios justifies the generality and flexibility of DGEL. Several factors determine that of DGEL: 1) The time-decay neighbor augmentation and symbiotic local structure learning, which capture the collaborative signals independently, only depends on the temporal interaction

Table 3.3: Ablation study on core modules of DGEL: Inherent Interaction Potential (IIP), Time-decay Neighbor Augmentation (TNA), Symbiotic Local Structure Learning (SLS), Re-scaling Enhancement Network (RENet) and Future Drifting (Drift). 'w/o' means 'without'.

Models	Reddit		Wikipedia		Foursquare	
	MRR	Recall	MRR	Recall	MRR	Recall
DGEL w/o IIP	0.675	0.824	0.637	0.723	0.354	0.683
DGEL w/o TNA	0.682	0.830	0.659	0.736	0.361	0.690
DGEL w/o SLS	0.689	0.827	0.654	0.730	0.346	0.675
DGEL w/o RENet	0.661	0.820	0.648	0.709	0.336	0.671
DGEL w/o Drfit	0.692	0.833	0.652	0.728	0.365	0.684
DGEL	**0.701**	**0.849**	**0.674**	**0.756**	**0.381**	**0.697**

sequence. 2) All components in dynamic representation learning are adopted with single-layer convolution to avoid over-fitting and over-smoothing caused by multiple aggregations, which also speeds up the updating process. 3) Our re-scaling enhancement network can adaptively explore the impact of various dynamic sub-embeddings.

- **Dynamic Graph Superiority**. In most cases from the comparison result, the dynamic recommendation models that apply dynamic graph networks (e.g., DGCF, TREND, and our proposed DGEL) achieve superiority over non-graph dynamic recommendation. We contribute this to the natural investigation of the graph on the implicit connectivity between users and items, which reveals the possibility of users linking with other items. Although the incremental dynamic recommendation FIRE brings the fast updating strategy on interval-based sessions, it is inadequate in real-time preference update leading to comparable results in terms of Recall ratio only. By contrast, our DGEL performs better as it focuses on fast-updating methods for the current interaction and accurately explores the users' rapid-changing preferences.

3.4.3 Ablation Study of DGEL

We examine the effects of the core modules in DGEL from 1) the three dynamic updating methods for node evolution; 2) the re-scaling enhancement network; and 3) the future drifting for generating recommendations. Table 3.3 lists the results.

- **Effect of Dynamic Updating Methods for Node Evolution**. We investigate the effect derived from our inherent interaction potential module, time-decay neighbor augmentation module, and symbiotic local structure learning module by removing each of them separately. We observe that lacking exploration of the interaction itself (e.g., corresponding user and item, interaction feature, etc.) severely degrades the performance of DGEL on Reddit and Wikipedia datasets, where both have rich information describing the interaction. Compared with the inherent interaction potential module, equipping with time-decay neighbor augmentation and symbiotic local structure learning modules improves the performance by capturing the collaborative signals, indicating the necessity of fast and efficient graph convolution for dynamic node evolution.

- **Effect of Re-scaling Enhancement Network**. As we can observe from the results, the performance of DGEL without the re-scaling strategy is almost the worst in all the ablation experiments, which verifies our assumption: the different scales of various embeddings that are conducted from their own space and learning method will harm the performance of graph-based recommendation, not to mention the scale distortion issues existing in both static and dynamic recommendation. Moreover, in our scenario, the three dynamic updating methods will repeat multi times for target users (or items) since users will interact with items continuously. Hence, DGEL requires the re-scaling network to adjust different dynamic embeddings to ensure the accuracy of node evolution.

- **Effect of Future Drifting for Predicting Recommendations**. We also investigate the effectiveness of applying future drifting in the recommendation system, which aims to drift user preferences to any future time and then discover the potential items that the user will consume. Specifically, we replace the drifted user preference embedding with the updated user embedding to predict future items directly. From the results, it is clear that on the basis of the other complete modules, introducing the future drifting component could bring a rise in the overall performance through its ability to exploit future preference. We can further conclude that additional future drifting will benefit our whole framework in predicting recommendations.

3.4.4 Research on Feasible Variants

In this section, we conduct experiments to analyze different feasible variants of our proposed DGEL: 1) sum and max pooling strategies for symbiotic local structure learning; 2) Feed-Forward Network, Additive Network, and basic L2 normalization for re-scaling enhancement network. and 3) linear weighted strategy for combining dynamic embeddings. The results are shown in Table 3.4.

- **Pooling Selection for Local Structure Learning**. For our symbiotic local structure learning, we replace the average pooling strategy with two options, i.e., max pooling and sum pooling, which are widely used to encode graph structure. From the results, we can observe that sum pooling is not competitive with max pooling and our adopted average

Table 3.4: Research on feasible variants of DGEL: pooling variants for symbiotic local structure learning, normalization variants for re-scaling enhancement network, fusion strategy for combining dynamic embeddings. We also report their corresponding ablation results (if they have).

Variants	Reddit		Wikipedia		Foursquare	
	MRR	Recall	MRR	Recall	MRR	Recall
Max	0.692	0.831	0.667	0.732	0.373	0.695
Sum	0.698	0.843	0.661	0.715	0.372	0.680
w/o SLS	0.689	0.827	0.654	0.730	0.346	0.675
Feed-Forward	0.696	0.840	0.668	0.731	0.351	0.687
Additive	0.682	0.830	0.670	0.738	0.364	0.679
L2	0.667	0.825	0.665	0.729	0.362	0.673
w/o RENet	0.661	0.820	0.648	0.709	0.336	0.671
Linear Weighted	0.688	0.835	0.669	0.734	0.377	0.692
DGEL	**0.701**	**0.849**	**0.674**	**0.756**	**0.381**	**0.697**

pooling as the data sparsity is over-low or over-high (e.g., Foursquare and Wikipedia). We contribute this to the continuous multi-time updating characteristic that leads to a fast increase of embedding value in the dynamic recommendation. On the other hand, compared with the ablation of local structure learning, any pooling selection for that brings positive performance improvement.

- **Various Re-scaling Network Forms**. The proposed re-scaling enhancement network (RENet) is one of the core components in our DGEL because it bridges the normalization with model learning adaptively and automatically. The observations are two-fold. First, the design of the re-scaling factor by a 2-layer MLP executes efficiently and scales the embeddings simply without complex learning paradigms such as Feed-Forward Network and Additive Network, gaining more generality for the whole framework. Second, adopting neural network based re-scaling methods can outperform the decoupled heuristic L2 normalization, not to mention that remove normalization for maintaining original embeddings. This further indicates the importance of bridging the normalization process with model learning.

- **Weighted Fusion of Dynamic Embeddings**. From the results, it is obvious that the linear weighted fusion of dynamic embeddings derived from different perspectives is still competitive across all the datasets, proving the power of the feed-forward layer to extract deep information. However, the concatenation of embeddings speeds up the learning process. Moreover, under the effect of our re-scaling enhancement network, the concatenation of embeddings maintains its own dynamic information from different perspectives and keeps consistency with RENet.

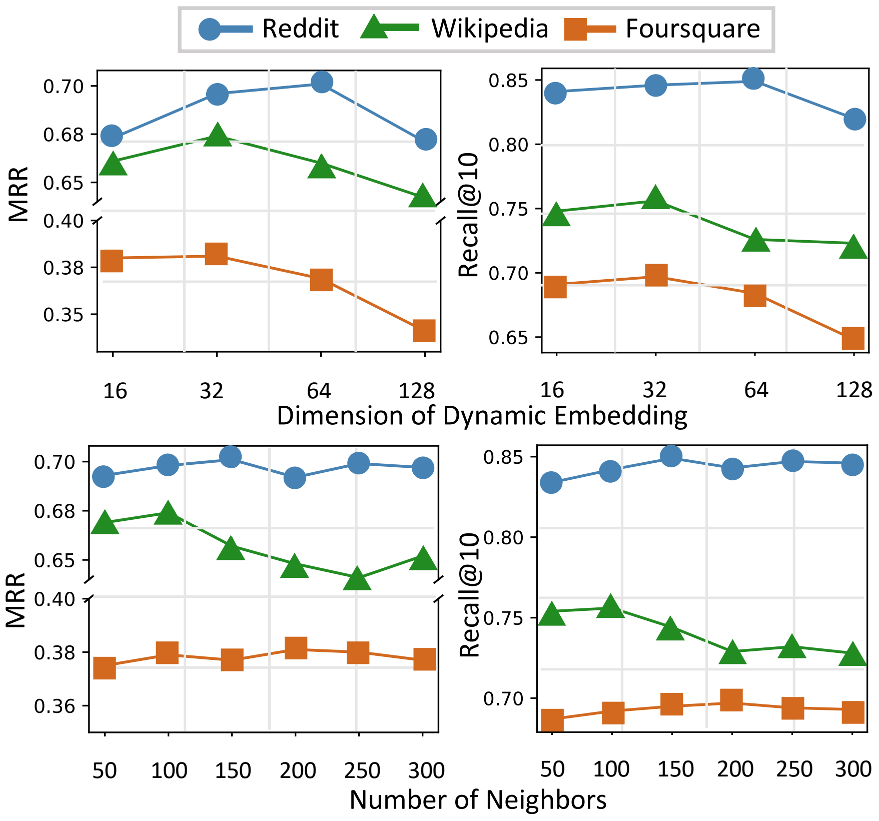


Figure 3.3: Sensitivity of hyperparameters: The dimension of dynamic embedding and the number of neighbors.

3.4.5 Hyperparameter Sensitivity Analysis

We conduct sensitivity analysis of two hyper-parameters on the performance of DGEL in Figure 3.3:

- **Dimension of Dynamic Embedding**. This determines the complexity of all the components and the length of concatenating dynamic embeddings derived from three perspectives. From the results, we have two significant observations. First, compared with the Wikipedia and Foursquare datasets, Reddit consists of a large amount of interactions and requires more dimensions to extract user and item features. Second, across all the datasets, overdimensioning (e.g., 128) severely degrades the performance, and no further improvements are observed due to saturation.

- **Number of Neighbors**. Both time-decay neighbor augmentation and symbiotic local structure learning are influenced by this parameter. Similarly, we still have two observations. On the one hand, the best number of neighbors for all the datasets is strongly related to their sparsity (e.g., 100 produces the best performance for the Wikipedia dataset because it has the least average interactions). On the other hand, the Foursquare dataset seems less sensitive to the increasing number of neighbors. We contribute this phenomenon to the data characteristic.

3.5 Summary of DGEL

In this chapter, we propose a novel framework, named Dynamic Graph Evolution Learning (DGEL), to realize the satisfying performance in the multi-round dynamic recommendation. The proposed DGEL can comprehensively and efficiently capture user and item dynamics over time from the perspectives of inherent interaction potential, time-decay augmentation, and symbiotic local structure learning. Meanwhile, DGEL performs the initial attempts to adaptively and automatically bridge the normalization of dynamic embeddings with the model learning process for the dynamic recommendation. Moreover, DGEL is designed to update user preferences in real time and adjust the recommendation results to satisfy users' new requirements as users continuously interact with the system. Extensive experiments on three real-world datasets demonstrate the superiority and effectiveness of our DGEL.

CHAPTER 4

TEMPORAL COLLABORATION-AWARE GRAPH LEARNING FOR RECOMMENDATION

Dynamic recommendation systems, where users interact with items continuously over time, have been widely deployed in real-world online streaming applications. To update representations dynamically, existing studies have investigated event-level and history-level dynamics by modeling the newly-arrived interactions and aggregating historical interactions, respectively. However, most of them directly learn the representation evolution as new interactions occur, without exploring the collaboration between the newly-arrived and historical interactions, thus failing to scrutinize whether those new interactions would benefit the evolution learning process when generating dynamic representations. Moreover, most of them model the two levels of dynamics independently, explicitly ignoring the inherent co-evolving correlation between them. In this work, we propose the **T**emporal **C**ollaboration-Aware **G**raph **C**o-Evolution Learning (TCGC) for the dynamic recommendation scenario. First, we explore the effectiveness of collaborative information and devise the collaboration-aware indicator to guide the evolution learning process. Second, we design a temporal co-evolving graph network, enabling our framework to capture the correlation between event and history dynamics. Third, we leverage the evolution task and recommendation task together for joint training. Extensive experiments on four public datasets demonstrate the superiority and effectiveness of our proposed TCGC.

The content in this chapter mainly focuses on addressing the research challenge **RC 2** and overcoming the limitation **EL 2** by proposing a novel model TCGC.

4.1 Brief Introduction of TCGC

Dynamic recommendation systems where users would interact with items continuously have been widely deployed in many real-world applications, e.g., online shopping, short-video streaming, social network, and real-time navigation [22, 30, 81, 103]. In such dynamic environments, the burst of interactions between users and items rapidly changes the short-term preference and the interactive pattern [96, 40, 16], leading to a drift in the status of the recommendation system. Therefore, learning users' evolving preferences and behavior patterns from continuous user-item interactions is crucial for generating accurate recommendations. Recently, temporal graph representation learning has demonstrated its superiority in modeling users' evolving preferences for dynamic recommendation [67, 85, 132, 101]. In contrast to static graph, temporal graph comprises temporal sequences of interactions, which can continuously model complex dynamics of node representations by incorporating sequential dependency and temporal information of interactions [44, 76, 1].

Existing temporal-graph based studies have adequately investigated user and item evolution from the event level [40, 141, 8, 107] and the history level [44, 114, 118, 94]. They represent two common update types by modeling newly-arrived interactions and aggregating historical interactions, respectively. The event level is to leverage the new information from the newly-arrived interactions, such as the new interaction features, node degrees, the interacting user and item embeddings, and also the time interval between previous and current timestamps [40, 107]. It projects the learned short-term information into embedding updates by the message-passing mechanism [141]. The history level aims to utilize historical interactions by aggregating information from historical neighbors [44, 127], thus exploring the long-term information for representation updates [94, 114]. Both of them are capable of letting user and item representations evolve over the graph, since they take full advantage of exploring high-order connectivity from the graph [149, 119] while injecting temporal information into the dynamic recommendation process.

Although these temporal graph-based methods have achieved remarkable success for dynamic recommendation, several significant problems still remain to be solved. The first one is the **"collaboration effectiveness" issue**. In the dynamic environment, as a new user-item interaction comes into the graph at the current timestamp, temporal graph networks (T-GNN) would utilize the information propagation to incrementally update the temporal representations for the triggered user and item nodes [109, 96, 100, 78]. The updated representations are learned from the collaboration between the newly-arrived interaction and the historical neighbors under the information propagation paradigm [44, 141, 94, 96].

However, most existing studies directly execute the temporal representation learning process, without carefully scrutinizing whether the captured collaborative effect from both the historical neighbors and newly-arrived interaction would benefit the temporal representation learning. In other words, every time the new interaction triggers the representation update, they would roughly inject the learned collaborative information to make the user (also item) representation evolve from the previous timestamp to the current timestamp [141, 103]. However, some essential questions naturally arise regarding *how well the newly-arrived interaction could match the temporal history*, and *how much information could be learned according to the current context of historical neighbors*. The similar issue also exists in static conventional graph networks [92, 21, 28]. Considering the temporal graph networks would frequently update node representations in the dynamic environment [76], controlling the effectiveness of continuous interactions for this issue tends to be more challenging and thorny, which encourages our work in this chapter.

One possible solution could be the graph attention mechanism since it is capable of measuring the importance of interactions in graphs and has been widely adopted in many existing graph-based models [109, 80, 7, 136, 48]. Nevertheless, the graph attention mechanism is designed from the perspective of embedding relationship only between the two interacting nodes [4, 80]. For example, if being employed in our dynamic recommendation scenario, the graph attention would compute the attention score just by the target user's embedding and the current interacted item's embedding, to denote the significance of their interaction. Other historical neighbors are only used to normalize the score into a probability. Consequently, the key score computation neglects the collaborative information from the historical neighbors and totally depends on the learned embedding space and attention parameters [23], which may face the over-fitting problem [83]. Most importantly, it cannot reveal the relationship between the newly-arrived interaction and historical neighbors, thus failing to answer the above-raised questions. Therefore, we aim to conduct exploration from the perspective of the collaboration between newly-arrived interaction and historical neighbors, to verify *the new interaction could match the temporal history and deliver beneficial information to the temporal representation learning*. This motivates us to design the non-parametric collaboration-aware indicators.

The second drawback of existing studies is the **"insufficient correlation mining" problem** [127]. Generally, *Event Modeling* and *History Replay* provide two representative views to form the dynamics for the T-GNN [85, 96, 102, 40, 67]. Specifically, the event-level dynamic indicates the short-term interactive information from the incoming user-item event itself [40, 67], while the history-level dynamic reveals the long-term impact of history on

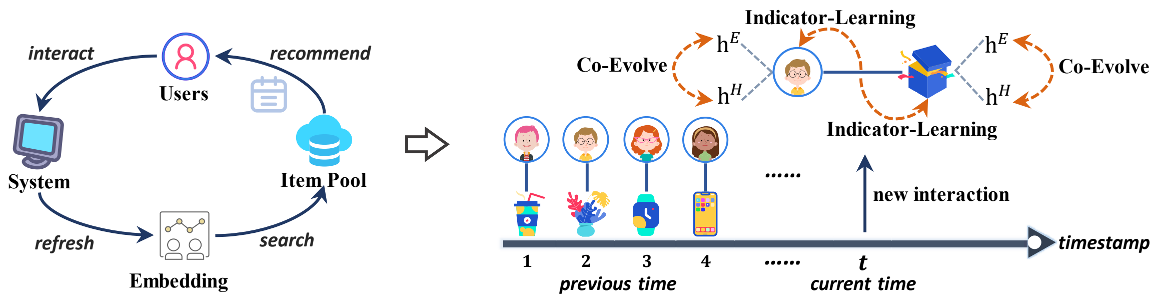


Figure 4.1: TCGC for dynamic recommendation: we guide the evolution learning of temporal representations by the temporal collaboration-aware indicator to generate event- and history-level dynamics $\mathbf{h}^E$ and $\mathbf{h}^H$ while allowing them to co-evolve with each other.

triggering new interactions [85, 96, 102]. They exhibit two different means to derive the temporal representations. Nonetheless, even though the multi-order structural correlation of the user-item graph has been discussed in [127] and [119], few works consider the inherent co-evolving correlation between the event-level and history-level dynamics. In other words, most studies model these two dynamics independently, explicitly neglecting the fact that they complement each other and also they are coherent in the meantime. However, at the current timestamp, the newly-arrived interaction between the user and item is naturally the joint result of their neighbors, and the neighbors would be further affected by absorbing the incoming interaction's information. Thus, another question naturally arises regarding *how the event-level dynamic and history-level dynamic co-evolve with each other*, which motivates us to further explore the co-evolving correlation between them and build our temporal co-evolution learning framework.

In summary, to tackle the above-mentioned problems, as shown in Figure 4.1, we propose a novel temporal graph network for dynamic recommendation, namely **T**emporal **C**ollaboration-Aware **G**raph **C**o-Evolution Learning (TCGC). While the DGEL proposed in Chapter 3 investigates the user/item dynamics from multiple perspectives, it cannot verify the effectiveness of the interactions. In contrast, our new TCGC in this chapter takes the data quality into consideration to achieve more accurate temporal learning. Firstly, we justify the collaborative effectiveness by analyzing how the new interaction would help to update the user's preferences over items. Furthermore, we design a temporal collaboration-aware indicator that indicates the effectiveness of collaboration over the temporal graph, to dynamically guide the temporal evolution learning for both user and item representations (section 4.3.1). Different from the graph attention [80, 109], the role of collaboration-aware indicator is to quantify the evolution benefit from the newly-arrived interaction before any T-GNN operation. Meanwhile, the collaboration-aware indicator can be integrated with existing dynamic recommendation models to improve their performance from the data-centric perspective. Next, we construct the temporal co-evolving graph network to explore the inherent correlation between event-level and history-level dynamics (section 4.3.2), strengthening the expressiveness of these two dynamic representations by forcing them to adaptively complement each other. Last, we jointly train our TCGC by leveraging the evolution and recommendation tasks to maintain both the temporal evolution and system performance simultaneously (section 4.3.3). The main contributions of this work can be summarized as follows:

- To the best of our knowledge, we are the first to demystify the effectiveness of collaborative information triggered by newly-arrived interactions for the dynamic recommendation and propose the temporal collaboration-aware indicator to dynamically guide the temporal representation learning process. This design could be smoothly and efficiently integrated into other dynamic graph-based recommendation models.

- We design a novel temporal co-evolving graph network with the support of collaboration-aware indicators to adaptively bridge the potential correlation between the event-level and history-level dynamics.

- We leverage the evolution task and recommendation task together for joint training to improve the temporal evolution and system performance simultaneously.

- We conduct extensive experiments on four real-world datasets, and the experimental results demonstrate both the superiority and effectiveness of our proposed TCGC.

4.2 Preliminaries

Suppose our scenario involves M users and N items with the user set $\mathcal{U}$ and item set $\mathcal{V}$. Let $\mathcal{E} = \{e_{ij}^t | i \in \mathcal{U}, j \in \mathcal{V}\}$ be the stream of observed events, where e_{ij}^t corresponds to an interaction between user i and item j at timestamp t. We denote the temporal representations of user i and item j as $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$, respectively. Following prior works, we define a temporal snapshot of a graph as $\mathcal{G}^t = \{\mathcal{U}, \mathcal{V}, \mathcal{E}^t\}$, where $\mathcal{E}^t = \{e_{ij}^\tau \in \mathcal{E} | \tau \leq t\}$. Therefore, the temporal graph set is formalized as $\mathcal{G} = \{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^{|\mathcal{E}|}\}$. The timestamped stream of events indicates the dynamic formalization of the temporal graph.

Recap Temporal Graph Neural Network. Given an interaction between user i and item j forming the $\mathcal{G}^t$ at current timestamp t, we use the following temporal GNN [96] to learn the evolution of them:

$$\mathbf{h}_i^t = \sigma(\mathbf{W}_1^u \mathbf{h}_i^{t^-} + \sum_{x \in \mathcal{H}_i^t} \mathbf{W}_2^u \kappa(x, \mathcal{H}_i^t) \mathbf{h}_x^{t^-}), \tag{4.1}$$

$$\mathbf{h}_j^t = \sigma(\mathbf{W}_1^v \mathbf{h}_j^{t^-} + \sum_{z \in \mathcal{H}_j^t} \mathbf{W}_2^v \kappa(z, \mathcal{H}_j^t) \mathbf{h}_z^{t^-}), \tag{4.2}$$

where σ is the activation function, i.e., LeakyReLU, and it is applied throughout the chapter. $\mathbf{W}_1^u$, $\mathbf{W}_1^v$, $\mathbf{W}_2^u$, and $\mathbf{W}_2^v$ are learnable parameters. x is the historical interacted item of user i and $\mathcal{H}_i^t$ denotes i's historical neighbor set at current timestamp t. z is the historical interacted user of item j and $\mathcal{H}_j^t$ represents j's historical neighbor set at timestamp t. $\mathcal{H}_i^t$ and $\mathcal{H}_j^t$ are continuously changing over time. $\kappa(\cdot)$ returns the time-aware decay effect based on the timestamp of each historical interaction. It assigns each neighbor with a weight for aggregation. In this chapter, the $\kappa(\cdot)$ is determined as softmax function [96] (see Eq. 4.10) because (1) it efficiently converts the time interval into the probability distribution where a larger interval would be assigned with a smaller probability while a smaller interval would get a larger probability; (2) the sum of this probability distribution equals to 1, ensuring the aggregation is the weighted average and thus avoiding embedding exposure; and (3) it is a non-parametric function without additional trainable parameters. Thus, it is suitable for our dynamic scenario to model the time-decay information in temporal GNN. Similar to the conventional graph network, we adopt the one-hop neighbor set. However, the higher-order information can still be propagated because in the dynamic environment, the temporal graph network keeps running over time and the representation learning could continuously walk over the graph [94].

Dynamic Recommendation Task. Traditional recommendation tasks follow the one-time recommendation-evaluation fashion where each user only receives one fixed recommendation list during the test stage [30]. In contrast, the dynamic recommendation task, which keeps in line with real-world online applications and follows the multi-round recommendation paradigm, aims to refresh recommendation results efficiently and dynamically after obtaining real-time feedback from users at each timestamp [40, 141, 44]. By this means, the recommendation system can generate multiple different lists for the target user since he (or she) may access the system multiple times. Here, we define the multi-round recommendation task as follows:

$$\mathbf{h}_i^t, \mathbf{h}_j^t \leftarrow \mathcal{F}_g(\mathbf{h}_i^{t^-}, \mathbf{h}_j^{t^-}), \tag{4.3}$$

$$\text{Rec}_i^{t^+} = \{v | v \in \mathcal{V}, \arg\overset{k}{\min_v} \left\| \mathbf{P}^{t^+} - [\mathbf{h}_v^t, \bar{\mathbf{h}}_v] \right\|_2\}, \tag{4.4}$$

where $\mathscr{F}_g$ is the temporal GNN model and k is the length of recommendation list. $[\cdot]$ is concatenation. In addition to temporal representations, we also set the static representations (e.g., one-hot ID information) $\bar{\mathbf{h}}_i$ and $\bar{\mathbf{h}}_v$ for user i and item v. When conducting the recommendation stage, we concatenate the learned temporal representations with static representations to denote the final representations of the user and item. Our duty is to learn the accurate temporal representations $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$. Then, $\mathbf{P}^{t^+}$ is the predicted item representation for the user's next interaction at the next timestamp t^+. Based on the future prediction $\mathbf{P}^{t^+}$, we would generate the recommendation list by returning the Top-k items whose representations are nearest to the $\mathbf{P}^{t^+}$.

Hence, after learning the evolution of user i and item j at current time t by Eq. 4.3, the system will update their temporal representations and prepare a new recommendation list $\text{Rec}_i^{t^+}$ for user i at future time t^+ by Eq. 4.4. When user i interacts with the system at future time t^+, we would repeat the Eq. 4.3 and Eq. 4.4. Consequently, we deploy the dynamic recommendation. It is worth noting that the evaluation is also conducted in a real-time fashion, where we would evaluate the refreshed recommendation list with user actual interaction at each timestamp. A simple toy example is shown in the left part of Figure 4.1. The temporal representations are evolving and the recommendation lists are being refreshed continuously. This process will keep running until the system is shut down.

4.3 TCGC Model Design

In this section, we introduce the Temporal Collaboration-Aware Graph Co-Evolution Learning (TCGC) and show the overall architecture in Figure 4.2. First, we analyze the newly-arrived interaction's collaborative effectiveness for the temporal GNN and propose the temporal collaboration-aware indicators to adaptively guide the evolution learning of temporal representations for both user and item (section 4.3.1). Second, with the effective constraints of the collaboration-aware indicators, we design a new temporal co-evolving graph network to capture the inherent correlation between event-level and history-level dynamics (section 4.3.2). The two designs are our core contributions. Finally, we conduct predictions and leverage both evolution loss and recommendation loss for joint training to optimize our TCGC (section 4.3.3).

4.3.1 Temporal Collaboration-Aware Indicator

Given historical user-item interactions, the key is to utilize the collaborative effect to predict future interactions between users and items [54, 89]. The power of collaborative information has been fully mined by existing graph-based recommendation models. The collaboration between newly-arrived interaction and historical neighbors would jointly generate a new temporal representation for the triggered user and item. Nevertheless, existing methods face **"collaboration effectiveness" issue** where they directly learn the evolution of the temporal representation without carefully scrutinizing whether the incoming interaction would truly benefit the evolution learning process. As a consequence, the risk of deriving uninformative or even harmful representations tends to increase significantly.

To better explain the collaboration derived by the newly-arrived interaction, we take the standard item score function $y^t_{ij} = \mathbf{h}^t_i \cdot \mathbf{h}^t_j$ as an example. For most dynamic recommendation tasks, generating a recommendation list at time t is based on the ranking score y^t_{ij} by the product of the user and item. If we inject temporal GNN (omitting activation) in Eq. 4.1 and Eq. 4.2 into y^t_{ij}, it would be:

$$\begin{aligned} y^t_{ij} &= (\mathbf{W}^u_1\mathbf{h}^{t^-}_i + \sum_{x\in\mathcal{H}^t_i}\mathbf{W}^u_2\kappa(x,\mathcal{H}^t_i)\mathbf{h}^{t^-}_x)(\mathbf{W}^v_1\mathbf{h}^{t^-}_j + \sum_{z\in\mathcal{H}^t_j}\mathbf{W}^v_2\kappa(z,\mathcal{H}^t_j)\mathbf{h}^{t^-}_z) \\ &= \mathbf{W}^u_1\mathbf{h}^{t^-}_i \cdot \mathbf{W}^v_1\mathbf{h}^{t^-}_j + \sum_{z\in\mathcal{H}^t_j}\mathbf{W}^v_2\kappa(z,\mathcal{H}^t_j)\mathbf{h}^{t^-}_z \cdot \sum_{x\in\mathcal{H}^t_i}\mathbf{W}^u_2\kappa(x,\mathcal{H}^t_i)\mathbf{h}^{t^-}_x \\ &+ \mathbf{W}^u_1\mathbf{h}^{t^-}_i \sum_{z\in\mathcal{H}^t_j}\mathbf{W}^v_2\kappa(z,\mathcal{H}^t_j)\mathbf{h}^{t^-}_z + \mathbf{W}^v_1\mathbf{h}^{t^-}_j \sum_{x\in\mathcal{H}^t_i}\mathbf{W}^u_2\kappa(x,\mathcal{H}^t_i)\mathbf{h}^{t^-}_x . \end{aligned} \tag{4.5}$$

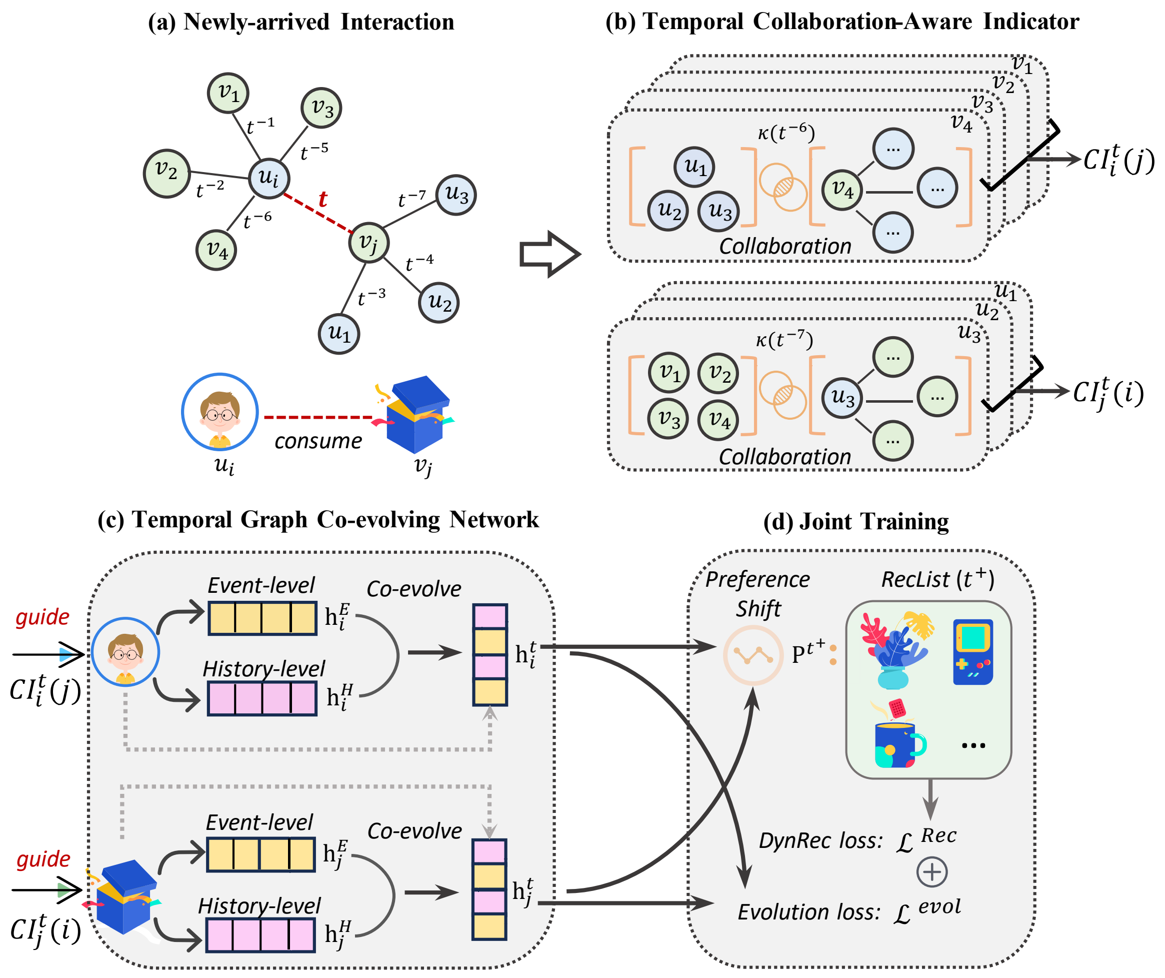
We focus on the collaboration-aware parts integrating the representations of user i and item j. Thus, Eq. 4.5 changes to:

$$\hat{y}^t_{ij} = \mathbf{W}^u_1\mathbf{h}^{t^-}_i \sum_{z\in\mathcal{H}^t_j}\mathbf{W}^v_2\kappa(z,\mathcal{H}^t_j)\mathbf{h}^{t^-}_z + \mathbf{W}^v_1\mathbf{h}^{t^-}_j \sum_{x\in\mathcal{H}^t_i}\mathbf{W}^u_2\kappa(x,\mathcal{H}^t_i)\mathbf{h}^{t^-}_x . \tag{4.6}$$

We let $\mathfrak{C}(\cdot)$ denote the combination of $\mathbf{h}^{t^-}_i$ and each $\mathbf{W}^v_2\kappa(z,\mathcal{H}^t_j)\mathbf{h}^{t^-}_z$ (same operation on $\mathbf{h}^{t^-}_j$). Then the Eq. 4.6 can be replaced as:

$$\hat{y}^t_{ij} = \underbrace{\mathbf{W}^u_1 \sum_{z\in\mathcal{H}^t_j}\mathfrak{C}(\mathbf{h}^{t^-}_i, \mathbf{h}^{t^-}_z)}_{i \text{ and neighbors of } j} + \underbrace{\mathbf{W}^v_1 \sum_{x\in\mathcal{H}^t_i}\mathfrak{C}(\mathbf{h}^{t^-}_j, \mathbf{h}^{t^-}_x)}_{j \text{ and neighbors of } i} . \tag{4.7}$$

Thus, based on Eq. 4.7, we have the following observations to answer the collaborative effect captured by temporal GNN for the interaction between user i and item j at time t:



(a) Newly-arrived Interaction
(b) Temporal Collaboration-Aware Indicator
(c) Temporal Graph Co-evolving Network
(d) Joint Training
consume
Collaboration
Event-level
History-level
Co-evolve
guide
Preference Shift
RecList (t^+)
DynRec loss: $\mathcal{L}^{Rec}$
Evolution loss: $\mathcal{L}^{evol}$
$CI_i^t(j)$
$CI_j^t(i)$

Figure 4.2: The overall framework of TCGC for the dynamic recommendation system. There are three main components: (a) temporal collaboration-aware indicator learning, (b) temporal graph co-evolving network, and (c) multi-task joint training.

- **Temporal Neighbor Contribution**. Each neighbor of both user and item would influence the learning process by devoting its time-aware impact to current loss computation through graph collaborative filtering.

- **Intersectional Collaboration Paradigm**. The temporal collaborative effect is intersectional where user i's impact is on item j's neighbors while item j's impact is on user i's neighbors. The informative gain of current interaction is determined by how ideally they match the neighbors of each other.

Therefore, in order to guarantee the accuracy of learning the temporal representations for user i and item j, it is necessary to justify whether the current interaction at time t is beneficial to the evolution process. Otherwise, uninformative or even harmful representations could be learned. Consequently, they may damage the performance of the system by bringing inappropriate loss. This consideration is to protect both the graph evolution and system performance. Inspired by the work for static recommendation [92], we propose the temporal Collaboration-Aware Indicator (CI) to guide the learning process for dynamic recommendation. Specifically, at current time t, we generate CI for user i and item j, respectively:

$$CI_i^t(j) = \sum_{x \in \mathcal{H}_i^t} \kappa(x, \mathcal{H}_i^t) f(\{\mathcal{H}_x^t \cup x\}, \{\mathcal{H}_j^t \cup j\}), \tag{4.8}$$

$$CI_j^t(i) = \sum_{z \in \mathcal{H}_j^t} \kappa(z, \mathcal{H}_j^t) f(\{\mathcal{H}_z^t \cup z\}, \{\mathcal{H}_i^t \cup i\}), \tag{4.9}$$

$$\kappa(x, \mathcal{H}_i^t) = \frac{e^{-(t-t(x))}}{\sum_{x' \in \mathcal{H}_i^t} e^{-(t-t(x'))}}, \kappa(z, \mathcal{H}_j^t) = \frac{e^{-(t-t(z))}}{\sum_{z' \in \mathcal{H}_j^t} e^{-(t-t(z'))}}, \tag{4.10}$$

where $\kappa(\cdot)$ returns the time-decay weight and $t(x)$ $(t(z))$ represents the timestamp of the historical neighbor x (z). The function $f(\cdot)$ computes the similarity between two sets. Here, we let $f(A, B)$ be Jaccard similarity. $CI_i^t(j)$ explores the evolution benefit from item j for the target user i while $CI_j^t(i)$ examines the evolution benefit from user i for the target item j, which follow the previous observations. In other words, we assume that if item j could bring more informative gain for user i at the current time, j's neighbors would pose more commons to the neighbors of each neighbor of i, and vice versa. Another specific case is that, if we use binary 0-1 to denote CI, it would degenerate into the interaction-based denoising problem [51]. We have to recognize that there are many similarity-measurement options for the function $f(\cdot)$, but determining the best similarity measurement is out of scope in this work. Eventually, the temporal collaboration indicators would be applied to our proposed temporal co-evolving graph network (see section 4.3.2).

The reasons we do not directly compute the common nodes between user i and item j are: 1) the user-item interaction graph is a bipartite graph; 2) The intersectional collaboration works between user i and item j's neighbors, and between item j and user i's neighbors. Additionally, it is worth noting that we adopt a one-hop neighbor setting throughout this chapter to guarantee efficiency for the dynamic recommendation scenario.

4.3.2 Temporal Co-evolving Graph Network

Event modeling and *history replay* are two core perspectives to form the dynamics in temporal graph network [85, 96, 102, 40, 67]. Event-level dynamic reveals the explicit short-term information from the incoming user-item interaction itself [40, 67], while the history-level dynamic explains the long-term impact of historical items on the user's current decision under the dynamic environment [85, 96, 102]. However, existing studies model the two kinds of dynamics independently, neglecting the inherent co-evolving correlation between them and thereby making the whole system suboptimal. Thus, our second objective is to bridge the co-evolution learning between them and exploit the co-evolving correlation via our proposed temporal co-evolving graph network.

Following the works in [141, 67, 40], we use two recurrent message-passing channels to update event dynamics for users and items, respectively, which are shared across all timestamps during the multi-round process. Meanwhile, we integrate the temporal collaboration-aware indicators derived by section 4.3.1 into the event dynamics to guide the evolution learning. More formally,

$$\mathbf{h}^E_{i,t} = \sigma(\mathbf{W}^u_1\mathbf{h}^{t^-}_i + CI^t_i(j)(\mathbf{W}^u_2\mathbf{h}^{t^-}_j + \mathbf{W}^u_3 x^t_{ij}) + \mathbf{W}^u_4 \triangle(t, t^-)), \tag{4.11}$$

$$\mathbf{h}^E_{j,t} = \sigma(\mathbf{W}^v_1\mathbf{h}^{t^-}_j + \underbrace{CI^t_j(i)(\mathbf{W}^v_2\mathbf{h}^{t^-}_i + \mathbf{W}^v_3 x^t_{ij})}_{\text{guide learning from each other}} + \mathbf{W}^v_4 \triangle(t, t^-)), \tag{4.12}$$

where $\mathbf{h}^E_{i,t}$ and $\mathbf{h}^E_{j,t}$ are temporal event-level representations for user i and item j at time t. x^t_{ij} is the feature describing the new interaction and $\triangle(\cdot)$ computes the interval between two timestamps. $CI^t_i(j)$ determines how much information would be learned from item j's event for the user i, while $CI^t_j(i)$ works for the item j.

The temporal GNNs in Eq. 4.1 and Eq. 4.2 exactly encode the history-level dynamics by aggregating historical neighbors. In the same way, we inject the temporal collaboration-aware indicators into the temporal GNN to encode accurate aggregations for the user and item, which also helps us guide the learning process. Although the temporal collaboration-aware indicators are derived from the perspective of the newly-arrived interaction, the aggregation of historical neighbors would be triggered by the newly-arrived interaction to update new temporal representations. Also, the computation of temporal indicators is still based on the history. Thus, the temporal collaboration-aware indicators do influence the history-level dynamics. The temporal collaboration-aware GNNs are shown as follows:

$$\mathbf{h}^H_{i,t} = \sigma(\mathbf{W}^u_5\mathbf{h}^{t^-}_i + CI^t_i(j)(\sum_{x\in\mathcal{H}^t_i} \mathbf{W}^u_6\kappa(x, \mathcal{H}^t_i)\mathbf{h}^{t^-}_x)), \tag{4.13}$$

$$\mathbf{h}^H_{j,t} = \sigma(\mathbf{W}^v_5\mathbf{h}^{t^-}_j + \underbrace{CI^t_j(i)(\sum_{z\in\mathcal{H}^t_j} \mathbf{W}^v_6\kappa(z, \mathcal{H}^t_j)\mathbf{h}^{t^-}_z)}_{\text{guide learning from history}}), \tag{4.14}$$

where $\mathbf{h}^{H}_{i,t}$ and $\mathbf{h}^{H}_{j,t}$ are temporal history-level representations for user i and item j at time t. Here, $CI^{t}_{i}(j)$ and $CI^{t}_{j}(i)$ refine the temporal GNNs for user and item, respectively, and determine how much information could be learned from neighbor aggregations to derive history-level representations more precisely. The function $\kappa(\cdot)$ returns the time-decay weight and $t(x)$ $(t(z))$ represents the timestamp of the historical neighbor x (z). We still adopt the softmax function to compute the time-decay weight, which could be referred to the Eq.4.10.

The two kinds of dynamics complement each other and are coherent in the meantime. On the one hand, interaction event naturally originates from the long-term impact of the historical neighbors [119]. On the other hand, the historical neighbors would be further updated when they accept the incoming short-term event as a member of them [127]. Thus, the inherent correlation exists between the two kinds of dynamics, encouraging us to leverage co-evolution for them. Specifically, we let them perceive each other's 'existence' by the element-wise attentional mechanism. Task user i as example:

$$\alpha^{t}_{u} = \text{softmax}(\sigma(\mathbf{W}^{\alpha}[\mathbf{h}^{E}_{i,t}, \mathbf{h}^{H}_{i,t}])), \tag{4.15}$$

$$\beta^{t}_{u} = \text{softmax}(\sigma(\mathbf{W}^{\beta}[\mathbf{h}^{H}_{i,t}, \mathbf{h}^{E}_{i,t}])), \tag{4.16}$$

$$\mathbf{h}^{E*}_{i,t} = \mathbf{h}^{E}_{i,t} \odot \alpha^{t}_{u}, \mathbf{h}^{H*}_{i,t} = \mathbf{h}^{H}_{i,t} \odot \beta^{t}_{u}, \tag{4.17}$$

where α^{t}_{u} and β^{t}_{u} are attentional vectors for event dynamic and history dynamic, respectively. $[\cdot]$ is concatenation operation and $\odot$ refers to element-wise product. By this means, the two temporal representations are co-evolving with each other. This endows them with the capability of learning representation information from each other adaptively and further strengthens the expressiveness of representations. We still employ the activation σ (LeakyReLU) in Eq. 4.15 and Eq. 4.16 because (1) we expect the results derived by the co-evolving network could be distinguishable within the range of positive numbers to investigate how well they could co-evolve with other; (2) Within the range of negative numbers, it is unnecessary to extremely distinguish them since all the negative numbers indicate they may not have potential co-evolving correlation. Similarly, we adopt the same co-evolving design for item j and get $\mathbf{h}^{E*}_{j,t}$ and $\mathbf{h}^{H*}_{j,t}$:

$$\alpha^{t}_{v} = \text{softmax}(\sigma(\mathbf{W}^{\alpha}[\mathbf{h}^{E}_{j,t}, \mathbf{h}^{H}_{j,t}])), \tag{4.18}$$

$$\beta^{t}_{v} = \text{softmax}(\sigma(\mathbf{W}^{\beta}[\mathbf{h}^{H}_{j,t}, \mathbf{h}^{E}_{j,t}])), \tag{4.19}$$

$$\mathbf{h}_{j,t}^{E*} = \mathbf{h}_{j,t}^{E} \odot \alpha_v^t, \mathbf{h}_{j,t}^{H*} = \mathbf{h}_{j,t}^{H} \odot \beta_v^t. \tag{4.20}$$

So far, we have constructed the temporal collaboration-aware indicators to guide the evolution learning of temporal representations and proposed the temporal co-evolving graph network to capture the inherent correlation between event and history dynamics. Now, we are ready to obtain the final temporal representations for user i and item j, which combines the event-level and history-level dynamics:

$$\mathbf{h}_i^t = \sigma(\mathbf{W}_7^u[\mathbf{h}_{i,t}^{E*}, \mathbf{h}_{i,t}^{H*}] + \mathbf{b}^u), \tag{4.21}$$

$$\mathbf{h}_j^t = \sigma(\mathbf{W}_7^v[\mathbf{h}_{j,t}^{E*}, \mathbf{h}_{j,t}^{E*}] + \mathbf{b}^v). \tag{4.22}$$

Next, we will provide our dynamic recommendation list by utilizing the learned temporal representations above.

4.3.3 Training for Dynamic Recommendation

In order to refresh the recommendation result when user i accesses our system at future time t^+, we use the *preference shift* [44, 30], which is based on the fact that user preference may drift as the new generation in the future. Such shift is determined by the interval between the future timestamp t^+ and the current timestamp t. We formalize *preference shift* as follows:

$$\mathbf{h}_i^{t^+} = (1 + \mathbf{w}_1^{\text{sft}} \triangle (t, t^+)) \odot \mathbf{h}_i^t, \tag{4.23}$$

where $\odot$ is element-wise product operation and $\triangle$ denotes the interval. $\mathbf{h}_i^{t^+}$ indicates the user i's future representation at t^+. Hence, we can make prediction at t^+ by the preference shift of user i and the latest representation of item j (user i's previous interaction at t):

$$\mathbf{P}^{t^+} = (\mathbf{W}_2^{\text{sft}}\mathbf{h}_i^{t^+} + \mathbf{W}_3^{\text{sft}}\bar{\mathbf{h}}_i) + (\mathbf{W}_4^{\text{sft}}\mathbf{h}_j^{t^+} + \mathbf{W}_5^{\text{sft}}\bar{\mathbf{h}}_j) + \mathbf{b}^{\text{sft}}, \tag{4.24}$$

where $\mathbf{P}^{t^+}$ is the predicted item representation that the user may interact with in the future. $\bar{\mathbf{h}}_i$ and $\bar{\mathbf{h}}_j$ are static embeddings, e.g., ID information. The predicted $\mathbf{P}^{t^+}$ comprehensively considers both temporal and static representations, which is more practical for real-world platforms. The new recommendation $Rec_i^{t^+}$ list at t^+ would be prepared by returning top-k items whose representations are nearest to $\mathbf{P}^{t^+}$. When it comes to t^+, we collect the new interaction from user i and recall our proposed TCGC to continue the temporal representation learning. This process would keep running until the system is shut down, as introduced in Section 4.2.

Recommendation Loss. From the recommendation inferring above, it is clear that we would continuously predict the potential item representation at each timestamp that the user may interact with, so as to generate continuous recommendation lists based on the predicted item representation in real-time. Therefore, we hope that at each timestamp, the predicted item representation could be closer to the actual item representation that the user truly interacts with. It directly determines the quality of the dynamic recommendation. Here, we formally define the recommendation loss as:

$$\mathscr{L}^{\text{rec}} = \sum_{e_{ij}^{t} \in \mathscr{E}} \|\mathbf{P}^{t^+} - \mathbf{P}_{j^*}^{t^+}\|_2, \tag{4.25}$$

where $\mathbf{P}_{j^*}^{t^+}$ is the ground-truth item embedding that user i actually interacts with at t^+. In addition, $\mathbf{P}_{j^*}^{t^+} = [\mathbf{h}_{j^*}^{t^+}, \bar{\mathbf{h}}_{j^*}]$.

Evolution Loss. Real-time representation update (for both user and item) is the basis of predicting real-time interaction. However, the rapid-evolving of representations may face the over-fitting problem [44, 84] since every time the user (e.g., i) interacts with an item (e.g., j), we would immediately let their representations evolve from the previous timestamp to the current timestamp. Especially when the time interval between timestamps is smaller, the update would be more frequent. Thus, to avoid the performance degradation caused by the over-fitting problem and prevent embedding space from frantically evolving over time, we define the evolution loss as follows:

$$\mathscr{L}^{\text{evol}} = \sum_{e_{ij}^{t} \in \mathscr{E}} \|\mathbf{h}_i^t - \mathbf{h}_i^{t^-}\|_2 + \|\mathbf{h}_j^t - \mathbf{h}_j^{t^-}\|_2, \tag{4.26}$$

Eventually, the overall loss of our TCGC is:

$$\mathscr{L}^{\text{all}} = \mathscr{L}^{\text{rec}} + \mathscr{L}^{\text{evol}} + \varphi\|\Theta\|_2, \tag{4.27}$$

where the last term is the regularization over the parameter set Θ of our model. φ controls its impact. On one hand, we hope the predicted item representation could be closer to the ground-truth item representations for constructing a better recommendation system. On the other hand, we hope the same user (or item) representations between timestamps could be similar for smoothing and reliable evolution. We utilize them together to train our model. Nevertheless, parallelized training in a dynamic recommendation task is not as easy as a conventional recommendation task. For example, in a training batch, if it involves more than one interaction from the same user i, there will be a conflict in updating its temporal representation. One possible solution is to sequentially train the model based on each individual interaction. However, it would cause a huge computational cost and a serious over-fitting problem to the training.

Thus, inspired by DGCF [44], we adopt the **temporal-batching method** to overcome these challenges and conduct temporal graph learning. In practice, we assign each interaction e_{ij}^t to a particular batch B_l, where $l \in [1, \mathscr{E}]$. We first initialize $|\mathscr{E}|$ empty batches (the worst case is that each batch only contains one interaction). We iterate through the interaction stream and add each interaction to a batch B_l. Let $maxBatch(s, r)$ be the batch with the largest index that has an interaction involving a node s (user or item) till the interaction e^r. Then, the next interaction e^{r+1} (i.e., between user i and item j), is assigned to the batch with index = $max(1 + maxBatch(i, r), 1 + maxBatch(j, r))$. This temporal-batching method satisfies two requirements: (1) no common user or item in the same batch, such that all interactions in each batch could be processed in parallel, and (2) processing the batches in increasing order of their index maintains the temporal ordering of the interactions.

```
Algorithm 1: Training of TCGC (one epoch)
   Input: $\mathscr{U}$, $\mathscr{V}$, $\mathscr{E}^{tr}$, $d$, $l^{mx}$, temporal batch $B = \emptyset$
   Output: trained $\Theta$ of this epoch
1  Randomly initialize the parameter set $\Theta$ of TCGC;
2  Uniformly initialize all representations with dimension $d$;
3  foreach event $e_{ij}^t \in \mathscr{E}^{tr}$ do
4      Update:
5      add user $i$ to $\mathscr{H}_j^t$ and item $j$ to $\mathscr{H}_i^t$ constrained by $l^{mx}$;
6      compute collaboration indicators $CI_i^t(j)$ and $CI_j^t(i)$ by Eq. 4.8 and Eq. 4.9;
7      update event-level $\mathbf{h}_{i,t}^E$ and $\mathbf{h}_{j,t}^E$ by Eq. 4.11 and Eq. 4.12;
8      update history-level $\mathbf{h}_{i,t}^H$ and $\mathbf{h}_{j,t}^H$ by Eq. 4.13 and Eq. 4.14;
9      explore co-evolving correlation between $\mathbf{h}_{i,t}^E$ and $\mathbf{h}_{i,t}^H$, and between $\mathbf{h}_{j,t}^E$ and $\mathbf{h}_{j,t}^H$ by Eq.
         4.17 to Eq. 4.20;
10     update final temporal $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ by Eq. 4.21 and Eq. 4.22;
11     Temporal-batch Optimization:
12     if $i$ and $j \notin B$ then
13         add $e_{ij}^t$ to $B$ ;
14     else
15         obtain the recommendation loss of $B$ by Eq. 4.25;
16         obtain the evolution loss of $B$ by Eq. 4.26;
17         optimize the $\Theta$ of TCGC based on Eq. 4.27;
18         let $B = \{e_{ij}^t\}$;
19     end
20 end
```

Algorithm 2: Inferring of TCGC

```
Input: $\mathcal{U}, \mathcal{V}, \mathcal{E}^{ts}, d, l^{mx}, k$, trained $\Theta$,
Output: Recall@10 and MRR
1  Initialize performance set $Res = \emptyset$;
2  foreach event $e_{ij}^{t} \in \mathcal{E}^{ts}$ do
3      Recommendation:
4      fetch the $h_i^{t^-}$ from previous time;
5      generate current top-$k$ recommendation list for $u_i$ based on Eq. 4.23 and Eq. 4.24;
6      add current Recall@10 and MRR to $Res$ by comparing $e_{ij}^{t}$;
7      Update:
8      add user $i$ to $\mathcal{H}_j^t$ and item $j$ to $\mathcal{H}_i^t$ constrained by $l^{mx}$;
9      compute collaboration indicators $CI_i^t(j)$ and $CI_j^t(i)$ by Eq. 4.8 and Eq. 4.9;
10     update event-level $\mathbf{h}_{i,t}^E$ and $\mathbf{h}_{j,t}^E$ by Eq 4.11 and Eq. 4.12;
11     update history-level $\mathbf{h}_{i,t}^H$ and $\mathbf{h}_{j,t}^H$ by Eq 4.13 and Eq. 4.14;
12     explore co-evolving correlation between $\mathbf{h}_{i,t}^E$ and $\mathbf{h}_{i,t}^H$, and between $\mathbf{h}_{j,t}^E$ and $\mathbf{h}_{j,t}^H$ by Eq.
         4.17 to Eq. 4.20;
13     update final temporal $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ by Eq. 4.21 and Eq. 4.22
14 end
```

4.3.4 Complexity Analysis

We perform optimization by the temporal-batching gradient descent method for the training stage. The complexity of creating the batches is $\mathcal{O}(|\mathcal{E}^{tr}|)$, i.e., linear in the number of training interactions $\mathcal{E}^{tr}$, as each interaction is only looked at once. Therefore, the worst training complexity is $\mathcal{O}(\tau|\mathcal{E}^{tr}|l^{mx})$, where τ is the number of epochs, $|\mathcal{E}^{tr}|$ is the number of training interactions, and l^{mx} is the maximum number of historical neighbors in temporal GNN. In the inferring stage, we evaluate each test interaction sequentially to conduct the multi-round recommendation process. The worst inferring complexity is $\mathcal{O}(|\mathcal{E}^{ts}|(l^{mx} + \gamma^{rec}))$, where $|\mathcal{E}^{ts}|$ is the number of test interactions, γ^{rec} is the cost of generating top-k list after Eq. 4.24. γ^{rec} could be regarded as linear in the number of items. Note that we do not re-train the model on the test data. Hence, the total model complexity satisfies the efficiency requirement of our scenario. The pseudo-codes of the training and inferring are attached in Algorithm 1 and Algorithm 2, respectively.

4.4 Experiments on TCGC

In this section, we conduct extensive and comprehensive experiments to evaluate our proposed TCGC model, assessing its performance for dynamic recommendation. Specifically,

Table 4.1: Basic dataset statistics.

Dataset	#Users	#Items	#Interactions	#Average	Repetition
Wikipedia	8,227	1,000	157,474	19	61%
Foursquare	1,000	1,000	209,730	209	72%
Reddit	10,000	1,000	672,447	67	79%
Movielens	3,000	3,000	790,424	263	0%

we provide in-depth analyses of TCGC, including overall comparison with SOTA, performance robustness study, ablation study, the benefit of temporal-aware indicators, hyperparameters sensitivity, and training efficiency. These evaluations offer a thorough understanding of the capabilities and performance of our TCGC model. We aim to answer the following research questions:

- **RQ1**: How does TCGC perform when comparing with the state-of-the-art dynamic recommendation methods?
- **RQ2**: Whether the collaboration-aware indicator delivers improvement to existing temporal GNNs?
- **RQ3**: How do different core modules in our TCGC framework contribute to the overall performance?
- **RQ4**: How do the key hyperparameters influence the performance of our proposed TCGC?
- **RQ5**: What is the computational cost of TCGC compared with other methods?

4.4.1 Experimental Setup

4.4.1.1 Datasets

We evaluate our TCGC on four public datasets from diverse areas, the details of which are provided in Table 4.1. These datasets are full of user-item timestamped interactions. **Wikipedia**[1]: This is the one-month public dataset of edits made by the online users on the Wikipedia pages, generating 157,474 interactions. **Foursquare**[2]: This dataset contains point-of-interest check-ins in Tokyo collected about 10 months on the Foursquare, resulting in 209,730 interactions [113]. **Reddit**[3]: This dataset consists of one month of 672,447

[1]https://meta.wikimedia.org/wiki/Data_dumps
[2]https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[3]http://files.pushshift.io/reddit/

posts made by users on subreddits, which is collected from one of the biggest social networks, i.e., Reddit. **Movielens1M**[4]: This popular dataset records users' timestamped feedback on the movies within three years, containing 790,424 interactions. All datasets are filtered out by most active users and active items [40]. Note that there is no repetitive interaction in the Movielens dataset, which would verify the performance on 'unseen' items.

4.4.1.2 Baselines

We compare our proposed TCGC with various dynamic recommendation baselines:

- **Time-LSTM** [151]: A popular variant of LSTM that equips LSTM with the time-aware gates to model time intervals and users' sequential interactions.
- **DeepCoevolve** [13]: A deep evolutionary embedding network for recommendation by learning user and item features based on their interaction graph.
- **LatentCross** [3]: An easy-to-use technique that injects context information into the recurrent embedding network to deploy recurrent recommendations.
- **TGN** [67]: A representative temporal graph learning network that efficiently combines the memory module and graph-based operator to model sequential interactions.
- **DGCF** [44]: A dynamic graph collaborative filtering method that leverages three different structural orders to capture the dynamic collaborative information for recommendations.
- **FIRE** [101]: A novel incremental learning model that fully utilizes the graph signal processing to speed up the continuous-recommendation generation.
- **TREND** [96]: A novel framework for fine-grained temporal graph learning by incorporating the temporal event and node evolution with temporal Hawkes process-based graph network.
- **TIGER** [141]: A novel temporal interaction graph embedding method that extends the TGN with a restart module and a dual-memory module to better exploit neighbor information.
- **NeuFilter** [103] A novel data-centric algorithm based on neural Kalman Filtering for temporal recommendation to continuously learn more accurate user and item embeddings with the noisy interactions.

[4]https://grouplens.org/datasets/movielens/1m/

4.4.1.3 Evaluation Protocols and Metrics

For each dataset, we chronologically split it by 8:1:1 into training, validation and test. The training and validation data are treated as the offline phase, while test data is used to simulate the online inferring phase.

Following previous studies [40, 44, 101], we employ existing dynamic evaluation protocols and metrics, and the experiments consist of two stages, i.e., the training stage and inferring stage. For the **training stage**, we adopt the temporal-batching method, which is designed for the dynamic recommendation system. It continuously updates user and item representations without negative samples. It parallelizes the training of the interactions in each batch without causing conflict in updating user and item representations. The details of the temporal-batching method can be found in Section 4.3.3 and Algorithm 1. For the **inferring stage**, we evaluate each test interaction sequentially by generating new top-k recommendation lists at each timestamp. This is quite different from the static recommendation that only conducts a one-time evaluation. Thus, we are able to construct a dynamic recommendation system where the recommendation is a continuous multi-round process, keeping the same as the real-world platforms. At each timestamp, we adopt Recall@10 and MRR as evaluation metrics and report their average of all timestamps on the inferring stage. The inferring stage can be referred to Algorithm 2. All the selected baselines follow the same settings during training and inferring to conduct a fair comparison. Note that we do not re-train our model and other baselines on test data to avoid touching future information. Some previous studies use the test data to re-train the model during the inferring stage (by leveraging test loss to optimize the model). However, training a model based on test data and then conducting an evaluation is unfair. Thus, in this work, all methods, including baselines and our TCGC, do not leverage test data for training.

4.4.1.4 Reproducibility

For baseline methods, we use hyper-parameters as suggested in the original chapters. For those unspecified hyper-parameters due to new datasets, we use grid search for hyper-parameter tuning. The searching strategy is to select the hyper-parameters (e.g., the dimension of embedding and the length of neighbor set) that yield the best performance on the validation set. For our TCGC, we use the AdamW [56] with the learning rate of $1e-3$ as our optimizer for training. The 1-hop neighbor set is adopted for efficiency. The dimension of temporal representation and the length of the neighbor set are selected from $\{32, 64, 128, 256\}$ and $\{50, 100, 150, 200, 250, 300\}$, respectively. The dimension of static one-

Table 4.2: The overall performance on four datasets. The last bold row indicates the performance of our proposed TCGC.

Dataset	**Wikipedia**		**Foursquare**		**Reddit**		**Movielens**	
Metric	**MRR**	**Recall@10**	**MRR**	**Recall@10**	**MRR**	**Recall@10**	**MRR**	**Recall@10**
Time-LSTM	0.247	0.342	0.135	0.279	0.387	0.573	0.128	0.225
DeepCoevolve	0.515	0.563	0.217	0.326	0.171	0.275	0.196	0.303
LatentCross	0.424	0.481	0.193	0.289	0.421	0.588	0.174	0.237
TGN	0.593	0.679	0.336	0.651	0.606	0.784	0.235	0.401
DGCF	0.646	0.713	0.368	0.672	0.629	0.795	0.256	0.439
FIRE	0.259	0.648	0.154	0.524	0.332	0.814	0.152	0.394
TREND	0.588	0.691	0.370	0.663	0.647	0.809	0.289	0.453
TIGER	0.640	0.715	0.364	0.668	0.638	0.807	0.273	0.504
NeuFilter	0.651	0.724	0.378	0.672	0.684	0.837	0.302	0.516
Our TCGC	**0.670**	**0.744**	**0.394**	**0.691**	**0.708**	**0.853**	**0.324**	**0.581**

hot representation is determined by user and item sizes. To save the training cost, it is allowed to prepare the temporal collaboration-aware indicators in the data pre-processing phase. In addition, we apply l_2 regularization with $\varphi = 1e-2$. Last, all experiments are conducted with Pytorch in two NVIDIA 3090 GPUs.

4.4.2 Performance Comparison with SOTA

We report the **overall performance** of all recommendation methods in Table 4.2. It is observed that our proposed TCGC outperforms all dynamic baselines across all datasets, especially winning significant improvements on the Movielens datasets. We attribute this result to 1) collaboration-aware indicators that guide the evolution learning of users and items and 2) the temporal co-evolving graph network that studies the inherent correlation between event-level and history-level dynamics. NeuFilter, which aims to control the noisy information of representation learning, is second only to our TCGC in most cases. This suggests the power of data-centric exploration to strengthen representation learning. In addition, we find that most temporal GNN-based recommendation methods are significantly competitive in performance comparison, demonstrating the advantage of temporal graph networks for dynamic recommendation. Moreover, the considerable improvement on the Movielens dataset (no repetitive user-item interaction) indicates our TCGC could effectively discover the potential items that users have never seen before.

To evaluate the **model robustness** of our TCGC, we compare the performance with other GNN-based recommendation baselines by varying the proportion of training data from 100% (Table 4.2) to **75%**, **50%**, and **25%**. The results are shown in Table 4.3. We find

Table 4.3: Robustness evaluation over different temporal GNN-based recommendation models. For simplicity, we omit '@10'.

Proportion	**Dataset**	**Wikipedia**		**Foursquare**		**Reddit**		**Movielens**	
	Metric	**MRR**	**Recall**	**MRR**	**Recall**	**MRR**	**Recall**	**MRR**	**Recall**
75%	**TGN**	0.459	0.553	0.294	0.575	0.589	0.772	0.208	0.339
	DGCF	0.473	0.579	0.317	0.608	0.616	0.784	0.220	0.359
	TREND	0.523	0.608	0.332	0.587	0.632	0.791	0.254	0.409
	TIGER	0.553	0.632	0.325	0.614	0.625	0.788	0.246	0.421
	TCGC	**0.573**	**0.669**	**0.351**	**0.637**	**0.695**	**0.836**	**0.300**	**0.535**
50%	**TGN**	0.327	0.446	0.264	0.531	0.565	0.746	0.172	0.265
	DGCF	0.363	0.500	0.275	0.565	0.586	0.758	0.199	0.302
	TREND	0.407	0.529	0.292	0.542	0.609	0.776	0.227	0.342
	TIGER	0.440	0.524	0.305	0.574	0.617	0.764	0.205	0.368
	TCGC	**0.495**	**0.552**	**0.322**	**0.590**	**0.687**	**0.820**	**0.282**	**0.482**
25%	**TGN**	0.298	0.382	0.253	0.473	0.546	0.723	0.129	0.203
	DGCF	0.335	0.417	0.261	0.504	0.571	0.739	0.157	0.243
	TREND	0.319	0.433	0.269	0.480	0.589	0.768	0.189	0.296
	TIGER	0.340	0.441	0.270	0.517	0.598	0.750	0.178	0.280
	TCGC	**0.376**	**0.458**	**0.289**	**0.534**	**0.657**	**0.801**	**0.254**	**0.405**

that TCGC constantly outperforms other baselines even though less data is used. There are obvious performance reductions over the Wikipedia, Foursquare, and Movielens datasets across all baselines. One possible reason could be that the repetition rate of user-item interactions in those datasets is relatively lower (especially Movielens has no repetitive interaction), making the graph network more sensitive to the newly-arrived unseen interactions. However, the Reddit dataset still contains the same interaction records even though we decrease the number of interactions. Therefore, it could withstand the reduction of data. Also, we find that the comparison results among baselines would change with the data proportion. For example, TREND could be competitive with TIGER at 75% and 50% data proportion while TIGER outperforms TREND when only 25% data is used. Last, as we gradually decrease the training data of Foursquare and Wikipedia datasets, the performance of TCGC is getting closer to other methods. This may hint that all temporal GNN-based recommendation models, including TCGC, would be the same at an extremely small proportion of training data. However, before reaching that, our TCGC can remain robust to outperform other baselines.

Table 4.4: Evaluation on the benefit of temporal collaboration-aware indicator.

Dataset	**Wikipedia**		**Foursquare**		**Reddit**		**Movielens**	
Metric	**MRR**	**Recall**	**MRR**	**Recall**	**MRR**	**Recall**	**MRR**	**Recall**
TGN	0.593	0.679	0.336	0.651	0.606	0.784	0.235	0.401
TGN-CI	0.629	0.694	0.358	0.661	0.635	0.806	0.251	0.442
Improvement(%)	**6.1↑**	**2.2↑**	**6.5↑**	**1.5↑**	**4.8↑**	**2.7↑**	**6.8↑**	**10.2↑**
DGCF	0.646	0.713	0.368	0.672	0.629	0.795	0.256	0.439
DGCF-CI	0.660	0.722	0.377	0.680	0.653	0.813	0.280	0.475
Improvement(%)	**2.2↑**	**1.3↑**	**2.4↑**	**1.2↑**	**3.8↑**	**2.3↑**	**9.3↑**	**8.2↑**
TREND	0.588	0.691	0.370	0.663	0.647	0.809	0.289	0.453
TREND-CI	0.622	0.708	0.382	0.674	0.662	0.818	0.318	0.497
Improvement(%)	**5.8↑**	**2.5↑**	**3.2↑**	**1.7↑**	**2.3↑**	**1.1↑**	**10.0↑**	**9.7↑**
TIGER	0.640	0.715	0.364	0.668	0.638	0.807	0.273	0.504
TIGER-CI	0.652	0.726	0.375	0.679	0.679	0.824	0.299	0.547
Improvement(%)	**1.9↑**	**1.5↑**	**3.0↑**	**1.6↑**	**6.4↑**	**2.1↑**	**9.5↑**	**8.6↑**

4.4.3 Benefit of Temporal Collaboration-Aware Indicator

To further explore the benefit of our proposed **temporal collaboration-aware indicators**, we extend temporal GNN-based recommendation models, i.e., TGN, DGCF, TREND, and TIGER, by incorporating indicators (CI) into their core module. The temporal collaboration-aware indicator aims to scrutinize whether the continuous user-item interaction events would benefit the evolution learning of users and items. The details of implementation are:

- **TGN with CI**: The collaboration-aware indicators are extended to the user-item message function and graph attention network in TGN [67].
- **DGCF with CI**: The collaboration-aware indicators are extended to the zero-order, first-order, and second-order embedding update modules in DGCF [44].
- **TREND with CI**: The collaboration indicators are extended to excitement computation of both the historical items and items of the Hawkes-based graph network in TRNED [44].
- **TIGER with CI**: The collaboration indicators are extended to the user-item dual-memory message generation and temporal preference-update network of TIGER [141].

The results over the four datasets are presented in Table 4.4. It is obvious that the performance of all baselines would increase as we extend them with our temporal-aware col-

Table 4.5: Ablation study on key components of TCGC.

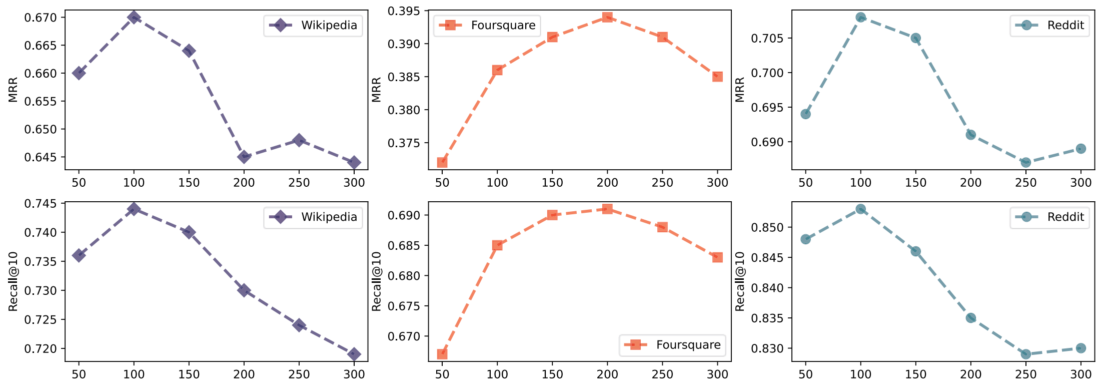
Dataset	**Wikipedia**		**Foursquare**		**Reddit**		**Movielens**	
Metric	**MRR**	**Recall**	**MRR**	**Recall**	**MRR**	**Recall**	**MRR**	**Recall**
w/o CI	0.642	0.721	0.362	0.670	0.683	0.829	0.281	0.519
w/o EveCI	0.649	0.728	0.374	0.681	0.690	0.835	0.295	0.537
w/o HisCI	0.653	0.730	0.369	0.672	0.695	0.840	0.310	0.554
w/o CoEvol	0.640	0.719	0.357	0.665	0.687	0.832	0.276	0.512
w/o PreSft	0.658	0.727	0.383	0.680	0.693	0.841	0.307	0.566
TCGC	**0.670**	**0.744**	**0.394**	**0.691**	**0.708**	**0.853**	**0.324**	**0.581**

laboration indicators. This phenomenon exactly verifies that existing dynamic recommendation models that directly learn the evolution of users and items would make the system performance suboptimal. They update temporal representations without investigating the actual impacts of the newly-arrived interactions. Thus, it leads to learning and absorbing inappropriate information. The improvement of Recall@10 over the Foursquare dataset is relatively smaller. This is probably due to the highest average of user interaction in Foursquare, which is not easily affected by noisy interactions. However, in most cases, incorporating our proposed temporal collaboration-aware indicators could deliver significant improvement in performance.

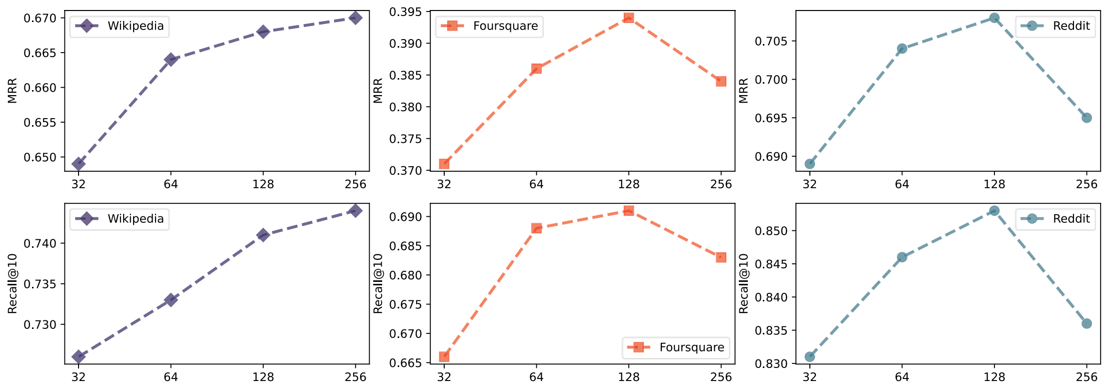
4.4.4 Ablation Study of TCGC Framework

To understand the contribution of each key component in TCGC, we study the following ablated variants. (1) **w/o CI**, which doesn't learn the collaboration-aware indicators; (2) **w/o EveCI**, which only learns collaboration-aware indicator for history-level dynamic; (3) **w/o HisCI**, which only learns collaboration-aware indicator for event-level dynamic; (4) **w/o CoEvol**, which doesn't adopt the co-evolving design; (5) **w/o PreSft**, which directly generate predictions without shifting user preference future time.

Table 4.5 shows the performance decreases severely when we remove all collaboration-aware indicators, which proves the effectiveness of guiding the evolution learning for both users and items. Meanwhile, we find that compared with the history-level collaboration indicator, the event-level collaboration indicator has a more positive impact on improving performance. This may suggest that learning the interaction itself is more informative during the dynamic multi-round procedure. Moreover, the temporal co-evolving network is also essential, which helps us to explore the inherent correlation between event-level



(a) The size of temporal neighbor set



(b) The dimension of temporal representation

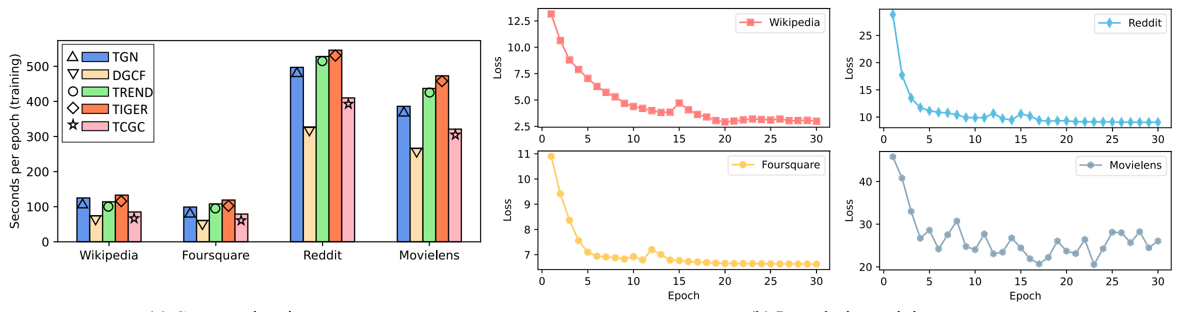
Figure 4.3: Hyperparameter sensitivity.

and history-level dynamics. The results indicate that removing this network would severely damage the performance of TCGC in all cases. In addition, since the dynamic system continuously refreshes the recommendation list over time, utilizing the preference-shifting module could be beneficial for discovering potential items in the future.

4.4.5 Hyperparameter Sensitivity

We analyze the sensitivity of hyperparameters over the Wikipedia, Foursquare, and Reddit datasets. The hyperparameters are (a) **the size of the temporal neighbor set** and (b) **the dimension of temporal representation**.

The results are reported in Figure 4.3. The size of the temporal neighbor set is searched from 50 to 300, while the dimension of temporal representation is selected from 32 to 256. It is confirmed that these hyperparameters do affect the performance of TCGC, and a relatively optimal selection exists for them. Along with the growth of neighbor size, performance first increases obviously and then goes down, which suggests that too many neigh-



(a) Computation time cost (b) Loss during training

Figure 4.4: Computational cost and loss of the training stage.

bors damage the performance, not to mention they would bring more computational cost. This may be caused by the noise data in temporal aggregation. However, due to the fact that the Foursquare dataset is the most dense one, it requires more neighbors than the other two datasets. Furthermore, a larger dimension of embedding could improve the overall performance. Especially, the Wikipedia dataset, which is much sparser, needs the largest dimension for representation learning. One possible reason is that there is less shared information between users (or items) due to its sparsity. In such cases, obtaining accurate temporal representations requires more dimensions to capture the actual interaction pattern from the users (or items). However, the excessive pursuit of large dimensions would sacrifice efficiency and overload the model with a huge computational cost.

4.4.6 Training Efficiency

To validate the efficiency of our TCGC, we have compared it with the other GNN-based baselines in terms of the training cost. The results are illustrated in Figure 4.4 (a). The two state-of-the-art methods, TIGER and TREND, which leverage sophisticated temporal GNNs, cost a lot. Also, the early representative model TGN cannot deliver significant improvement to training efficiency. We attribute this to its complex message-passing mechanism. In our TCGC, the collaboration indicators could be prepared in the data-preprocessing stage. Thus, the training of TCGC is more efficient among them and second only to DGCF. It could be observed that, for the same level datasets (e.g., Wikipedia and Foursquare, or Reddit and Movielens), the higher-average one tends to have less cost.

In addition, To verify the convergence of our TCGC, we illustrate the loss variation in Figure 4.4 (b). The losses of the Wikipedia, Foursquare, and Reddit datasets decrease constantly and reach their convergences around the 20th epoch. However, there are obvious fluctuations in the loss of the Movielens dataset. In general, its loss still declines and would converge from the whole training process. One possible reason is that there are no repetitive user-item interactions in the Movielens datasets, forcing the recommendation system to keep handling unseen interactions. Thus, exploiting the fast evolution for both users and items requires more effort. On the contrary, there are many repetitive interactions in the other three datasets, e.g., the repetition of the Reddit dataset is 79% (see Table 4.1). As a result, it is effortless to derive the interaction pattern between users and items. Therefore, the loss could get convergent swimmingly.

4.4.7 Discussion on Cold-start Problem

The cold-start situation where new users (also items) access the recommendation system for the first time, is a crucial problem in most real-world platforms. The interaction data of new users or new items are often very sparse, leading to the challenge of learning effective user or item representations.

Considering there is no side information (i.e., users' profile information and items' content information) in the datasets we adopted, we apply the normal trainable embedding table for initialization. The initialized embedding table is trained with the whole framework and could benefit from the global preference pattern driven by all other interactions. Nonetheless, the real-world platforms involve rich and complex side information, e.g., user's age, gender, location, and occupation, and item's title, category, color, and description. Hence, we suggest two aspects to better employ our TCGC in practice:

User-oriented Group-based Initialization. Similar users who share similar attributes tend to have similar preferences. Based on this assumption, when a new user accesses the recommendation, we could aggregate the embeddings of a certain number of existing users (denoted by G_u) who have the same profile as him/her, e.g., the same age, location, region or occupation as him/her. This group-based strategy could provide a warm and personalized initialization for the new user. Despite that, the time information should also be considered for the dynamic environment. For example, for two users q and z in G_u, if q enters the recommendation platform earlier than z, its impact should be larger. In addition, the number of interactions of each user in G_u is also an important factor in determining their impact since active users may indicate more explicit preferences. Hence, when we leverage the G_u to initialize the new user, we can assign each user in G_u with a time-decay weight and an activity-level weight. After we get the warm and personalized initialization of the new user, we feed it into our TCGC to generate suitable recommendations.

Item-oriented Text-based Initialization. Items usually involve more text information than users, e.g., the titles and descriptions of items. Based on rich texts, one possible solution for new item initialization is utilizing a pre-trained text encoder to convert the text information into dense embeddings. For example, we can incorporate a light pre-trained Transformer as an individual module with our TCGC to provide a warm text-based initialization for new items. To avoid additional computation costs during the model training, we could freeze the text encoder because its only purpose is to provide an initialized embedding for new items. Nonetheless, due to the fast change in the dynamic environment, it may be suitable to regularly fine-tune the pre-trained text encoder on the collected item information or directly replace the existing encoder with other state-of-the-art encoders.

4.5 Summary of TCGC

In this chapter, we investigate the temporal collaborative effectiveness existing in the evolution learning of dynamic recommendation scenarios. To the best of our knowledge, we are the first to propose temporal collaboration-aware indicators to precisely guide the evolution learning of temporal representations for both users and items. Then, in order to capture the inherent co-evolving correlation between event-level and history-level dynamics, we present a novel graph network, named Temporal Collaboration-Aware Graph Co-Evolution Learning (TCGC), which leverages the proposed collaboration-aware indicators to generate accurate temporal representations. Extensive experiments demonstrate both the superiority and efficiency of our TCGC on the dynamic recommendation task.

Chapter 5

Temporal Self-correcting Interaction Network for Recommendation

Dynamic interaction network (DIN), which is an evolving temporal user-item interaction graph, is prevalent in real-world online scenarios, e.g., dynamic multi-round recommendation. Existing studies have learned the evolving representations of users and items from the perspectives of newly-arrived and historical interactions. However, most of them assume the historical interactions would reflect the user/item's actual representation evolution, without scrutinizing whether they benefit the representation learning. In addition, they neglect the negative excitement of the newly-arrived interactions on learning representation. In this chapter, we propose the CREAM, a dynamic self-**C**orrecting inte**R**action n**E**twork with neg**A**tive excite**M**ent. Specifically, CREAM consists of four core components: (i) warm initialization, which generates flexible embeddings for the cold-start users and items; (ii) adaptive dual-side corrector, which dynamically and adaptively corrects the histories for both the user side and item side; (iii) temporal excitement learning, which learns the temporal negative impact of new interactions; and (iv) update and prediction, which integrates both temporal and static representations for interaction prediction. Extensive experiments on three public datasets demonstrate the proposed CREAM outperforms state-of-the-art baselines for the future interaction prediction task.

The content in this chapter mainly focuses on addressing the research challenge **RC 3** and overcoming the limitation **EL 3** by proposing a novel model CREAM.

5.1 Brief Introduction of CREAM

Dynamic interaction network (DIN), where users continuously interact with items over time [78], has widely existed in real-world online scenarios, e.g., social networks, streaming media, news platforms, and dynamic recommendation systems [141, 103, 30, 127]. It forms an evolving temporal user-item interaction graph. The same user may interact with different items over a period of time, and those interactions change over time, leading to the rapid evolution of both user and item representations [40, 148]. An important challenge is how to dynamically update user and item representations for predicting future interactions [95].

There is a considerable body of prior work toward this goal. Some early models [40, 8, 3], based on deep recurrent learning paradigm, leverage the newly-arrived interactions to encode new information into the old embeddings. Later on, temporal graph-based learning methods have been applied to this area [78, 96, 134, 141, 147]. Benefiting from the graph convolution mechanism, they efficiently propagate long-term information from historical interactions. Thus, we have witnessed remarkable achievements in the field of dynamic interaction network.

However, there are still some significant issues that prevent further improvement in predicting future interactions. First, most existing methods explicitly assume that all historical interactions would reflect the actual representation evolution of users and items (**Issue 1**), which means they do not scrutinize whether those historical interactions would benefit the temporal representation learning. In fact, some user-item interactions may be imperfect where they cannot represent the users' actual preferences or items' intrinsic attributes [52, 92, 21, 77]. Especially in the dynamic environment, users' behaviors may dynamically deviate from their real intents over time [138], causing it difficult to update representation incrementally.

For example, a user always prefers political news on a social platform. One day, due to the homepage's news feed, he/she occasionally watched a piece of entertainment news. Although it does not indicate a noise interaction, it is still inappropriate to update the user's actual representation based on that. Unfortunately, existing studies would leverage it to learn the historical information [96, 64, 45, 78]. Thus, a question naturally arises regarding *how to efficiently discover valuable interactions from history for accurate temporal representation learning*. To this end, we aim to dynamically and adaptively correct the history and filter out those inappropriate interactions for update. This is our core motivation in this chapter. More importantly, considering users and items are two types of stakeholders [60,

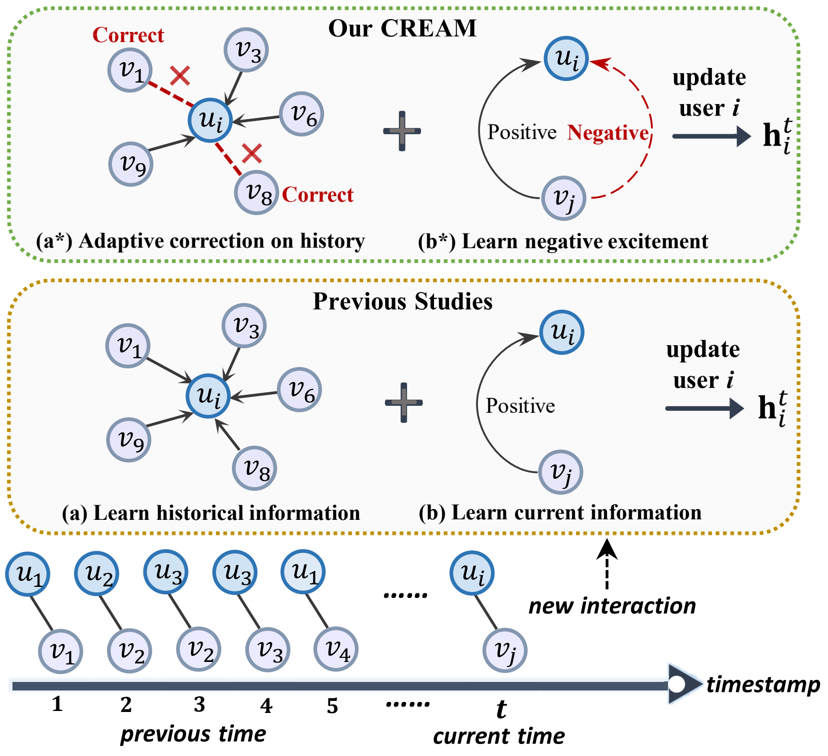

Our CREAM
Correct
Correct
v_1 v_3 u_i v_6 v_9 v_8
Positive Negative
update user i
$\mathbf{h}_i^t$
(a*) Adaptive correction on history
(b*) Learn negative excitement
Previous Studies
(a) Learn historical information
(b) Learn current information
new interaction
timestamp
1 2 3 4 5 ...... t
previous time
current time

Figure 5.1: A simple illustration of our motivation. we aim to: **(i)** dynamically and adaptively correct the history by filtering out inappropriate historical interactions for better aggregating historical information and **(ii)** thoroughly capture both positive and negative dynamics for learning current information.

82], we also seek to address this issue for both users' history and items' history.

Another issue is that most existing studies may not thoroughly capture the dynamics of newly-arrived interactions since they usually focus on the positive excitement, i.e., learning the positive effect of interactions on user future intentions [151, 40, 8, 57]. However, the negative excitement, which weakens users' actual intentions on the next interactions, has been neglected (**Issue 2**). The current interactions could negatively affect user future intentions [37, 68]. For instance, if a user is visiting an art-themed exhibition, he/she may not choose the same type for the next visit in a short period. Hence, another question arises regarding *how to dynamically model the negative excitement of the newly-arrived interactions.* This encourages us to build excitement learning from the perspectives of both positive and negative effects, thoroughly capturing the dynamics from newly-arrived interactions.

To sum up, we propose a novel method in this chapter, which is dynamic self-**C**orrecting inte**R**action n**E**twork with neg**A**tive excite**M**ent (CREAM). As shown in Figure 5.1, our objective is to deal with the two aforementioned issues. On the one hand, we carefully devise an **adaptive dual-side corrector** (ADC) to dynamically and adaptively filter out the inappropriate interactions that may not benefit the temporal representation learning for both users and items. However, ADC would not permanently delete interactions since its self-correction only affects the current update. By this means, the filtered-out interactions at early timestamps would still have the chance to return to history and then contribute to future timestamps. Compared with the TCGC (in Chapter 4) that examines the newly-arrived interactions from the perspective of collaborative neighbours, our CREAM in this chapter scrutinizes the historical interactions via the adaptive self-correction, thereby learning the long-term information better. On the other hand, we build a **temporal excitement learning** (TEL) to model both the positive and negative excitements from the newly-arrived interactions over users' future interaction behaviors. TEL leverages the time interval to learn the trend of evolving representation, making it a time-aware component. Besides, considering there are always new users and items continuously joining the system, we design a **warm initialization** for new users and items by converting raw features into dynamic embeddings and also providing a new item-wise strategy for static embeddings. Finally, drawing support from ADC and TEL, we update the temporal representations of the users and items in a light way and conduct prediction for future interactions. The main contributions of this work are four-fold:

- We propose an adaptive dual-side corrector module that could dynamically correct histories of user-item interactions for temporal representation learning on historical information.

- We present a temporal excitement learning module that could thoroughly capture both positive and negative dynamics of newly-arrived interactions.

- We devise a warm initialization module that could flexibly initialize the dynamic and static embeddings for new users and items.

- We present a new strategy to conduct future prediction by utilizing both temporal and static representations in one unified framework.

We conduct extensive experiments on three real-world datasets and compare our proposed CREAM with state-of-the-art baselines. The experimental results demonstrate the superiority and also the effectiveness of the proposed CREAM for the future interaction prediction task.

5.2 Preliminaries

Let $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ denote user set and item set, respectively. We use e_{ij}^t to represent the user-item interaction between the user u_i and item v_j at timestamp t. Thus, the continuous interaction behaviors between users and items form the interaction stream $\mathcal{E} = \{e_{ij}^t | u_i \in \mathcal{U}, v_j \in \mathcal{V}\}$. For simplicity, we also directly leverage i and j to denote u_i and v_j throughout this chapter. Following the previous work [40, 78, 141], we have the following definitions:

Definition 1 (Dynamic Interaction Network). Given the user set $\mathcal{U}$, item set $\mathcal{V}$ and stream set $\mathcal{E}$, the dynamic interaction network could be formalized by $\mathcal{G} = \{\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{X}\}$ where users are continuously interacting with items over time and this network is constantly changing with the growth of interactions. $\mathcal{X}$ represents the embedding space of all users and items involved in the network. For each user (also item), it has both temporal representation $\mathbf{h} \in \mathbb{R}^d$ and static representation $\bar{\mathbf{h}} \in \mathbb{R}^s$ where the d and s denotes their dimensions, respectively.

Definition 2 (Temporal Representation Learning). In dynamic interaction network, the temporal representations of users and items are evolving over time due to continuous interactions. Therefore, the objective of dynamic interaction network is to continuously learn the evolving temporal representation: $\mathbf{h}_i^t, \mathbf{h}_j^t \leftarrow \Phi(u_i, v_j, e_{ij}^t, t), u_i \in \mathcal{U}, v_j \in \mathcal{V}$. Φ is the learnable model function. Then, take the learned embeddings $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ to predict the future interactions for future timestamps.

5.3 CREAM Model Design

In this section, we introduce the technical details of our proposed CREAM, the dynamic self-correcting interaction network with negative excitement. As illustrated in Figure 5.2, CREAM contains five components, i.e., Warm Initialization, Adaptive Dual-side Corrector, Temporal Excitement Learning, Update and Prediction. Through the above components, our CREAM dynamically and adaptively corrects inappropriate historical interactions to better aggregate historical information, while investigating the negative excitement of newly-arrived interactions to thoroughly capture their dynamic effects. Besides, CREAM provides a flexible warm initialization for new users and items, and a light temporal update for representation evolution. In addition, the preference shift endows CREAM with the capability of generating prediction at any future timestamp.

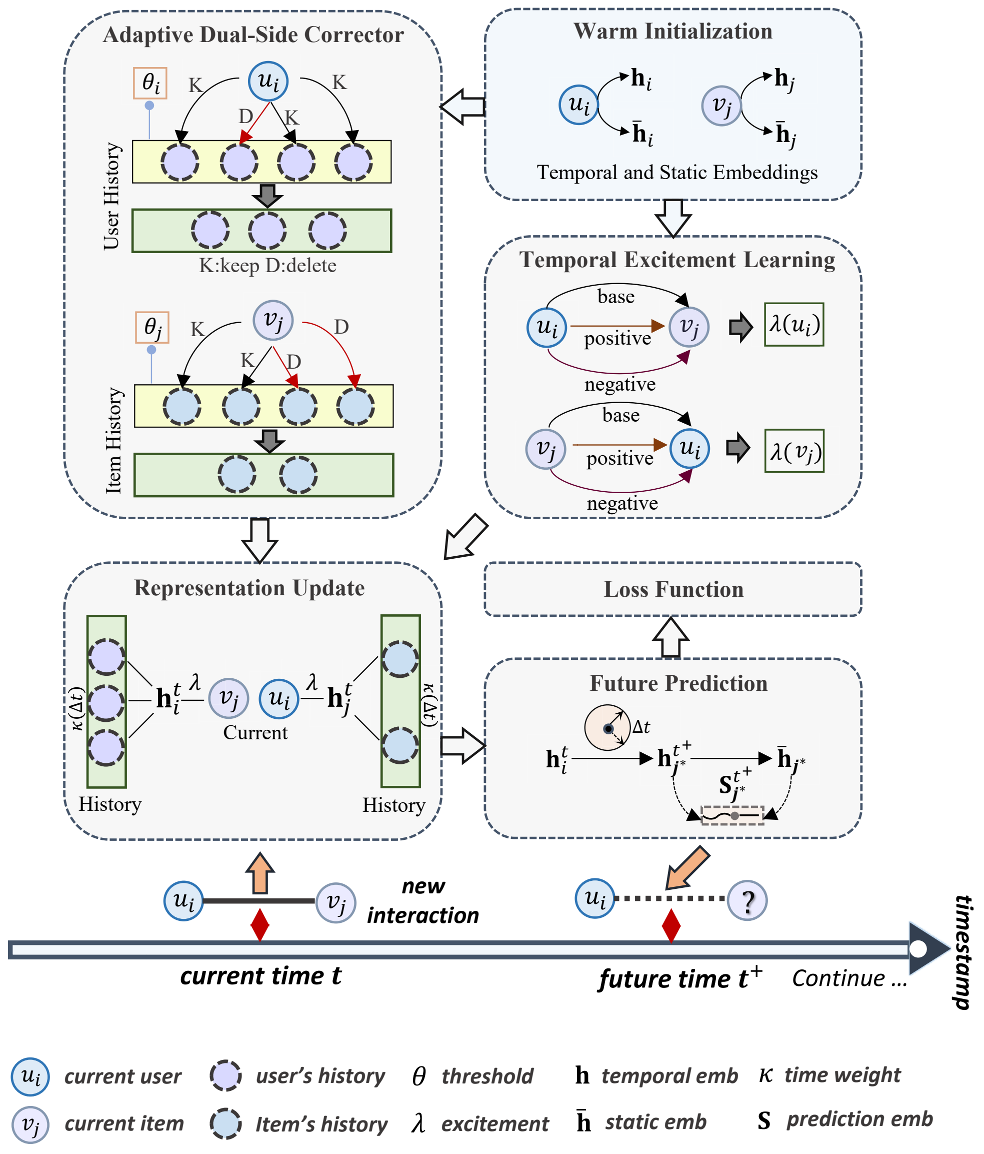


Figure 5.2: The overall architecture of our CREAM. First, the warm initialization provides flexible temporal and static representations for unseen user and item. Then, the adaptive dual-side corrector adaptively and dynamically corrects the historical interactions from both the user and item sides. Next, the temporal excitement learning explores the positive and negative effects of newly-arrived interaction. Last, we update the temporal representations of user and item and further predict the future interaction.

5.3.1 Warm Initialization

For any dynamic scenario, there are always new users and items constantly joining the system. Hence, the first urgency is how to handle the unseen users and items. One common solution is to randomly initialize embeddings, while it may impair the performance due to uncontrollable randomness. In this chapter, we first propose a flexible warm initialization to solve the problem. Specifically, we initialize two kinds of representations for both users and items, i.e., the temporal representation and the static representation.

Temporal Representation refers to the trainable embeddings that reflect users'/items' temporal evolution (e.g., user preference) over time. We take user u_i and item v_j as example. Inspired by StreamE [130], we present the feature-based initialization network:

$$\mathbf{h}_i = \sigma(\mathbf{W}^u \mathrm{e}_i + \mathbf{b}^u), \tag{5.1}$$

$$\mathbf{h}_j = \sigma(\mathbf{W}^v \mathrm{e}_j + \mathbf{b}^v), \tag{5.2}$$

where $\mathbf{W}^u$ and $\mathbf{b}^u$, $\mathbf{W}^v$ and $\mathbf{b}^v$, are learnable parameters for user and item, respectively. e_i and e_j are u_i's and v_j's feature inputs, which are flexibly determined. σ represents the LeakyReLU function throughout the chapter. However, sometimes we cannot acquire the feature embeddings. Then, one simple alternative is to initialize a global learnable embedding table [89].

Static Representation refers to the untrainable embeddings that denote the inherent characteristics. Previous studies leverage one-hot ID embeddings as static representations [78, 40, 8]. Here, we propose a novel item-wise multi-hot embedding strategy. For the item side, we still employ one-hot ID embeddings, i.e., $\bar{\mathbf{h}}_j \in \mathbb{R}^{|\mathcal{V}|}$, while for the user-side, we also set $\bar{\mathbf{h}}_i \in \mathbb{R}^{|\mathcal{V}|}$ where the corresponding dimension in $\bar{\mathbf{h}}_i$ would be activated if user u_i interacts with an item. By this means, the user static representation keeps the same size as the item static representation. Moreover, it constantly trances the interacted items of the user and would not be fixed over time. This is more suitable for the dynamic environment.

5.3.2 Adaptive Dual-side Corrector

Suppose user u_i is interacting with the item v_j at the current timestamp t. The goal of the dynamic interaction network is to learn the current temporal representations $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ that are evolving from previous temporal representations $\mathbf{h}_i^{t^-}$ and $\mathbf{h}_j^{t^-}$. Existing studies fully aggregate historical interactions to explore long-term information [44, 96, 141, 78]. However, they explicitly assume that all historical interactions would reflect the actual evolution of users and items, without scrutinizing whether the historical interactions truly benefit the temporal representation learning. This may pose the risk of learning uninformative or even harmful user/item representations [92, 77]. Even though some interactions are not noise data, they cannot be beneficial for learning the temporal representations of users and items. Roughly leveraging them into the update may negatively affect the interaction prediction. For instance, after a user purchases a mobile phone, this interaction is learned by the model and the user's preference is updated. Consequently, the system would recommend other phone brands to the user based on the learned preference. However, products like mobile phones are typically not repurchased frequently within a certain time period, especially for individual users. It is challenging to explore the correlation among users and items and discover those 'intended-but-inappropriate' interactions.

To this end, we aim to build the corrector to dynamically correct history in terms of inappropriate historical interactions. Our considerations come from two parts: **(i)** The corrector should be adaptive and efficient since it is oriented for the dynamic environment; and **(ii)** The corrector should work for both user side and item side since item representations are evolving as well as user representations. Thus, inspired by reliable long-term representation learning [52, 133], we propose the **Adaptive Dual-side Corrector** (ADC) in this chapter. Let $\mathcal{H}_i^t$ and $\mathcal{H}_j^t$ denote the u_i's and v_j's histories, respectively. $\mathcal{H}_i^t$ records interacted items of u_i while $\mathcal{H}_j^t$ records interacted users of v_j. We first inject positional embeddings into the historical interactions:

$$\mathbf{z}_x = \mathbf{h}_x^{t^-} + \varphi(x), \forall x \in \mathcal{H}_i^t, \tag{5.3}$$

$$\mathbf{z}_y = \mathbf{h}_y^{t^-} + \varphi(y), \forall y \in \mathcal{H}_j^t, \tag{5.4}$$

where x is the historical interacted item of user u_i and y is the historical interacted user of item v_j. $\mathbf{h}_x^{t^-}$ and $\mathbf{h}_y^{t^-}$ are the temporal representation of x and y at last timestamp. $\varphi(\cdot)$ returns the positional embedding. Following the conventional strategy in natural language processing, we adopt two learnable position embedding tables $\mathbf{P}^u$ and $\mathbf{P}^v$:

$$\varphi(x) = \mathbf{P}^v\left[\text{pos}(x), :\right], \varphi(y) = \mathbf{P}^u\left[\text{pos}(y), :\right], \tag{5.5}$$

where $\text{pos}(\cdot)$ indicates the position in history. Thus, $\varphi(\cdot)$ would look up the corresponding embedding from the position table. Then, we devise the two-layer MLP to capture the inherent correlation between the target embedding and the historical embedding that has been already injected with position information:

$$\text{Cor}(x|u_i) = \sigma(\mathbf{w}_1^u(\mathbf{W}_2^u < \mathbf{h}_i^{t^-}, \mathbf{z}_x >) + b^u), \tag{5.6}$$

$$\text{Cor}(y|v_j) = \sigma(\mathbf{w}_1^v(\mathbf{W}_2^v < \mathbf{h}_j^{t^-}, \mathbf{z}_y >) + b^v), \tag{5.7}$$

where $\mathbf{w}_1^u, \mathbf{W}_2^u, b^u$ and $\mathbf{w}_1^v, \mathbf{W}_2^v, b^v$ are learnable parameters for the user side and item side, respectively. $< \cdot >$ represents the concatenation operation. Now we have derived the correlation scores $\text{Cor}(x|u_i)$ and $\text{Cor}(y|v_j)$ to measure u_i's historical interaction and v_j's historical interaction. We could manually determine a threshold to filter out interactions with correlation scores below the threshold. However, the manual threshold cannot adaptively deal with different histories, causing insufficient or excessive corrections. Therefore, we aim to dynamically conduct self-correction with the adaptive thresholds. We set the learnable threshold-aware embeddings for each user and item to denote their own specific dynamic attributes on thresholds:

$$\mathbf{r}_{u_i} = \mathscr{T}^u[i,:], \mathbf{r}_{v_j} = \mathscr{T}^v[j,:], \tag{5.8}$$

Then, we inject the threshold-aware dynamics into u_i's representation and v_j's representation to obtain their adaptive thresholds:

$$\theta(i) = \mathbf{r}_{u_i}(\mathbf{w}_\theta^u \mathbf{h}_i^{t^-}), \theta(j) = \mathbf{r}_{v_j}(\mathbf{w}_\theta^v \mathbf{h}_j^{t^-}), \tag{5.9}$$

where $\mathbf{w}_\theta^u$ and $\mathbf{w}_\theta^v$ are trainable parameters. $\theta(i)$ and $\theta(j)$ are the adaptive thresholds for u_i and v_j, respectively. How to filter out the historical interactions of u_i is totally determined by u_i. Similarly, how to filter out the historical interactions of v_j is decided by v_j itself. Next, by utilizing $\theta(i)$ and $\theta(j)$, we correct the original histories $\mathscr{H}_i^t$ and $\mathscr{H}_j^t$ as follows:

$$\widetilde{\mathscr{H}}_i^t = \{I_{<\theta(i)}(\text{Cor}(x_1|u_i)), I_{<\theta(i)}(\text{Cor}(x_2|u_i)), I_{<\theta(i)}(\text{Cor}(x_3|u_i)), ...\}, \tag{5.10}$$

$$\widetilde{\mathscr{H}}_j^t = \{I_{<\theta(j)}(\text{Cor}(y_1|v_j)), I_{<\theta(j)}(\text{Cor}(y_2|v_j)), I_{<\theta(j)}(\text{Cor}(y_3|v_j)), ...\}, \tag{5.11}$$

where $I_{>\theta(\cdot)}(\cdot)$ is the indicator function. If the input is smaller than $\theta(\cdot)$, remove it from history. $\widetilde{\mathscr{H}}_i^t$ and $\widetilde{\mathscr{H}}_j^t$ are the corrected histories for u_i and v_j at current timestamp t. Note that the $\widetilde{\mathscr{H}}_i^t$ and $\widetilde{\mathscr{H}}_j^t$ would only affect the temporal representation learning of the current timestamp t, which means the filtered-out interactions still have chance returning to history at other timestamps. Therefore, our proposed ADC module could dynamically measure interactions and adaptively execute corrections over time.

5.3.3 Temporal Excitement Learning

Short-term information from the newly-arrived interactions, is another important aspect to derive the evolution of temporal representations. Since user u_i is interacting with item v_j at t, they tend to co-evolve by learning information from each other. Existing studies inject the u_i's (v_j's) representation into v_j's (u_i's) representation to let their temporal representations evolve from each other [13, 40, 8, 78]. However, they only model the positive excitement (i.e., positive effect on update) from the new interaction between u_i and v_j, neglecting the potential negative excitement that actually indicates unnecessary update.

To this end, we aim to thoroughly learn the excitement from both positive and negative perspectives. Inspired by Hawkes process-based learning [122, 37], we propose the **Temporal Excitement Learning** (TEL) in this chapter. Nevertheless, different from complex sequential modeling, this module only focuses on modeling the effect of the current new interaction. Take v_j as example and our TEL could be formalized as:

$$\lambda(v_j|u_i,t) = \lambda_0 + \varphi_p(\Delta t, v_j|u_i) - \varphi_n(\Delta t, v_j|u_i), \tag{5.12}$$

where $\lambda(v_j|u_i,t)$ represents the excitement from item v_j on user u_i at current timestamp t. Δt is the time interval between t and the timestamp of u_i's previous interaction. λ_0 is the base excitement. $\varphi_p(\Delta t, v_j|u_i)$ is the positive excitement while $\varphi_n(\Delta t, v_j|u_i)$ is the negative excitement. For the first term λ_0, we have:

$$\lambda_0 = (\mathbf{h}_i^{t^-})^T \mathbf{h}_j^{t^-} + b_u + b_v, \tag{5.13}$$

where b_u and b_v are learnable user bias and item bias, respectively. The base excitement is determined by the inner product of u_i and v_j. For the positive and negative excitements, we consider time-aware attentive modeling that incorporates representation with time interval information, because u_i's previous interaction would affect her/his intention on her/his next interaction [30]:

$$\mathbf{h}_{j|p}^{\Delta t} = \mathbf{h}_j^{t-} \odot (\mathbf{w}_1^p \Delta t), \mathbf{h}_{j|n}^{\Delta t} = \mathbf{h}_j^{t-} \odot (\mathbf{w}_1^n \Delta t), \tag{5.14}$$

where $\mathbf{h}_{j|p}^{\Delta t}$ and $\mathbf{h}_{j|n}^{\Delta t}$ are the time-aware positive and negative embeddings of v_j. $\mathbf{w}_1^p$ and $\mathbf{w}_1^n$ are learnable parameters. $\odot$ denotes the element-wise product. Next, we can derive the positive and negative excitements based on $\mathbf{h}_{j|p}^{\Delta t}$ and $\mathbf{h}_{j|n}^{\Delta t}$:

$$\varphi_p(\Delta t, v_j|u_i) = Sig(\mathbf{w}_2^p(\mathbf{W}^p < \mathbf{h}_{j|p}^{\Delta t}, \mathbf{h}_i^{t^-} >) + b^p)), \tag{5.15}$$

$$\varphi_n(\Delta t, v_j|u_i) = Sig(\mathbf{w}_2^n(\mathbf{W}^n < \mathbf{h}_{j|n}^{\Delta t}, \mathbf{h}_i^{t^-} >) + b^n)) \tag{5.16}$$

where $\mathbf{w}_2^p, \mathbf{W}^p, b_p$ and $\mathbf{w}_2^n, \mathbf{W}^n, b_n$ are learnable parameters for positive and negative excitements, respectively. Sig is the Sigmoid function that returns a value in (0,1). Now, we can utilize λ_0, $\varphi_p(\Delta t, v_j|u_i)$ and $\varphi_n(\Delta t, v_j|u_i)$ to accomplish the Eq. 5.12 and get final excitement $\lambda(v_j|u_i, t)$ which indicate the v_j's effect on updating u_i's representation. Our advantage is that we thoroughly capture the dynamic effect from both positive and negative excitements.

Similarly, we can derive the $\lambda(u_i|v_j, t)$ that learns the excitement from u_i on v_j. We omit this part because it is the same as the above process.

5.3.4 Update and Prediction

On the one hand, we have constructed the adaptive dual-side corrector ADC that would dynamically and adaptively correct histories of both user side and item side to better model long-term information. On the other hand, we have presented temporal excitement learning TEL that would investigate both positive and negative excitements to learn short-term information. We leverage the ADC and TEL together to update the temporal representations of user u_i and item v_j at the current timestamp t.

Update. Considering we are dynamic scenario where users continuously interact with items over time, we decide to take a lightweight but efficient strategy without additional trainable parameters to update temporal representations, which is formalized as:

$$\mathbf{h}_i^t = \sigma(\lambda(v_j|u_i, t)\mathbf{h}_j^{t^-} + \sum_{x \in \widetilde{\mathcal{H}}_i^t} \kappa(\Delta t)\mathbf{h}_x^{t^-}), \tag{5.17}$$

$$\mathbf{h}_j^t = \sigma(\lambda(u_i|v_j, t)\mathbf{h}_i^{t^-} + \sum_{y \in \widetilde{\mathcal{H}}_j^t} \kappa(\Delta t)\mathbf{h}_y^{t^-}), \tag{5.18}$$

where $\lambda(v_j|u_i, t), \lambda(u_i|v_j, t)$ are excitements derived from TEL module and $\widetilde{\mathcal{H}}_i^t, \widetilde{\mathcal{H}}_j^t$ are the corrected histories derived from ADC module. Δt denotes the time interval between historical interaction and current timestamp. $\kappa(\cdot)$ is the time-decay weight function [96] which would assign each historical interaction with a weight according to the time interval. Take original $\mathcal{H}_i^t$ as example: $\kappa(\Delta t) = \frac{\exp(-\Delta t)}{\sum_{x' \in \mathcal{H}_i^t} \exp(-\Delta t')}$.

The first term in the update is used to capture short-term information based on the temporal excitement while the second term is designed for modeling long-term information from the corrected history. With our ADC and TEL modules, we could more precisely update temporal representations for both user and item. As a consequence, u_i evolves from $\mathbf{h}_i^{t^-}$ to $\mathbf{h}_i^t$ and v_j evolves from $\mathbf{h}_j^{t^-}$ to $\mathbf{h}_j^t$.

Prediction. The goal of dynamic interaction network is to predict the future interaction between users and items. However, the current representation of users may not reflect their future intentions [40, 30]. Thus, we refer to the Eq. 5.14 and design a time-aware preference shift to predict the future preference embedding of u_i at timestamp t^+:

$$\mathbf{h}^{t^+} = \sigma(\mathbf{W}_1^{\text{sft}}(\mathbf{h}_j^t \odot (\mathbf{1} + \mathbf{w}_2^{\text{sft}} \Delta t)), \tag{5.19}$$

where $\mathbf{W}_1^{\text{sft}}$ and $\mathbf{w}_2^{\text{sft}}$ are learnable parameters. Δt is the time interval bewteen the future time t^+ and current time t. $\odot$ denotes the element-wise product. The core advantage of this preference shift is that we can predict any future timestamp, which is unlimited.

However, to accurately conduct interaction prediction, we should also consider the static representation part. Previous studies, e.g., DGEL [78], prefer directly predicting the whole combination of temporal and static representations. In our work, we propose a **new strategy**, which utilizes the temporal representation to dynamically refine existing static representation. Specifically, we derive an attentive vector based on u_i's predicted preference embedding and adjust its static embedding in a fine-grained way:

$$\bar{\mathbf{h}}^{t^+} = \text{ReLu}(\mathbf{W}^{\text{rfn}}(<\mathbf{h}^{t^+}, \bar{\mathbf{h}}_j>)) \odot \bar{\mathbf{h}}_i, \tag{5.20}$$

where $\mathbf{W}^{\text{rfn}}$ is learnable parameter and $\odot$ is the element-wise product. We take the static embedding $\bar{\mathbf{h}}_j$ of item v_j into account because the user may repeatedly interact with the same item. The predicted preference embedding $\mathbf{h}^{t^+}$ not only affects the user's dynamic preference in the future but also determines the possible static attributes that relate to her/him.

We use the concatenation $\mathbf{S}^{t^+}$=$<\mathbf{h}^{t^+}, \bar{\mathbf{h}}^{t^+}>$ of u_i's predicted preference embedding and static embedding as the objective, and find an item j' whose $<\mathbf{h}_{j'}^t, \bar{\mathbf{h}}_{j'}>$ is nearest to $\mathbf{S}^{t^+}$, as the predicted future interaction.

Our goal is that the prediction $\mathbf{S}^{t^+}$ could hit the representation of the user's actual-interacted item. Therefore, the loss of our overall framework is defined as follows:

$$\mathscr{L} = \sum_{e_{i'j'}^t \in \mathscr{E}} \left\| \mathbf{S}^t - <\mathbf{h}_{j'}^{t^-}, \bar{\mathbf{h}}_{j'}> \right\|_2 + \alpha \|\Phi\|_2 \tag{5.21}$$

where $\mathbf{S}^t$ is the predicted objective of user i' at t. j' is the ground-truth interacted item. $\mathbf{h}_{j'}^{t^-}$ and $\bar{\mathbf{h}}_{j'}$ are the temporal and static representations of it. Φ contains all learnable parameters of our CREAM. The second term is the regularization with a control weight α. Leveraging the loss, we adopt the temporal-batch method to train the whole framework. We introduce it in the experimental setup section.

Algorithm 3: Training of CREAM (one epoch)

```
   Input: $\mathcal{U}, \mathcal{V}, \mathcal{E}^{tr}$, temporal batch $B = \emptyset$
   Output: trained $\Phi$ of this epoch
1  Randomly initialize the parameter set $\Phi$ of TCGL;
2  Warmly initialize all representations by Eq. 5.1 and Eq. 5.2;
3  foreach interaction $e_{ij}^t \in \mathcal{E}^{tr}$ do
4      Update:
5      add $u_i$ to $\mathcal{H}_j^t$ and $v_j$ to $\mathcal{H}_i^t$;
6      correct user history and item history by Eq. 5.10 and Eq. 5.11 to get the corrected $\widetilde{\mathcal{H}}_i^t$
         and $\widetilde{\mathcal{H}}_j^t$;
7      learn the temporal excitement by Eq. 5.15 and Eq. 5.16 to get the $\lambda(v_j|u_i, t)$ and
         $\lambda(u_i|v_j, t)$ ;
8      update temporal representations $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ for $u_i$ and $v_j$ by Eq. 5.17 and 5.18 ;
9      Temporal-batch Optimization:
10     if $u_i$ and $v_j \notin B$ then
11         add $e_{ij}^t$ to $B$ ;
12     else
13         obtain the loss of $B$ by Eq. 5.21;
14         optimize the $\Phi$ of CREAM;
15         let $B = \{e_{ij}^t\}$;
16     end
17 end
```

5.3.5 Complexity Analysis

For the training stage, we perform optimization over batches of training interactions using the temporal-batching gradient descent method. The training complexity is $\mathcal{O}(\tau|\mathcal{E}^{tr}|(\ell^{\text{ADC}}+\ell^{\text{TEL}}))$, where τ is the number of epochs and $|\mathcal{E}^{tr}|$ is the number of training interactions. ℓ^{ADC} and ℓ^{TEL} are the computation cost of the adaptive dual-side corrector and the temporal excitement learning, respectively. They are affected by both the dimension d of temporal representation and the size of the neighbor set. For the inferring stage, we continuously generate interaction prediction and sequentially evaluate each test interaction to conduct the dynamic interaction network. The inferring complexity is $\mathcal{O}(|\mathcal{E}^{ts}|(\ell^{\text{ADC}}+\ell^{\text{TEL}}+\gamma^{out}))$, where $|\mathcal{E}^{ts}|$ is the number of test interactions and γ^{out} is the cost of finding the nearest item after prediction. The pseudo-codes of training and inferring can be found in the Algorithm. 3 and Algorithm. 4, respectively. Note that our scenario is the dynamic interaction network between users and items in real-world platforms. Thus, we continuously conduct predictions over time, which distinguishes our work from one-time prediction scenarios.

Algorithm 4: Inferring of CREAM (one epoch)

```
   Input: $\mathscr{U}, \mathscr{V}, \mathscr{E}^{ts}$, trained $\Phi$
   Output: evaluation results
1  foreach interaction $e_{ij}^{t} \in \mathscr{E}^{ts}$ do
2      Prediction:
3      fetch the $\mathbf{h}_i^{t^-}$ from previous time;
4      generate the next-interaction prediction at time $t$ for $u_i$ based on Eq. 5.19 and Eq. 5.20;
5      compare with $e_{ij}^{t}$ for evaluation;
6      Update:
7      add $u_i$ to $\mathscr{H}_j^t$ and $v_j$ to $\mathscr{H}_i^t$;
8      correct user history and item history by Eq. 5.10 and Eq. 5.11;
9      learn the temporal excitement by Eq. 5.15 and Eq. 5.16;
10     update temporal representations $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ for $u_i$ and $v_j$ by Eq. 5.17 and 5.18 ;
11 end
```

Table 5.1: Basic dataset statistics: the number of users (Users), the number of items (Items), the number of interactions (Interactions), the average number of interacted items per user (Avg.(Item)), the average number of interactions per user (Avg.(Inter)), and the sparsity computing the proportion of zero elements in the user-item matrix.

Dataset	Wikipedia	Foursquare	Reddit
Users	8,227	1,000	10,000
Items	1,000	1,000	1,000
Interactions	157,474	209,730	672,447
Avg.(Item)	2.22	39.08	67.24
Avg.(Inter)	19.14	209.73	67.24
Sparsity	99.78%	95.89%	99.20%

5.4 Experiments on CREAM

In this section, we present extensive and comprehensive experiments to demonstrate the superiority and effectiveness of our CREAM on dynamic interaction task. Specifically, our experiments include the overall performance comparison, the ablation study, the variant exploration, and the hyperparameters sensitivity. We aim to answer the following research questions:

- **RQ1**: Whether our proposed CREAM can outperform the state-of-art methods (i.e., deep recurrent learning methods and temporal graph learning methods) on the dynamic interaction prediction task?
- **RQ2**: How do the core modules (e.g., the adaptive dual-side corrector and the temporal excitement learning) in our CREAM contribute to the overall performance?

- **RQ3**: Whether other variants of our CREAM (e.g., predicting the combination of temporal and static representations) could improve the performance of our CREAM?
- **RQ4**: How sensitive is our CREAM with the key hyperparameters (i.e., the size of temporal history and the dimension of temporal representation)?

5.4.1 Experimental Setup

5.4.1.1 Datasets.

We conduct experiments on three real-world interaction datasets that are widely used in the field of dynamic interaction network. **Wikipedia**[1]: This dataset contains one month of edits on Wikipedia and 157, 474 interactions are filtered out by 8,227 users and 1,000 edited pages. **Foursquare**[2]: This dataset contains location check-ins in Tokyo for about 10 months, totaling 209, 730 check-ins determined by 1,000 users and 1,000 visited locations. **Reddit**[3]: This dataset consists of one month of posts made by users on subreddits, resulting in 672, 447 interactions by 1,000 subreddits as items and 10,000 users. The dataset statistics can be found in Table 5.1. All datasets can be downloaded publicly.

5.4.1.2 Baselines.

We compare CREAM with two groups of state-of-the-art baselines. The first group is deep recurrent learning methods: **Time-LSTM** [151], **Jodie** [40], **HILI** [8], and **NeuFilter** [103]. The second The second group is the temporal graph learning methods: **TGN** [67], **TREND** [96], **TIGER** [141], and **DGEL** [78]. The details of those baselines are as follows:

- **Time-LSTM** [151]: A popular variant of the long short-term memory method LSTM by equipping it with gates to investigate time intervals.
- **Jodie** [40]: A representative recurrent method that builds a novel project function to continuously predict the future embedding trajectories of users;
- **HILI** [8]: A sequential model that aims to efficiently update embeddings and prevent the information asymmetry by attention layer and self-linear layer;
- **NeuFilter** [103]: A novel model that incorporates Kalman filtering into user-item interactions to construct robust learning and handle noise interactions.

[1] http://snap.stanford.edu/jodie/wikipedia.csv
[2] sites.google.com/site/yangdingqi/home/foursquare-dataset
[3] http://snap.stanford.edu/jodie/reddit.csv

- **TGN** [67]: A popular standard graph method that designs a generic framework to model temporal graph by timestamped events.
- **TREND** [96]: A novel framework that presents the temporal Hawkes process-based graph network to capture fine-grained event and node dynamics.
- **TIGER** [141]: A novel interaction graph embedding method that devises a temporal dual-memory graph neural network to develop embedding update.
- **DGEL** [78]: A novel dynamic graph model that fully learns the temporal graph evolution from the perspectives of both newly-arrived and historical interactions.

5.4.1.3 Evaluation Protocols and Metrics.

Our experimental settings strictly follow the DGEL and we conduct the future interaction prediction task. For each dataset, we chronologically split it by 8:1:1 into training, validation and test. The training data are used for the training stage while the validation data and the test data are used for the inferring stage.

For the training stage, we adopt the temporal-batch training method proposed by Jodie and select the best epoch based on validation data. The temporal-batch method ensures three criteria: (1) the interactions in one batch should be processed simultaneously; (2) different batches should maintain their sequential ordering; (3) the users and items in each batch should not be duplicated (otherwise, it would lead to conflict when updating the same user or item). In practice, we assign each interaction e_{ij}^{t} to a particular batch B_l, where $l \in [1, \mathscr{E}]$. We first initialize $|\mathscr{E}|$ empty batches (the worst case is that each batch only contains one interaction). We iterate through the interaction stream and add each interaction to a batch B_l. Let $maxBatch(s, r)$ be the batch with the largest index that has an interaction involving a node s (user or item) till the interaction e^r. Then, the next interaction e^{r+1} (i.e., between user i and item j), is assigned to the batch with index = $max(1 + maxBatch(i, r), 1 + maxBatch(j, r))$.

For the inferring stage, we sequentially evaluate each test interaction by generating prediction and updating representation at each timestamp. This means we conduct continuous inferring across different timestamps, which is different from one-time evaluation. This ensures the multi-round procedure of the dynamic interaction network. At each timestamp, we adopt Recall@10 and MRR as evaluation metrics, and finally, we report their averages of all timestamps during the inferring stage. The pseudo-codes of training and inferring can be also found in the Algorithm. 3 and Algorithm. 4, respectively. Note that during

Table 5.2: The overall performance of future interaction prediction. We compare CREAM with different methods from two categories and conduct comparison on three datasets. The bold and the underline indicate the best and second-best results.

Dataset	**Wikipedia**		**Foursquare**		**Reddit**	
Metric	**MRR**	**Recall@10**	**MRR**	**Recall@10**	**MRR**	**Recall@10**
Time-LSTM	0.247	0.342	0.135	0.279	0.387	0.573
Jodie	0.639	0.705	0.349	0.652	0.658	0.812
HILI	<u>0.680</u>	0.746	<u>0.392</u>	0.685	<u>0.719</u>	0.847
NeuFilter	0.651	0.724	0.378	0.672	0.684	0.837
TGN	0.593	0.679	0.336	0.651	0.606	0.784
TREND	0.588	0.691	0.370	0.663	0.647	0.809
TIGER	0.640	0.715	0.364	0.668	0.638	0.807
DGEL	0.674	<u>0.756</u>	0.381	<u>0.697</u>	0.701	<u>0.849</u>
Our CREAM	**0.704**	**0.842**	**0.436**	**0.725**	**0.730**	**0.876**

the inferring stage, we DO NOT re-train the baselines and also our CREAM on test data, to conduct fair comparison among them.

5.4.1.4 Reproducibility.

For DGEL, TREND, and Time-LSTM, we fetch their performance results from the chapter of DGEL. For other baselines, we use their suggested hyper-parameters to yield the best performance on the validation set. For our proposed CREAM, we use the AdamW [56] with the learning rate of 0.001 as an optimizer for model training. The epochs are set as 30. The dimension of the temporal representation and the length of the neighbor set are selected from $\{16, 32, 64, 128, 256, 512\}$ and $\{10, 50, 100, 150, 200, 300\}$, respectively. The dimension of static embedding is determined by item size. In addition, we apply l_2 regularization with $\alpha = 0.001$. Last, all experiments are conducted with Pytorch in NVIDIA 4090 GPU and AMD EPYC 9754 CPU.

5.4.2 Overall Performance

To evaluate the overall performance of our proposed CREAM, we compare it with different methods over three datasets. The results are reported in Table 5.2. We have several findings.

First of all, our CREAM outperforms all baseline methods on all datasets, which demonstrates the superiority of our CREAM in continuously predicting the interactions for the dynamic interaction network. Compared with the second-best performance, our CREAM gains 3.5%, 11.2%, 1.5% improvements on MRR, and 11.4%, 4.0%, 3.2% improvements on

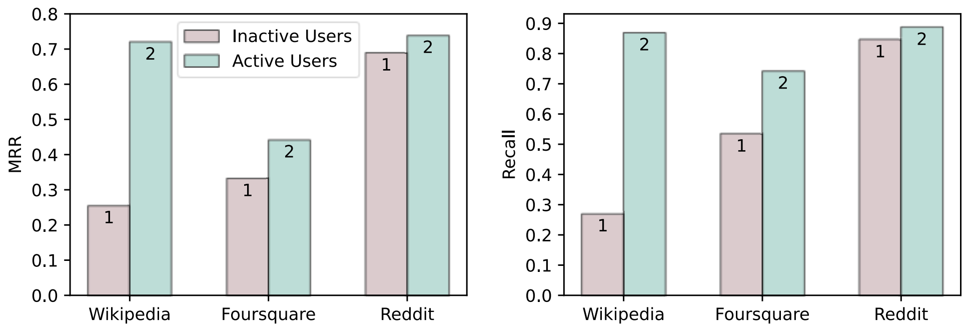


Figure 5.3: The overall performance over inactive user group (group 1) and active user group (group 2).

Recall@10. Second, it could be observed that compared with the Wikipedia and Foursquare datasets, the improvement on the Reddit dataset is relatively small. One possible reason is that the baselines have already achieved high performance on that dataset, e.g., the Recall@10 of DGEL is 0.849. Thus, the room for improvement is limited. However, our CREAM still significantly outperforms all baselines on the Reddit dataset. Third, we find that the two categories, deep recurrent learning and temporal graph learning, exhibit their own strengths. For example, NeuFilter and HILI are superior than temporal graph-based methods in some metrics, while DGEL can still outperform all deep recurrent baselines in the left metrics. On one hand, temporal graph learning fully investigates the temporal historical information from historical interactions. On the other hand, deep recurrent learning explores the newly-arrived interactions. Our proposed CREAM, which handles both historical and new interactions by ADC and TEL modules, exactly takes full advantage of the two categories. As a result, CREAM has achieved remarkable performance and significant improvement among all methods.

In addition, we report the performance over different user groups. We divide the user interactions into two groups: **inactive user group** and **active user group**, by treating the bottom 30% users with the fewest interactions as inactive users. The performance of the inactive user group is also our concern since we hope our model could generate accurate predictions for all users, not only for the active users. The results are shown in Figure 5.3.

It is observed that there is a significant performance gap on the Wikipedia dataset. However, the inactive user group still gets a certain performance via our CREAM. For the Foursquare and Reddit datasets, our CREAM performs quite well for the inactive user groups, whose performance is much closer to that of the active user groups. It indicates that all user groups could get satisfactory predictions by our proposed CREAM. This proves the effectiveness of our model for inactive users with fewer interactions.

Table 5.3: Ablation study on the key modules of CREAM. We separately remove the warm initialization (WR), the adaptive dual-side corrector (ADC), and the temporal excitement learning (TEL) from our CREAM. Also, we consider only leveraging the historical interactions (only History) or the newly-arrived interactions (only New) to update representations. Note that the warm initialization cannot be applied (NA) to the Foursquare dataset since it contains no features.

Dataset	**Wikipedia**		**Foursquare**		**Reddit**	
Metric	**MRR**	**Recall@10**	**MRR**	**Recall@10**	**MRR**	**Recall@10**
w/o WR	0.694	0.833	NA	NA	0.722	0.865
w/o ADC	0.641	0.792	0.407	0.693	0.706	0.852
w/o TEL	0.687	0.828	0.412	0.705	0.714	0.860
only New	**0.720**	0.771	0.386	0.629	0.668	0.704
only History	0.673	0.825	0.415	0.702	0.718	0.863
CREAM	0.704	**0.842**	**0.436**	**0.725**	**0.730**	**0.876**

5.4.3 Ablation Study

To investigate the effectiveness of each core module of our CREAM, we conduct the ablation study. We consider the following ablative variants: (1) **w/o WR**, which removes the warm initialization on temporal representations; (2) **w/o ADC**, which removes the adaptive dual-side corrector on historical interactions; (3) **w/o TEL**, which removes the temporal excitement learning on newly-arrived interactions; (4) **only New**, which only leverages new interactions to update representations; and (5) **only History**, which only leverages historical interactions to update representations. Table 5.3 lists the results.

It could be observed that the performance slightly decreases when dropping the warm initialization. This suggests the benefit of flexibly generating the temporal representations instead of random initialization. The adaptive dual-side corrector, as our core motivation, notably influences the performance of our framework. From the results, we can see that when we remove the corrector, the performance significantly decreases, which proves the importance of our designed ADC module on dynamically handling the inappropriate interactions for representation update. In addition, removing the temporal excitement learning would also weaken the performance, which demonstrates the effect of exploring newly-arrived interactions. Moreover, compared with the Foursquare and Reddit datasets, the Wikipedia dataset seems more sensitive to the corrector since its performance gets severely destroyed when removing the corrector. This may be because the Wikipedia dataset is the smallest one among all datasets. Therefore, its performance is unstable and could be easily affected by a few of the historical interactions.

Another interesting observation in the Wikipedia dataset is that its MRR would increase (from 0.704 to 0.720) when historical interactions are removed. However, considering the considerable decrease in Recall (from 0.842 to 0.771), we cannot replace our CREAM with that ablative version for the Wikipedia dataset. On the contrary, the Wikipedia dataset requires more operations to improve the historical information, which further verifies the previous finding that our proposed ADC would deliver more improvement on the Wikipedia dataset. For the Reddit and Foursquare datasets that contain many historical interactions, only utilizing historical interactions contributes much more to the performance than only leveraging new interactions. For example, over the Reddit dataset, the Recall would severely get destroyed from 0.876 to 0.704 when removing historical interactions. This suggests the power of historical information on representation learning when there is a large number of historical interactions.

5.4.4 Variant Exploration

To explore more possibilities of our CREAM, we conduct experiments on different feasible variants of CREAM: (1) **CREAM-1**, which directly adopts the linear network to compute the correction threshold; (2) **CREAM-2**, which leverages normal distribution to simulate the temporal excitement; and (3) **CREAM-3**, which overall predicts the combination of temporal and static representations. Here, we introduce the equation details of those variants:

- **CREAM-1**: which directly adopts the linear network to compute the correction threshold. We refine the Eq. 5.9 by:

$$\theta(i) = \mathbf{w}_{\text{td}}^{u}\mathbf{h}_i^{t^-}, \theta(j) = \mathbf{w}_{\text{td}}^{v}\mathbf{h}_j^{t^-}, \tag{5.22}$$

  where $\mathbf{w}_{\text{td}}^{u}$ and $\mathbf{w}_{\text{td}}^{v}$ are the learnable parameters. Compared with Eq. 5.9, Eq. 5.22 removes the supportive threshold-aware embeddings and directly outputs the thresholds based on the temporal representations.

- **CREAM-2**, which leverages normal distribution to simulate the temporal excitement. We refine the Eq. 5.15 and Eq. 5.16 by:

$$\varphi_p(\Delta t, v_j|u_i) = \mathbf{N}(\Delta t|0, \varepsilon_p), \varphi_n(\Delta t, v_j|u_i) = \mathbf{N}(\Delta t|0, \varepsilon_n), \tag{5.23}$$

$$\varepsilon_p = Sig(\mathbf{w}_{\text{nd}}^{1}\mathbf{h}_i^{t^-} + \mathbf{w}_{\text{nd}}^{2}\mathbf{h}_{j|p}^{\Delta t}), \tag{5.24}$$

$$\varepsilon_n = Sig(\mathbf{w}_{\text{nd}}^{3}\mathbf{h}_i^{t^-} + \mathbf{w}_{\text{nd}}^{4}\mathbf{h}_{j|n}^{\Delta t}), \tag{5.25}$$

Table 5.4: Variant exploration on CREAM. We consider three feasible variants: (1) CREAM-1 utilizing the linear network for correction threshold; (2) CREAM-2 using normal distribution for temporal effect; and (3) CREAM-3 predicting the combination of both temporal and static representations.

Dataset	**Wikipedia**		**Foursquare**		**Reddit**	
Metric	**MRR**	**Recall@10**	**MRR**	**Recall@10**	**MRR**	**Recall@10**
CREAM-1	0.672	0.815	0.403	0.701	0.719	0.865
CREAM-2	0.686	0.830	0.420	0.716	0.726	0.873
CREAM-3	0.659	0.746	0.391	0.684	0.693	0.852
CREAM	**0.704**	**0.842**	**0.436**	**0.725**	**0.730**	**0.876**

where ε_p and ε_n represents the standard deviations of normal distribution for positive and negative excitements, respectively. $\mathbf{w}^1_{\text{nd}}$ and $\mathbf{w}^2_{\text{nd}}$, $\mathbf{w}^3_{\text{nd}}$ and $\mathbf{w}^4_{\text{nd}}$ are learnable parameters. Function **N** is the normal distribution. Thus, taking the time interval Δt as input, we output the excitements by normal distribution with personalized standard deviations.

- **CREAM-3**, which roughly predicts the combination of temporal and static representations. We refine the Eq. 5.20 by:

$$\mathbf{S}^{t^+} = \mathbf{w}^1_{\text{pre}}\mathbf{h}^{t^+} + \mathbf{w}^2_{\text{pre}}\mathbf{h}^t_j + \mathbf{w}^3_{\text{pre}}\bar{\mathbf{h}}_i + \mathbf{w}^3_{\text{pre}}\bar{\mathbf{h}}_j, \tag{5.26}$$

where $\mathbf{w}^1_{\text{pre}}$, $\mathbf{w}^2_{\text{pre}}$, $\mathbf{w}^3_{\text{pre}}$, $\mathbf{w}^4_{\text{pre}}$ are the learnable parameters. $\mathbf{S}^{t^+}$ are the prediction on the final combination of the temporal and static representations. This is different from first leveraging temporal representation to refine static part and then concatenating them together.

The results are reported in Table. 5.4. It is observed that directly computing the threshold based on temporal representation would sacrifice the performance to some extent, which suggests the strength of introducing learnable threshold-aware embeddings. They indicate the user and item sensitivities on the threshold, and thus further help us examine the dynamic attributes of users and items. Moreover, we find that leveraging normal distribution would slightly decrease the performance. The explanation is that the specific function of trend simulation may cause over-fitting. However, it is still competitive with the MLP-based framework. Last, predicting the combination of the temporal and static representations would deliver a huge drop in performance and destroy the prediction task. This demonstrates the power of our new strategy for generating prediction. We carefully utilize predicted temporal representation to refine the static representation in a fine-grained way. In a word, compared with the different variants, our CREAM achieves the best performance for future interaction prediction.

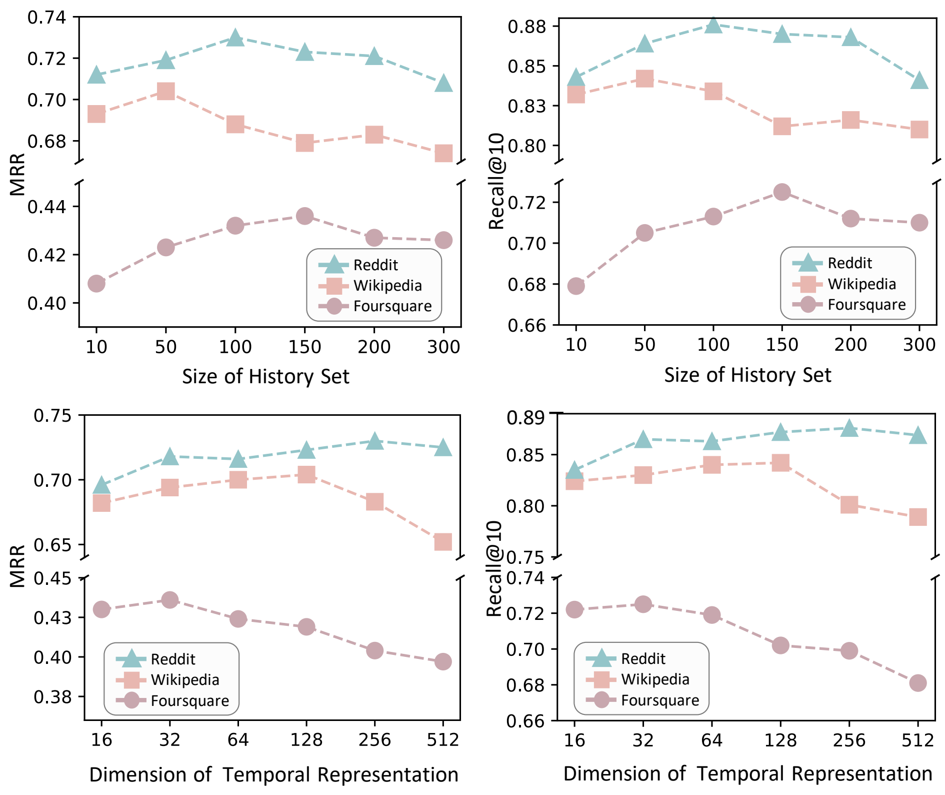


Figure 5.4: Hyperparameter sensitivity on performance: (1) the size of the history set, which is searched from $\{16, 32, 64, 128, 256, 512\}$ and (2) the dimension of temporal representation, which is searched from $\{10, 50, 100, 150, 200, 300\}$.

5.4.5 Hyperparameter Sensitivity

To further explore the sensitivity of CREAM to the important hyperparameters, we study how the two hyperparameters, the dimension of temporal representation and the size of history set, affect the performance of CREAM.

The results are shown in Figure 5.4. The size of the neighbor set is searched from 10 to 300, while the dimension of temporal representation is selected from 16 to 512. Clearly, increasing the history size would enhance the interaction prediction, especially on the Foursquare dataset. All datasets have their peaks. Along with the growth of history size, the performance of Wikipedia and Reddit datasets first goes up and then significantly decreases, reaching the peak at the small sizes, i.e., 50 and 100, respectively. However, the Foursquare achieves the optimal performance when the size is larger, i.e., 150. We attribute this to the dataset characteristic where the average of user interaction in the Foursquare dataset is the largest. Thus, compared with Wikipedia and Reddit datasets, the Foursquare dataset requires a larger size to precisely conduct representation learning.

Similarly, the model performance could be improved by increasing the dimension of temporal representation. Specifically, our CREAM achieves the best performance on the Wikipedia and Reddit datasets when the dimensions are 128 and 256, respectively. The Reddit dataset may have a potential chance to get minor improvement with dimensions larger than 512. However, it is unnecessary to pursue extremely-large dimensions. As for the Foursquare dataset, its optimal dimension is relatively small, i.e., 32. Compared with the Foursquare dataset, the Wikipedia and Reddit datasets contain less historical information for each user. Therefore, they need more dimensions to better learn the temporal representations.

5.4.6 User Case Study

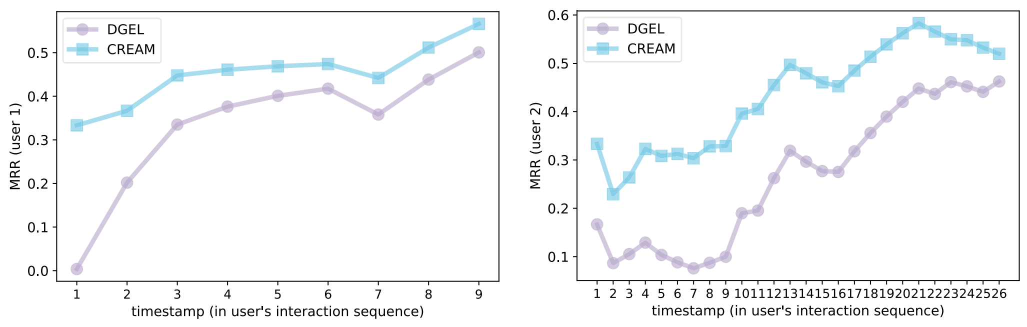


Figure 5.5: User case study. We show the performance gap between our proposed CREAM and the SOTA method DGEL over two users.

We conduct user case study to show the performance gap between our proposed CREAM and the SOTA method DGEL. The case study is conducted on the Wikipedia dataset and we select two users. The one has interactions lower than average while the other one has interactions higher than average. We anonymize their user IDs by User 1 and User 2. The case study illustrates the varying performance over time, which is calculated by all previous prediction results at the current timestamp. The results are shown in Figure 5.5.

From the results, we can observe that the accumulated performance of both DGEL and CREAM would gradually increase over time. Even though there are some fluctuations, especially for User 2, the performance still shows upward trends. This indicates that such temporal methods would better learn user preferences as more user interactions are received. More importantly, we can see that the proposed CREAM is constantly superior to the DGEL. The two users, who have opposite levels of interactions compared with the average, are gaining better performance via our CREAM. This indicates the superiority of our

CREAM in generating item predictions for users. Some users may get better predictions via the DGEL, but from the overall performance comparison in Table 5.2, our proposed CREAM significantly outperforms the DGEL and other baselines.

5.5 Summary of CREAM

In this chapter, we propose a novel dynamic self-correcting interaction network named CREAM, which aims to accurately and continuously predict interactions between users and items over time. Along this line, we carefully design the adaptive dual-side corrector to dynamically correct temporal histories of both the user side and item side over historical interactions, and the temporal excitement learning to comprehensively capture the effects of the newly-arrived interactions. Meanwhile, we present a warm initialization to flexibly initialize both temporal and static representations. In addition, we present a novel strategy which smoothly utilizes temporal and static representations together to conduct predictions on future interactions more precisely. Extensive experiments on three public datasets demonstrate both the superiority and effectiveness of our proposed CREAM.

CHAPTER 6

DUAL-SIDE ONLINE FAIRNESS LEARNING FOR RECOMMENDATION

Fairness in recommendation has drawn much attention since it significantly affects how users access information and how information is exposed to users. However, most fairness-aware methods are designed offline with the entire stationary interaction data to handle the global unfairness issue and evaluate their performance in a one-time paradigm. In real-world scenarios, users tend to interact with items continuously over time, leading to a dynamic recommendation environment where unfairness is evolving online. Moreover, previous methods that focus on mitigating the unfairness can hardly bring significant improvements to the recommendation task. Hence, in this chapter, we propose a **M**odel-agnostic **D**ual-side **O**nline **Fair**ness Learning method (MDOFair) for the dynamic recommendation. First, we carefully design dynamic dual-side fairness learning to trace the rapid evolution of unfairness from both the user and item sides. Second, we leverage the fairness and recommendation tasks in one utilized framework to pursue the double-win success. Last, we present an efficient model-agnostic post-ranking method for the dynamic recommendation scenario to mitigate the dynamic unfairness. Extensive experiments demonstrate the superiority and effectiveness of our proposed MDOFair by incorporating it into existing dynamic TGL models as a post-ranking stage.

The content in this chapter mainly focuses on addressing the research challenge **RC 4** and overcoming the limitation **EL 4** by proposing a novel method MDOFair.

6.1 Brief Introduction of MDOFair

Recommendation systems that alleviate information overload are ubiquitous around the world. These systems support personalized decision-making in various applications, such as e-commerce (e.g., Amazon and Taobao) and social medias (e.g., Facebook and Instagram) [125, 124, 77, 129]. Recently, fairness, which concerns equal opportunities for item exposure [34] or equal quality of user service [46], has gained significant attention because of its profound impact on how people access information and how information is exposed to users [115]. There has been a growing awareness that recommendation systems are subject to algorithmic fairness considerations along different dimensions, including stakeholder, type of benefit, context, and even morality [24, 15, 59, 145, 70]. Therefore, it is imperative to give careful and strict consideration towards fairness in recommendation systems to ensure the stability and development of the marketplace.

A series of studies have been proposed in the field of fairness for recommendation. For instance, item-side methods re-allocate the exposure of items based on their popularity to protect item fairness, such as FELIX [34] and FULTR [112]. Some popular items that have many interactions with users are fully learned by recommendation models. As a result, they are more likely to be recommended in the recommendation list, thereby gaining exposure. On the contrary, other items that lack interactions with users (e.g., new items), cannot get sufficient exposure by recommendation models. Consequently, the popular items would be more popular and get more exposure, while the unpopular items would be more unpopular and get less exposure. In fact, many items are users' truly-preferred items, but they are just not exposed to users.

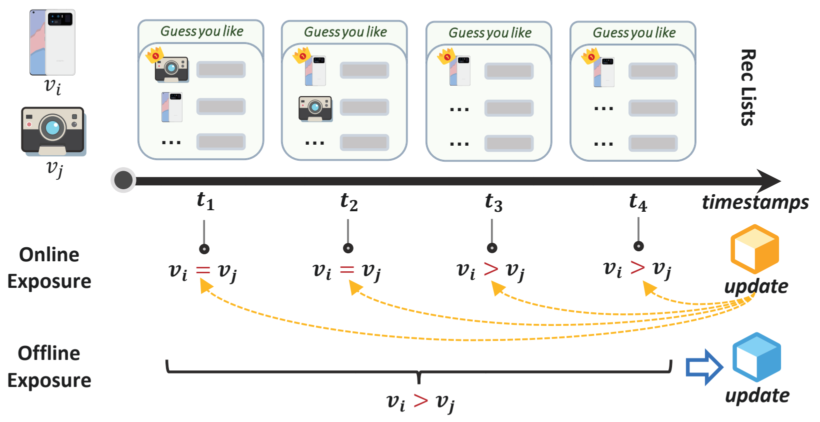
On the other hand, user-side methods provide more support for learning disadvantaged users' feedback to eliminate the disparity among advantaged and disadvantaged user groups, e.g., UGF [46] and FairRec [97]. They claim that the quality of the recommendation service should not depend on personal data, such as users' activity level. For example, an inactive user could also get good recommendations that are inferred based on only a few interactions from that user. Furthermore, dual-side methods, including TFROM[99] and CPFair[60], address fairness issues for both users and items simultaneously, aiming to find an optimal balance between them.

Although remarkable success has been achieved in fairness studies, there are still some challenges that remain to be solved. The first one is *dynamic evolution* of online fairness. Many previous fairness methods are designed for offline use to handle the global unfairness issue derived from the stationary data. However, in real-world scenarios, users tend to interact with items continuously over time [102, 116, 139], leading to a dynamic environment where recommendation system would update user preference and refresh recommendation result immediately after receiving each feedback [44, 58, 101]. This is called dynamic recommendation [78]. Consequently, online fairness is evolving rapidly due to the consecutive interaction between users and items [121, 78]. Thus, those offline fairness methods that require sufficient data and only conduct one-time static evaluations fail to handle the dynamic evolution of online fairness. Take two items shown in Figure 6.1(a) as example. From timestamp t_1 to t_2, the two items v_i and v_j have the same exposure since they are both exposed in the recommendation list. They are relatively fair to each other from the perspective of item-side exposure. After that, v_j are removed from the recommendation list while v_i still get high exposure. This could be caused by many reasons, e.g., the popular items would be sufficiently captured and learned by recommendation system and thus have a higher chance to dominate the recommendation list [58]. Then, at timestamp t_3 and t_4, it tends to be unfairer for the non-exposure item v_j compared with over-exposed item v_j. Fortunately, the online fairness-aware method would update its fairness policy dynamically to adapt to the current environment at different timestamps, ensuring the evolving fairness between items. However, the offline fairness evaluation would only computes the global fairness without tracing its fast evolution.

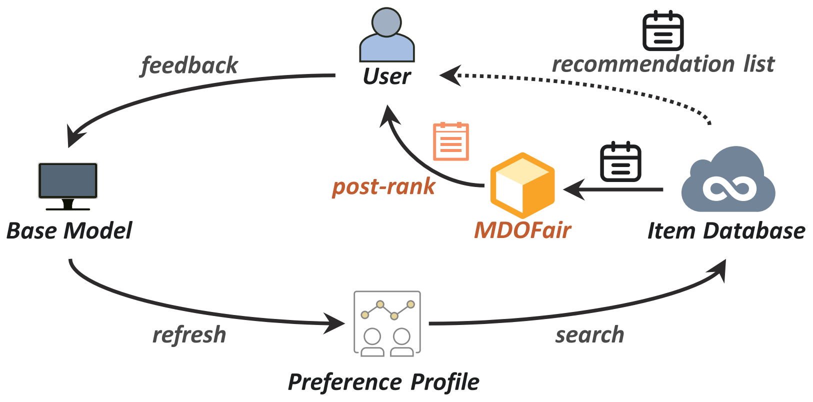
The second challenge is the *trade-off balance* problem. Most previous fairness methods primarily focus on mitigating unfairness and maintaining recommendation performance simultaneously [34, 46, 60, 49, 17]. Namely, they cannot deliver significant improvements to the recommendation task. They simply treat the balance problem as the fairness trade-off between the user and item sides, explicitly neglecting the fact that introducing the fairness task is essentially to strengthen the recommendation task [91]. Therefore, we argue that the trade-off balance problem manifests in two aspects:

(1) **The Fairness Optimization Balance** between the user side and the item side. Helping less popular items rank higher in the recommendation list may pose the risk of destroying user personalization [115, 72], while eliminating the disparities among users may cause inappropriate distribution of item exposure [97, 99].

(2) **The Multi-task Combination Balance** between the recommendation task and the fairness task. In general, it is challenging to achieve significant improvements in both fairness and recommendation simultaneously [60, 87], as the superior performance of the recommendation system naturally depends on those advantaged groups who may cause unfairness to disadvantaged groups [46, 47, 63]. Especially in a dynamic environment, the rapid evolution of fairness brings more uncertain dynamics into the recommendation task



(a). A toy example of online fairness and offline fairness between two items v_i and v_j from the perspective of item-side exposure over time.



(b). The dynamic online recommendation procedure with and without our proposed MDOFair.

Figure 6.1: The illustration of online fairness for dynamic multi-round recommendation system.

[24].

To address these aforementioned challenges, i.e., the dynamic evolution of online fairness and the multi-dimensional trade-off balance, we should tackle the following problems:

(1) *How to dynamically trace the evolution of dual-side unfairness for users and items.* The core of online fairness is to capture its rapid evolution during the continuous recommendation procedure. Moreover, learning online fairness should not be affected by the dynamic recommendation models themselves, ensuring generality and effectiveness.

(2) *How to adaptively utilize online fairness to correct unfair results.* In order to produce fair recommendation lists continuously, it is necessary to adjust the ranking scores based on the observed online fairness without affording expensive computational costs.

(3) *How to leverage online fairness and dynamic recommendation in one unified framework*. The trade-off balance between the user and item sides promotes fair recommendation in the online environment, while the trade-off balance between the fairness task and the recommendation task yields a comprehensive preeminent recommendation system. Propping up the disadvantaged items and users (e.g., unpopular items and inactive users) does not mean we would negatively affect the recommendation accuracy. Fairness and performance are both important. It would be a double-win success if we handle the unfairness issue while not sacrificing the recommendation performance for the whole system.

To this end, we propose a **M**odel-agnostic **D**ual-side **O**nline **Fair**ness Learning method (MDOFair) for dynamic recommendations. The objective of our MDOFair is to dynamically mitigate the online unfairness issue for both users and items while improving the system performance in dynamic recommendations. *Firstly*, we design dynamic online unfairness learning for the user side and item side, respectively. This enables us to explore and capture the rapid evolution of online fairness. *Secondly*, we inject the online fairness task into the dynamic recommendation task to obtain the double-win success of the whole system. *Finally*, we propose an efficient model-agnostic post-ranking method, which could be seamlessly integrated with any recommendation model (e.g., the proposed model DGEL in Chapter 3) by adaptively re-ranking recommendation lists based on fairness. The recommendation models generate initial recommendations, while our MDOFair revises the recommendation results in the post-ranking stage. The post-ranking paradigm for dynamic recommendation is shown in Figure 6.1(b). Before returning the current recommendation list to the target user, we adopt the proposed MDOFair to effectively adjust it according to the online dual-side fairness. This procedure would keep running until the system is shut down. Consequently, we mitigate the dynamic online unfairness issue while significantly improving the performance of dynamic recommendation.

Our contributions are highlighted as follows:

- **A New Problem.** To the best of our knowledge, we are the first to investigate the online dual-side unfairness from both user-side and item side for the continuous dynamic recommendation, where the fairness is rapidly evolving due to the consecutive interactions between users and items.
- **A Novel Online Learning Method for Fairness**. We carefully design a dual-side online unfairness learning method for both users and items to dynamically trace their evolution during the online environment.
- **A New Model-Agnostic Post-Ranking Approach**. We propose MDOFair, which is an

efficient model-agnostic post-ranking approach, to mitigate the dual-side online unfairness issue while improving the performance of dynamic recommendation significantly.

- **Extensive Experiments.** We conduct extensive and comprehensive experiments on three real-world datasets by extending our proposed MDOFair to existing dynamic models as a re-ranking stage. The experimental results demonstrate both the superiority and effectiveness of our MDOFair.

6.2 Definitions

Suppose the recommendation system involves N users and M items with the user set $\mathscr{U} = \{u_1, u_2, ..., u_N\}$ and the item set $\mathscr{V} = \{v_1, v_2, ..., v_M\}$. At timestamp t, the user-item interaction could be denoted as tuple $e^t_{ij} = (u_i, v_j, t)$, where $i \in [1, N]$ and $j \in [1, M]$. All timestamped interactions form the sequential stream $\mathscr{I}$. Following the prior work [78, 60], we define the dynamic recommendation and online fairness goal.

Definition 1 (Dynamic Recommendation). As shown in 6.1(b), given an online system, the recommendation model requires learning the current user-item interaction e^t_{ij} between user u_i and item v_j at time t and generating a new recommendation list to satisfy u_i's evolving preference and hit its future interaction $e^{t^+}_{ij}$ at time t^+. This iterative process continues until the system is shut down. We formalize the process as follows:

$$\mathrm{Rec}^{t^+}_{u_i} = \{v_x | v_x \in \mathscr{V}, \arg\max_{v_x}^{K} v_x \sim \mathscr{F}(e^t_{ij}, u_i, v_j)\}, \tag{6.1}$$

where K is the length of the recommendation list and $\mathscr{F}$ is the dynamic model that would learn the evolving preference from current e^t_{ij}. It prepares a refreshed recommendation list $\mathrm{Rec}^{t^+}_{u_i}$ for u_i for the next time he (or she) accesses the system.

Definition 2 (Dynamic Fairness). Given the above dynamic multi-round recommendation environment, we focus on the dual-side fairness learning in an online real-time manner, i.e., continuously mitigating the exposure discrepancy among items and the performance disparity among users. For the current timestamp t. the dynamic fairness is formally defined as follows:

$$\mathrm{D}^t_{\mathrm{exp}} = \mathrm{avg}(|\mathrm{Exp}^t(v_j) - \mathrm{Exp}^t(v_g)|), v_j, v_g \in \mathscr{V}, \tag{6.2}$$

$$\mathrm{D}^t_{\mathrm{qua}} = \mathrm{avg}(|\mathrm{Qua}^t(u_i) - \mathrm{Qua}^t(u_l)|), u_i, u_l \in \mathscr{U}, \tag{6.3}$$

where $\mathrm{D}^{t}_{\mathrm{exp}}$ measures the current fairness of temporal exposure for items while $\mathrm{D}^{t}_{\mathrm{qua}}$ is the current fairness of recommendation quality for users. Functions $\mathrm{Exp}^{t}()$ and $\mathrm{Qua}^{t}()$ derive the temporal exposure and recommendation quality from the dynamic system, respectively. We would realize them in our method. avg() returns the average. Due to the ongoing user-item interactions in dynamic environment, $\mathrm{D}^{t}_{\mathrm{exp}}$ and $\mathrm{D}^{t}_{\mathrm{qua}}$ are evolving over time, providing us with opportunities to post-process the recommendation results in real-time.

Definition 2 (Online Fairness-aware Ranking). After obtaining $\mathrm{Rec}^{t^+}_{u_i}$ and learning $\mathrm{D}^{t}_{\mathrm{exp}}$ and $\mathrm{D}^{t}_{\mathrm{qua}}$, the post-ranking stage would be triggered to adaptively re-rank all items in the list based on the current online fairness to fulfill some particular criteria. We formalize the fairness-aware ranking as follows:

$$\mathrm{Rec}^{t^+}_{u_i} \xrightarrow{\Theta} \mathrm{FairRec}^{t^+}_{u_i}, \tag{6.4}$$

where Θ is the post-ranking strategy that utilizes both $\mathrm{D}^{t}_{\mathrm{exp}}$ and $\mathrm{D}^{t}_{\mathrm{qua}}$ and $\mathrm{FairRec}^{t^+}_{u_i}$ is the final revised recommendation list that mitigate the online unfairness. Since the dynamic recommendation procedure is running over time, the online dual-side fairness learning would be continuously updated. As a consequence, we provide constant fairness-aware post-ranking at each timestamp, which dynamically mitigates the unfairness for the dynamic recommendation system.

6.3 MDOFair Method Design

In this section, we present the details of our proposed MDOFair to handle the evolving unfairness issues and generate a continuous post-ranking strategy for the dynamic recommendation. MDOFair consists of three parts: **(1) Dynamic Exposure-Oriented Unfairness**: an online item-side fairness learning method based on items' accumulating exposure over time. **(2) Dynamic Quality-Oriented Unfairness**: an online user-side fairness learning method based on recommendation quality obtained by users over time. **(3) Model-agnostic Fairness-aware Post-ranking**: an adaptive online re-ranking strategy to mitigate the dual-side unfairness and also improve the recommendation performance. Our proposed MDOFair would not change the dynamic model itself (e.g., $\mathscr{F}$ in Eq. 6.1). In contrast, we focus on the model-agnostic post-ranking (as shown in Eq. 6.4) to create generality and extensibility for different dynamic recommendation platforms.

6.3.1 Dynamic Exposure-oriented Unfairness

6.3.1.1 Exposure Estimation of Items

An ideal distribution of items' exposure in the ranking list is the first and foremost step to ensure the fairness of recommendation [60, 14]. Suppose at timestamp t, a user u accesses our recommendation system and obtains a recommendation list Rec^t. For each item in this list, its ranking position is determined by its relevance with u. In an online environment, the items beyond the top-K position actually cannot get any exposure. Thus, according to [12], we assume that the user u views top-K results from top to bottom and estimate the expected exposure of clicking item v_j at the position k as follows:

(6.5)
$$p^t_{v_j} = \begin{cases} r^t_{uv_j} \prod\limits_{x=1}^{k-1} (1 - r^t_{uv_j}), & k \le \mathrm{Top}K \\ \epsilon, & k > \mathrm{Top}K \end{cases},$$

where $r^t_{uv_j}$ indicates the current relevance (probability) of item v_j to user u, which is computed by the basic dynamic recommendation model. Here, we set a minimum ϵ for those items beyond the Top-K position to avoid zero error of division.

6.3.1.2 Accumulative Online Exposure

As users access our recommendation system continuously, item exposure may evolve over time accordingly. For each item at timestamp t, considering the online computational efficiency, we directly accumulate exposure that the item has gained so far. For instance, v_j's accumulative online exposure is defined as:

(6.6)
$$\exp^t_{v_j} = \sum_{\tau \le t} p^\tau_{v_j},$$

where the exposure estimation could be derived from Eq. 6.5. This accumulative online exposure, which requires no additional training-based parameter, is recorded in real time. For fair exposure comparison between different items, we also use the average to denote it: $\overline{\exp}^t_{v_j} = \frac{1}{t}\exp^t_{v_j}$.

6.3.1.3 Online Relevance Merit

In addition to online exposure, items' relevance is also evolving over time. It reflects the advantage of taking a higher position in the recommendation list, which is called merit [115]. However, most studies focus on the group merit of items that share the same group attribute, neglecting the individual merit. Hence, we define the online relevance merit as the total relevance that the item has gained so far. For example, v_j's online exposure merit is shown as follows:

$$\text{merit}_{v_j}^{t} = \sum_{\tau \leq t} r_{u_* v_j}^{\tau}, \tag{6.7}$$

where the relevance score $r_{u_* v_j}^{\tau}$ is derived from the basic dynamic model and u_* refers to the previous users who received a recommendation list by our service. Similarly, this online relevance merit is recorded in real time without additional parameters. For fair merit comparison between different items, we also use the average to denote it: $\overline{\text{merit}}_{v_j}^{t} = \frac{1}{t}\text{merit}_{v_j}^{t}$.

6.3.1.4 DEU: Dynamic Exposure-oriented Unfairness

For dynamic recommendation, imposing seemingly fair decisions through static criteria can lead to unexpected unfairness in the long run. In other words, online fairness cannot be defined in a static one-shot setting without considering the temporal impact [24]. Since we have already established accumulative online exposure and online relevance merit of each item, we can realize the online unfairness mitigation in such a dynamic environment.

Specifically, we extend the Disparity of Treatment criterion of [72] and [59] to the dynamic recommendation scenario. For any two individual items v_j and v_g, the exposure-oriented disparity between them at timestamp t is fulfilled:

$$\text{Dis}^{t}(v_j, v_g) = \left| \frac{\overline{\text{exp}}_{v_j}^{t}}{\overline{\text{merit}}_{v_j}^{t}} - \frac{\overline{\text{exp}}_{v_g}^{t}}{\overline{\text{merit}}_{v_g}^{t}} \right|. \tag{6.8}$$

This **dynamic exposure-oriented disparity** at the current time indicates whether the two items obtain appropriate exposure proportional to their expected relevance over past timestamps. Obviously, it would be updated dynamically due to the change of accumulative online exposure and online relevance merit.

The further the disparity is from zero, the greater the violation of fairness. Now, to capture and trace the unfairness at the current timestamp t, we define the **Dynamic Exposure-oriented Unfairness (DEU)** for the dynamic recommendation system as follows:

$$\text{DEU}^{t} = \frac{M(M-1)}{2} \sum_{j \leq M} \sum_{g<j} \text{Dis}^{t}(v_j, v_g), \tag{6.9}$$

where M is the size of the item set. Compared with the static offline evaluation in Figure 6.1(a), we successfully learn the dynamic online item-side unfairness, which is evolving continuously over time.

6.3.2 Dynamic Quality-oriented Unfairness

6.3.2.1 Historical Online Quality

From the perspective of users, the basic recommendation model dynamically exploits users' evolving preferences over time and continuously updates the recommendation lists based on users' ongoing history of consumption [78, 77]. This leads to rapid-varying quality of online service for users. For example, the target user u_i may get low-quality service at earlier timestamps due to limited interaction data, while receiving high-quality service after a certain time. The quality could be defined as any evaluation metric of recommendation, e.g., Recall, NDCG, and MRR. In this chapter, we simply select MRR to denote online quality. Thus, at timestamp t, u_i's historical online quality is defined as:

$$\mathrm{Qua}_{u_i}^t = \frac{1}{\mathcal{H}_{u_i}^t} \sum_{v \in \mathcal{H}_{u_i}^t} \begin{cases} \mathrm{MRR}^t(v), & v \in \text{ top}-K \text{ list} \\ 0, & \text{otherwise} \end{cases}, \tag{6.10}$$

where $\mathcal{H}_{u_i}^t$ is the current history of items that u_i has interacted with. For the item v that the user has interacted at timestamp t, the $\mathrm{MRR}^t(v)$ represents the corresponding MRR of v in the recommendation list of timestamp t. In general, the historical items have different contributions to user's online experience. We average the online quality of them by $\mathrm{Qua}_{u_i}^t$, to denote u_i's satisfaction at timestamp t. Additionally, items beyond top-K position cannot bring any satisfaction.

6.3.2.2 DQU: Dynamic Quality-oriented Unfairness

The online unfairness issue should be considered from not only the item side but also the user side because they both affect the marketplace significantly [46, 97]. A fair recommendation system would equally treat all users and provide satisfactory service to them, instead of favoring those advantaged users (e.g., some active users may have more chance to get high-quality service than inactive users). To achieve this, we consider the disparity between users in terms of historical online quality. For any two individual users u_i and u_l, the quality-oriented disparity between them at timestamp t is fulfilled:

$$\mathrm{Dis}^t(u_i, u_l) = \left|\mathrm{Qua}_{u_i}^t - \mathrm{Qua}_{u_l}^t\right|. \tag{6.11}$$

This **dynamic quality-oriented disparity** at the current time measures the gap between the two users in terms of the rapid-varying recommendation quality they have obtained over past timestamps.

Eventually, in order to investigate the unfairness issue from the user side at the current timestamp t, we define the **Dynamic Quality-oriented Unfairness (DQU)** for the whole system as follows:

$$\mathrm{DQU}^t = \frac{N(N-1)}{2} \sum_{i \leq N} \sum_{l<i} \mathrm{Dis}^t(u_i, u_l), \tag{6.12}$$

where N is the size of the user set. In addition to the item-side unfairness, we can now achieve online user-side unfairness learning. Obviously, this unfairness would also evolve continuously over time. In the next section, we will present the model-agnostic fairness-aware post-ranking to mitigate the dual-side online unfairness issue and improve the recommendation performance simultaneously.

6.3.3 Model-agnostic Fairness-aware Post-ranking

Previous studies on fairness encounter the trade-off balance problem, which contains two aspects: (1) the dual-side fairness balance between users and items, and (2) the multi-task balance between fairness and recommendation. However, most of them cannot bring significant improvement in recommendation performance when addressing unfairness issues. Especially in a dynamic environment, the rapid evolution of fairness would pose more uncertain dynamics to the recommendation task [24]. Thus, in this chapter, we aim to carefully overcome the multi-dimensional balance problem to achieve double-win success.

On one hand, inspired by [60, 46, 99], we formalize the dual-side online fairness task as follows:

$$\mathrm{FairTask}: \min \sum_{\tau \leq |\mathscr{I}|} (\alpha \mathrm{DEU}^\tau + (1-\alpha)\mathrm{DQU}^\tau), \tag{6.13}$$

where $\mathscr{I}$ is sequential stream in our scenario (see section 6.2). α is a core hyperparameter to balance the item and user sides. Under its impact, the recommendation system would continuously mitigate the dual-side unfairness during the online procedure. A larger α is suitable for item-centered recommendation platforms, while a smaller α may perform better for user-centered recommendation platforms.

On the other hand, the recommendation task in a dynamic scenario can be defined as follows:

$$\mathrm{RecTask}: \max \sum_{\tau \leq |\mathscr{I}|} r^t_{u^*, v^*} \tag{6.14}$$

where $r^t_{u^*,v^*}$ is the relevance score of the interaction between user u^* and item v^*. Note that we would not change (or enhance) the design of the dynamic recommendation model itself. Our objective is to help the online system mitigate unfairness and improve performance by introducing our fairness-aware post-ranking. Therefore, this relevance score is determined by the basic dynamic model, e.g., Jodie [40], DGCF [44], TREND [96], and DGEL [78]. Now, we are ready to conduct the multi-task optimization in one unified framework:

$$\max \sum_{\tau \le |\mathscr{T}|} (r^t_{u^*,v^*} - \alpha \mathrm{DEU}^\tau - (1-\alpha)\mathrm{DQU}^\tau). \tag{6.15}$$

In Eq. 6.15, we have a trade-off balance problem between dual-side fairness, and between the fairness task and the recommendation task. Due to the fact we would not change the dynamic model itself, we have to make full use of the online fairness to achieve the multi-dimensional balance.

We still take the user u_i as an example. At timestamp t, user u_i accesses our system and receives the recommendation list $\mathrm{Rec}^t_{u_i}$. It is generated by ranking items' relevance scores from the item set V. According to Eq. 6.4, we would search a post-ranking list $\mathrm{FairRec}^t_{u_i}$ for the basic $\mathrm{Rec}^t_{u_i}$, to dynamically solve the optimization in Eq. 6.15. However, considering the online computational efficiency and response requirement, it is unrealistic to examine all possible post-ranking results to find the best one, especially when $|V|$ is large.

Thus, for the sake of fast and efficient post-ranking, inspired by some outstanding studies [60, 87, 39, 26], we first construct two adaptive online coefficients for both item and user sides:

• **Item Fairness-aware coefficient**. For a candidate item v_j at timestamp t, if it occupies more exposure, it should be more responsible for the exposure-oriented unfairness:

$$\omega^t_{v_j} = \frac{\exp^t_{v_j}}{\max(\{\exp^t_v |, v \in V\})}. \tag{6.16}$$

• **User Fairness-aware coefficient**. If the target user u_i at timestamp t is marked as a disadvantaged user, the system should compensate for his (or her) online experience through the candidate item v_j.

$$\omega^t_{u_i|v_j} = \mu^t_{u_i} \frac{\mathrm{CNT}(u_i, v_j, t)}{\max(\{\mathrm{CNT}(u_i, v, t) | v \in V\})}, \tag{6.17}$$

$$\mu^t_{u_i} = \begin{cases} \frac{1}{\ln(N)}, & \mathrm{Qua}^t_{u_j} \ge \Delta \\ -\frac{1}{\ln(N)}, & \mathrm{Qua}^t_{u_j} < \Delta \end{cases}, \tag{6.18}$$

where function $\mathrm{CNT}(\cdot)$ counts the number of interactions between the user and the item before the current time. We use the $\mathrm{CNT}(\cdot)$ instead of directly leveraging the preference score predicted by dynamic recommendation models. The reasons are: (1) Different users have already been treated unequally by those dynamic models and the unfairness exists in their predicted score, and (2) our goal is to design a model-agnostic method to mitigate existing unfairness. If we leverage the predicted score from dynamic models to calculate the user-side coefficient, it would be still unfair and then make it difficult to help disadvantaged users.

Thus, we utilize the user actual interaction information (CNT) for the user-side coefficient, which is not determined by the dynamic models. $\mu^t_{u_i}$ indicates whether u_i is a disadvantaged user by comparing its online quality with Δ. Here, we use the average online quality of all users to denote Δ.

From the above item-side and user-side coefficients, we derive two conclusions: (1) Items with higher exposure should be punished more to protect those lower-exposure items. (2) Disadvantaged users with lower online quality should receive support when using the service, and the magnitude of such support is determined by the user size. In addition, these two coefficients are dynamically updated because users would interact with items continuously over time. Returning to $\text{Rec}^t_{u_i}$, for the candidate item v_j, we refresh its basic relevance score as follows:

$$\hat{r}^t_{u_i v_j} = r^t_{u_i v_j} - \alpha \cdot \omega^t_{v_j} \text{DEU}^t - (1-\alpha) \cdot \omega^t_{u_i|v_j} \text{DQU}^t. \tag{6.19}$$

We use Eq. 6.19 to revise the scores of all candidate items and then re-rank them to get a novel recommendation list $\text{FairRec}^t_{u_i}$. In this way, we fulfill our post ranking fast and efficiently without examining all possible ranking results, which could still mitigate online unfairness and improve the performance of the system. In addition, as an independent post-ranking method, it would not be affected by the basic recommendation model.

At each service time, we use our adaptive model-agnostic post-ranking method to handle the continuous unfairness issue for the dynamic recommendation. Different from previous studies, our proposed method is totally designed for online systems, which is more practical for industry applications. At last, We present the online procedure of our proposed MDOFair in Algorithm 5 with the limited interaction stream $\mathscr{I}$. Algorithm 5 follows the Figure 6.1(b). Such an online procedure would keep running until the system is shut down.

6.3.4 Complexity Analysis

We propose the MDOFair for the dynamic recommendation scenario as a post-ranking stage. At each service time, it takes the worst $\mathcal{O}(M^2)$ and $\mathcal{O}(N^2)$ to compute the DEU^t and

Algorithm 5: Dynamic Recommendation with MDOFair

```
Input: $\mathscr{U}, \mathscr{V}, \mathscr{I}$, basic $\mathscr{F}$
Output: Fair post-ranking list at each service time
1  for $t = 1$ to $|\mathscr{I}|$ do
2      Current accessing user: $u$;
       // Basic ranking from dynamic model $\mathscr{F}$
3      Compute relevance scores $R_u^t$ between $u_i$ and $\mathscr{V} \leftarrow \mathscr{F}$;
4      Get the basic unfair ranking list $\mathrm{Rec}_u^t$;
       // The dual-side online unfairness at $t$
5      Compute $\mathrm{DEU}^t$ and $\mathrm{DQU}^t$ by Eq. 6.9 and Eq. 6.12;
6      for $v = 1$ to $|\mathscr{V}|$ do
           // Adaptive dual-side fairness coefficients
7          Update $\omega_v^t$ and $\omega_{u|v}^t$ by Eq. 6.16 and Eq. 6.18;
           // Fairness-aware relevance score
8          Revise $r_{uv}^t \in R_u^t$ to $\hat{r}_{uv}^t$ by Eq. 6.19;
9      end
       // Post-ranking
10     Get a novel post-ranking list $\mathrm{FairRec}_u^t$;
11     return $\mathrm{FairRec}_u^t$ to user $u$;
12     $\mathscr{F}$ learns the actual feedback $\mathscr{I}^t$ for user $u$;
13 end
```

DQU^t if we simply adopt loops to update the exposure-oriented and quality-oriented unfairness. Next, it takes $\mathcal{O}(M)$ to refresh the scores of candidate items. Last, we could apply a sorting algorithm (e.g., Quicksort) to output the new recommendation list, which may require $\mathcal{O}(M \log M)$. In practice, the complexity of obtaining DEU^t and DQU^t could be compressed to $\mathcal{O}(M)$ and $\mathcal{O}(N)$ if we calculate the disparities in parallel. Thus, it is efficient to apply our MDOFair as the post-ranking method with acceptable cost.

6.3.5 Strengths of MDOFair

From the methodology section, we can conclude that our MDOFair dynamically learns the evolving unfairness and continuously post-processes the dynamic recommendation results based on the learned fairness. Here, we discuss the strengths of MDOFair to highlight our novelty.

Firstly, our MDOFair is designed for the dynamic multi-round recommendation. This is a practical scenario in most real-world online platforms where users are continuously interacting with the systems. The environment is changing over time due to ongoing interactions. We would generate real-time recommendation list at each timestamp while mitigating its unfairness.

Secondly, all the information, e.g., the temporal exposure of items and the historical quality of users, are being continuously accumulated and updated. They are determined by historical interactions and then they influence the current unfairness learning. Consequently, previous timestamps would affect future timestamps. We endow MDOFair with the capability of perceiving temporal information, and then dynamically derive the rapid evolution of fairness. More importantly, we focus on dual-side fairness learning to mitigate the unfairness for both users and items in one unified framework.

Thirdly, we post-process the recommendations based on captured dynamic unfairness, instead of directly predicting recommendations. We assign the item-side unfairness and the user-side unfairness with adaptive weights, respectively. Then we revise the ranking scores generated by recommendation models to achieve better and fairer ranking results. It maintains the recommendation capability of those models while mitigating the dynamic unfairness for them. This rule-based post-processing strategy ensures the generalizability and flexibility of our framework. We can integrate it with any dynamic model and implement it into any dynamic system.

6.4 Experiments on MDOFair

In this section, we conduct extensive experiments on three real-world datasets to evaluate both the superiority and effectiveness of our proposed MDOFair. We aim to answer the following questions through experiments.

- **RQ1**: Can our proposed MDOFair mitigate the dynamic online unfairness while improving the performance of dynamic recommendation?
- **RQ2**: How do the key components of online fairness learning (i.e., DEU^t and DQU^t) contribute to our MDOFair and recommendation system?
- **RQ3**: Can our MDOFair consistently handle the continuous online unfairness issues throughout the recommendation system's running?
- **RQ4**: How does the core hyperparameter (i.e., α) influence the trade-off balance between dual-side fairness and between multiple tasks?
- **RQ5**: What is the computational cost of the post-ranking MDOFair compared with the basic ranking?

Table 6.1: Statistics of the experimental datasets.

Dataset	User	Item	Interaction	Bans	Repetition
Wikipedia	8,227	1,000	157,474	366	61%
Reddit	10,000	1,000	672,447	217	79%
LastFM	980	1,000	1,293,103	-	8.6%

6.4.1 Experimental Settings

6.4.1.1 Datasets

We benchmark the MDOFair method on three public datasets collected from different real-world platforms, whose properties are introduced below:

- **Wikipedia**. This dataset consists of one month of edits on Wikipedia. 157,474 interactions are filtered out for recommendation by the editors who made at least 5 edits and the 1,000 most edited pages.
- **Reddit**. This dataset contains one month of user posts on subreddits. We select the 1,000 most active subreddits as items and the 10,000 most active users, resulting in 672,447 interactions.
- **LastFM**. This dataset is made of one month of who-listens-to-which song information. We selected all 1000 users and the 1000 most listened songs, resulting in 1,293,103 interactions.

The statistics of these datasets are presented in Table 6.1. In addition, data sources are available[1].

6.4.1.2 Baselines

Our MDOFair is an adaptive model-agnostic post-ranking method with dual-side online fairness learning, which is applied after the basic ranking results. Hence, to construct the basic ranking stage, we select four outstanding dynamic recommendation models:

- **Jodie** [40]: A temporal interaction network that continuously predicts the dynamic embeddings by real-time update and projection operations.
- **DGCF** [44]: A dynamic method leveraging the dynamic graphs to directly learn the collaborative and sequential information at the same time.

[1]https://snap.stanford.edu/jodie/

- **TREND** [96]: A novel framework for temporal graph representation learning driven by Hawkes process-based graph neural network to capture the temporal events and node dynamics.
- **DGEL** [78]: A state-of-the-art evolutionary model that comprehensively captures the rapid evolution of user and item dynamics over time with the learning-based re-scaling network.

We extend these basic recommenders with our MDOFair. For example, the DGEL that applies MDOFair as the post-ranking stage could be denoted as **FairDGEL**. We would compare DGEL and FairDGEL in terms of online fairness and recommendation performance. Similarly, we also have **FairJodie**, **FairDGCF**, and **FairTREND**.

6.4.1.3 Experiment Setups

For each dataset, the user-item interactions are arranged in chronological order. Then, we split the training, validation, and test in a proportion of 80%, 10%, and 10%. The training and validation are regarded as an offline stage. We train these dynamic recommendation models, and according to validation, we select their optimal epochs as recommenders. The offline training is based on the temporal batching method [78]. After the offline training stage, we obtain the basic recommenders.

In this chapter, we focus on the online dynamic unfairness learning for the online stage. Thus, the test data are used for simulating the online environment. We regard each test interaction as the individual service time and continuously generate basic recommendation lists by recommendation model. At each service time, we adopt our MDOFair to re-rank the basic list. Then, we evaluate both the basic ranking list and our post-ranking list by the ground-truth item of the current interaction. This process would be repeated until the stream of test data ends. Note that during the online stage, we do not leverage test interactions to re-train the recommendation models.

6.4.1.4 Evaluation Metrics

We adopt Recall@10 and MRR to evaluate recommendation performance and use our proposed DEU^t and DQU^t to evaluate online unfairness. Furthermore, we design two other metrics as follows:

$$\mathrm{mDEQU} = \mathrm{DEU}^t \times \mathrm{DQU}^t, \tag{6.20}$$

$$\text{C–score} = \frac{\text{Recall@10}}{\text{mDEQU}}, \tag{6.21}$$

where mDEQU is the dual-side unfairness and C–score denotes the comprehensive performance considering both fairness task and recommendation task.

All the metrics above would be computed at each individual service timestamp. For the research questions RQ1, RQ2 and RQ4, we report their average of all timestamps. A recommendation system would be much superior and fairer if it has higher Recall@10, MRR, C–score, and lower DEU, DQU, mDEQU.

6.4.1.5 Reproducibility

The offline training of these dynamic baselines follows the suggested parameter settings in their original chapters. For our online inferring procedure, we create a separate chunk to import our MDOFair for post-ranking. The core hyperparameter α is selected from 0.0 to 1.0, where $\alpha = 0.0$ ignores the item-side fairness learning and $\alpha = 1.0$ waives user-side fairness learning. The minimum ϵ is set to 0.0001. Last, all the experiments are implemented using Pytorch and executed with an NVIDIA 3090 GPU.

6.4.2 Overall Performance

In this section, we report the overall performance of our proposed MDOFair in terms of recommendation task and fairness task in Table 6.2. Moreover, we present the comprehensive performance combining both of them. From the results, we summarize the following observations.

Recommendation Performance. Our proposed MDOFair improves the performance of dynamic recommendation on both Recall@10 and MRR across all datasets, especially yielding significant and surprising improvements on Wikipedia and LastFM datasets. For instance, the Recall@10 of TREND on the LastFM dataset increases from 0.2914 to 0.4060 (39.3%↑) after applying our MDOFair as a post-ranking strategy. Also, on the Wikipedia dataset, FairJodie wins a 19.6% gain from basic Jodie, where Recall@10 increases from 0.6845 to 0.8193. These astounding results provide strong evidence that our MDOFair still remains applicable to the recommendation task. The improvement on the Reddit dataset is relatively small. One of the possible reasons is that the basic ranking performance is already high (e.g., The Recall@10 of DEGL is 0.8411). However, all MDOFair-based baselines consistently outperform their basic unfair versions.

Online Fairness. Our proposed MDOFair successfully mitigates the online dynamic unfairness for both user and item sides in terms of DEU and DQU across all datasets. Obvi-

Table 6.2: The overall performance of MDOFair in terms of fairness and recommendation. The baselines, Jodie, DGCF, TREND and DGEL, are used for the basic ranking. Then, we extend these baselines with our MDOFair, which is an adaptive model-agnostic post-ranking method. Thus, we have FairJodie, FairDGCF, FairTREND and FairDGEL. We report the comparison results with and without MDOFair.

Dataset	Model	Recommendation		Fairness		Comprehensiveness	
		Recall@10↑	MRR↑	DEU↓	DQU↓	mDEQU↓	C-score↑
Wikipedia	**Jodie**	0.6829	0.6370	0.0235	0.4735	0.0111	61.52
	FairJodie	0.8165	0.7248	0.0200	0.4601	0.0092	88.75
	Gain	**19.5%**	**13.7%**	**14.8%**	**2.8%**	**17.1%**	**44.2%**
	DGCF	0.7012	0.6402	0.0220	0.4718	0.0104	67.42
	FairDGCF	0.8387	0.7425	0.0184	0.4558	0.0083	101.05
	Gain	**19.6%**	**15.9%**	**16.3%**	**3.4%**	**20.1%**	**49.8%**
	TREND	0.7051	0.6048	0.0154	0.4707	0.0072	97.93
	FairTREND	0.8395	0.6920	0.0131	0.4568	0.0059	142.28
	Gain	**19.0%**	**14.4%**	**14.9%**	**3.0%**	**18.0%**	**45.2%**
	DGEL	0.7411	0.6617	0.0159	0.4681	0.0074	100.14
	FairDGEL	0.8446	0.7209	0.0141	0.4537	0.0063	134.06
	Gain	**13.9%**	**8.9%**	**11.3%**	**3.1%**	**17.5%**	**33.8%**
Reddit	**Jodie**	0.8029	0.6614	0.0278	0.4224	0.0117	68.62
	FairJodie	0.8133	0.6806	0.0202	0.4073	0.0082	99.18
	Gain	**1.3%**	**2.9%**	**27.3%**	**3.6%**	**29.9%**	**44.5%**
	DGCF	0.8102	0.6459	0.0192	0.4350	0.0083	97.61
	FairDGCF	0.8490	0.6788	0.0169	0.4133	0.0070	121.28
	Gain	**4.7%**	**5.1%**	**11.9%**	**4.9%**	**18.5%**	**24.2%**
	TREND	0.8197	0.6642	0.0173	0.3997	0.0069	118.79
	FairTREND	0.8528	0.7138	0.0161	0.3861	0.0062	137.54
	Gain	**4.0%**	**7.5%**	**6.9%**	**3.4%**	**10.1%**	**15.7%**
	DGEL	0.8461	0.6976	0.0153	0.3858	0.0059	143.40
	FairDGEL	0.8659	0.7170	0.0120	0.3775	0.0045	192.42
	Gain	**2.3%**	**2.8%**	**21.5%**	**2.1%**	**23.7%**	**34.1%**
LastFM	**Jodie**	0.2830	0.1899	0.0111	0.2142	0.0023	123.04
	FairJodie	0.3883	0.2262	0.0102	0.2004	0.0020	194.15
	Gain	**37.2%**	**19.1%**	**8.1%**	**6.4%**	**13.0%**	**57.8%**
	DGCF	0.3218	0.2098	0.0149	0.2213	0.0033	97.51
	FairDGCF	0.3806	0.2275	0.0122	0.2077	0.0025	152.24
	Gain	**18.2%**	**8.4%**	**18.1%**	**6.1%**	**24.2%**	**56.1%**
	TREND	0.2914	0.1998	0.0096	0.2207	0.0021	138.76
	FairTREND	0.4060	0.2384	0.0086	0.2155	0.0018	225.55
	Gain	**39.3%**	**19.3%**	**10.4%**	**2.3%**	**14.2%**	**62.5%**
	DGEL	0.2810	0.1943	0.0107	0.2171	0.0023	122.17
	FairDGEL	0.3849	0.2250	0.0097	0.2013	0.0019	202.58
	Gain	**36.9%**	**15.8%**	**9.3%**	**7.2%**	**17.3%**	**65.8%**

ously, the online unfair issue is determined by the recommendation scenario and selected model. On the LastFM dataset, the DEU of DGCF and DGEL are 0.0149 and 0.0107, respectively. This result suggests that DGEL is fairer than DGCF in this scenario. However, DGEL's DEU reaches 0.0153 on the Reddit dataset. Furthermore, it is observed that compared with item-side DEU, the reduction gain of user-side DQU is relatively smaller. For example, on the Wikipedia dataset, FairTREND reduces 14.9% DEU from TREND while only lessening 3.0% DQU. We attribute this to the inherent difficulty of satisfying all users' service-quality demands. To sum up, our MDOFair effectively helps dynamic recommendation systems mitigate online unfairness for both users and items.

Discussion on Improvement. We could conclude that our MDOFair not only mitigates the unfairness issue, but also yields remarkable recommendation performance. Here, we discuss the reasons for the performance improvement. The first reason is that, MDOFair is designed as a post-processing method to be integrated with existing dynamic recommendation models. Those models first generate basic recommendation results and then our MDOFair revises them based on the fairness. The two-stage recommendation procedure provides a powerful double guarantee for the system performance. A large amount of performance comes from the dynamic models themselves and the further performance is delivered by MDOFair. For example, the Recall@10 of FairDGEL on the Wikipedia dataset is 0.8446 where DGEL has already achieved 0.7411 and our MDOFair has further achieved 0.1035 improvement. In other words, the combination of in-processing recommendation models and post-processing fairness method could achieve optimal recommendation results.

The second reason is that, when we post-process the basic recommendation results, we efficiently leverage dual-side fairness learning and directly modify the ranking scores based on that. Not only the item-side exposure unfairness has been investigated, but user-side quality unfairness has been also studied. We mitigate unfairness issue while protecting the recommendation quality. The dual-side fairness learning comprehensively contributes to our method. More importantly, we directly modify the ranking scores based on the learned dual-side unfairness. This direct score revision would explicitly change the positions of candidate items. In addition, our MDOFair, could be implemented into any dynamic system and be integrated with any dynamic model. Different from in-processing mechanisms, MDOFair requires no tailored loss or training, thereby freeing it from the limitations of scenarios, data, basic models, and recommenders. This suggests the generalizability and flexibility of our MDOFair. For these datasets with poor recommendation performance (e.g., the basic average Recall@10 of the LastFM dataset is 0.2943), there is a huge room of per-

Table 6.3: The ablation study of MDOFair. We examine the key components of online fairness learning in our MDOFair by (1) removing the item-side fairness (w/o I), (2) removing the user-side fairness (w/o U), and (3) only adopting static dual-side fairness (Static). For each MDOFair-based recommendation model, we compare it with the ablated variants.

Model	**Wikipedia**			**Reddit**			**LastFM**		
	Recall@10↑	**mDEQU↓**	**C-score↑**	**Recall@10↑**	**mDEQU↓**	**C-score↑**	**Recall@10↑**	**mDEQU↓**	**C-score↑**
FairJodie	0.8165	0.0092	88.75	0.8133	0.0082	99.18	0.3883	0.0020	194.15
w/o I	0.6970	0.0104	67.02	0.8015	0.0113	70.92	0.3609	0.0019	189.94
w/o U	0.6896	0.0095	72.59	0.7946	0.0084	94.60	0.2888	0.0023	125.56
Static	0.7583	0.0097	78.16	0.8107	0.0090	90.08	0.3592	0.0022	163.27
FairDGCF	0.8387	0.0083	101.05	0.8490	0.0070	121.28	0.3806	0.0025	152.24
w/o I	0.6694	0.0097	69.01	0.8136	0.0078	104.30	0.3614	0.0026	139.00
w/o U	0.7261	0.0088	82.51	0.8355	0.0072	116.04	0.3189	0.0029	109.96
Static	0.7639	0.0091	83.94	0.8308	0.0073	113.81	0.3573	0.0027	132.33
FairTREND	0.8395	0.0059	142.28	0.8528	0.0062	137.54	0.4060	0.0018	225.55
w/o I	0.6804	0.0068	100.05	0.8301	0.0067	123.89	0.3820	0.0019	201.05
w/o U	0.7258	0.0061	118.98	0.8371	0.0063	132.87	0.2933	0.0019	154.36
Static	0.7531	0.0063	119.54	0.8224	0.0067	122.74	0.3796	0.0021	180.76
FairDGEL	0.8446	0.0063	134.06	0.8659	0.0045	192.42	0.3849	0.0019	202.58
w/o I	0.6757	0.0070	96.52	0.8513	0.0057	149.35	0.3626	0.0019	190.84
w/o U	0.7419	0.0064	115.92	0.8471	0.0045	188.24	0.2851	0.0021	135.76
Static	0.7802	0.0066	118.21	0.8587	0.0047	182.70	0.3562	0.0020	178.10

formance improvement, leaving our MDOFair with the opportunity to fulfill it.

6.4.3 Ablation Study

To understand the contribution of the key components of online fairness learning in MDOFair, we study the following ablated variants: (1) **w/o I**, which only considers user-side online fairness for the post-ranking; (2) **w/o U**, which only learns item-side online fairness for the post-ranking; (3) **Static**, which derives static dual-side fairness from offline training stage. For each MDOFair-based recommendation model, we compare it with the two types of ablated variants, in terms of Recall@10, mDEQU, and C-score. The results are presented in Table 6.3.

Item-side Online Fairness Learning. From the results, it is observed that item-side online fairness has a significant impact on the Wikipedia and Reddit datasets, but its impact is relatively small on the LastFM dataset. The C-score of FairJodie without item side decreases from 77.29 to 54.77 on the Wikipedia dataset while decreasing from 194.15 just to 189.94 on the LastFM dataset. Similarly, the C-score of FairDGEL without item side decreases from 192.42 to 149.35 on the Reddit dataset while just decreasing from 202.58 to 190.84 on the LastFM dataset. One possible reason is that Wikipedia and Reddit are item-centric platforms, i.e., entries from Wikipedia and tweets from Reddit. Thus, mitigating the unfairness from the item side may bring more gain for the recommendation system. However, LastFM is a music application where user personalization plays the dominant role.

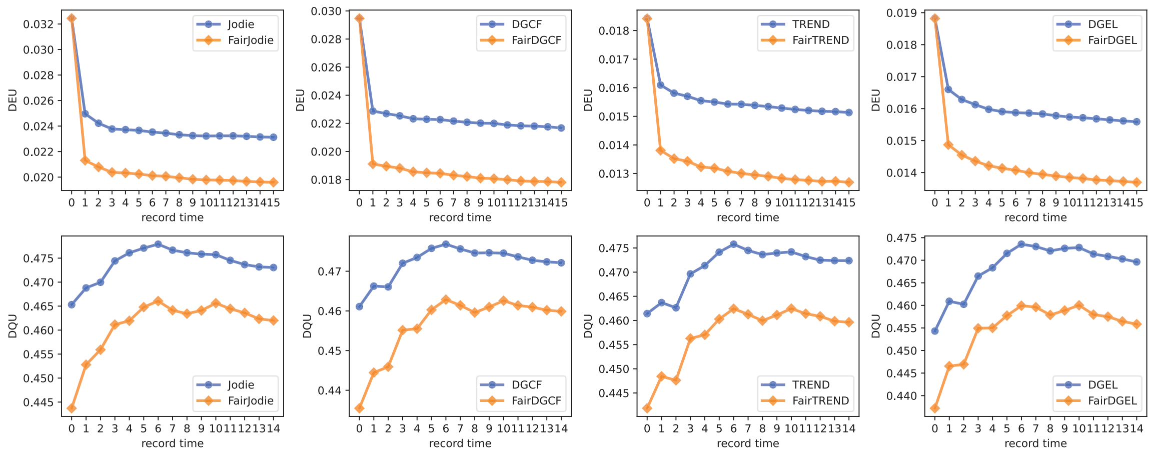
User-side Online Fairness Learning. Different from the item side, removing the user-side unfairness learning significantly reduces the performance on the LastFM dataset, but its reduction is small on the Wikipedia and Reddit datasets. For instance, the C-score of FairTREND without user side decreases severely from 225.55 to 154.36, while that without considering the item side just decreases to 201.05. However, on the Reddit dataset, the C-score of FairDGCF without considering the user side fairness decreases from 123.95 to 119.09, which seems insignificant because that without item side decreases to 91.16. Therefore, the user-side fairness learning has more impact on the LastFM dataset. Compared with Wikipedia and Reddit, LastFM is a user-centric platform, where user behavior is more susceptible to being determined by their own personalization. In this case, it would be better to pay more attention to user-side unfairness learning.

Static Dual-side Fairness Learning. Clearly, only deriving static fairness from offline training stage would decrease the comprehensive performance of MDOFair in terms of both recommendation and fairness, which indicates the importance of learning dynamic fairness in the online environment. For example, over the Wikipedia dataset, the C-score of static FairJodie decreases from 88.75 to 78.16. Similarly, over the LastFM dataset, the C-score of static FairDGCF decreases from 152.24 to 132.33. However, even though static fairness is learned on the offline stage, it still captures the dual-side unfairness for both users and items, and would have a certain positive effect on online re-ranking. Thus, the MDOFair based on static dual-side fairness, is still competitive with the MDOFair that only utilizes user-side or item-side learning. The static fairness learning could be an alternative to MDOFair when we aim to provide fairness-aware recommendation without causing additional computational overload for the dynamic recommendation system.

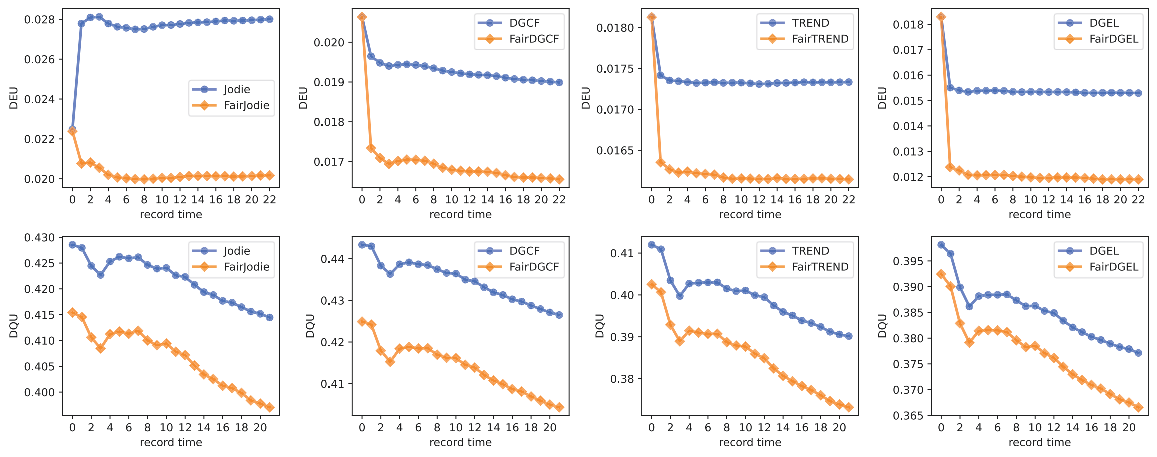
6.4.4 Dynamic Visualization

To verify that our MDOFair could continuously mitigate the unfairness issue during the dynamic recommendation procedure, we visualize the trends of DEU and DQU by plotting them every 1,000 timestamps on Wikipedia, 3,000 timestamps on Reddit. Due to the page limitation, we omit LastFM. The results are shown in Figure 6.2.

Continuous Unfairness Issue of Items. The unfairness issue of items persists throughout the entire runtime of the recommendation system. Except for the DGCF on the Reddit dataset, all baselines present a declining trend, which may reflect their ability for autonomous mitigation of item unfairness. In addition, item-side unfairness changes a lot in the early times. However, our MDOFair significantly helps those unfair baselines obtain a drastic reduction in item-side unfairness. This demonstrates our MDOFair is capable of



(a). Wikipedia

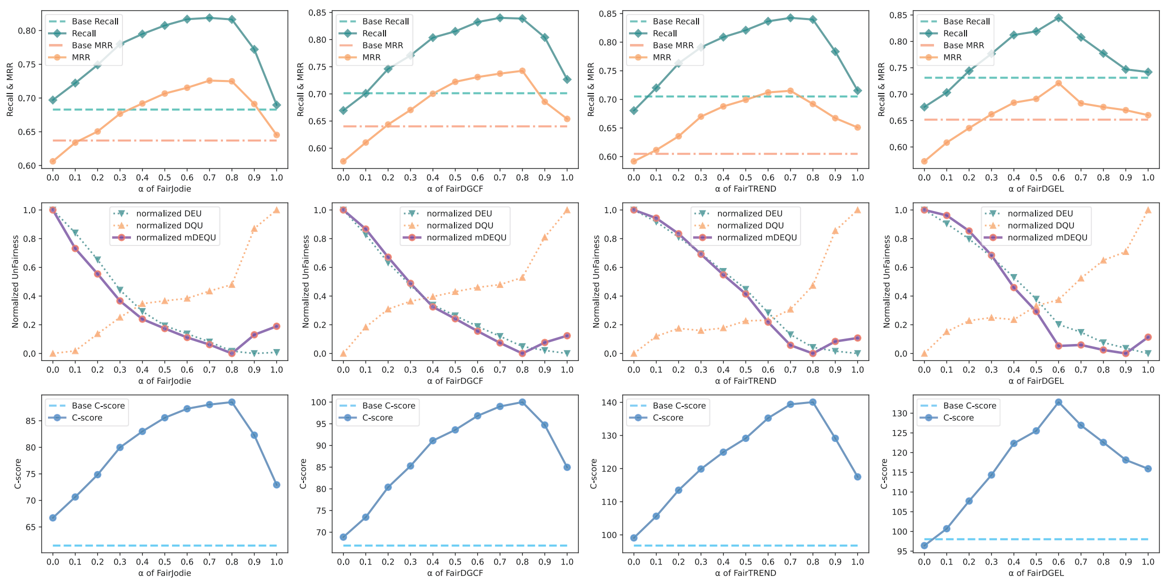


(b). Reddit

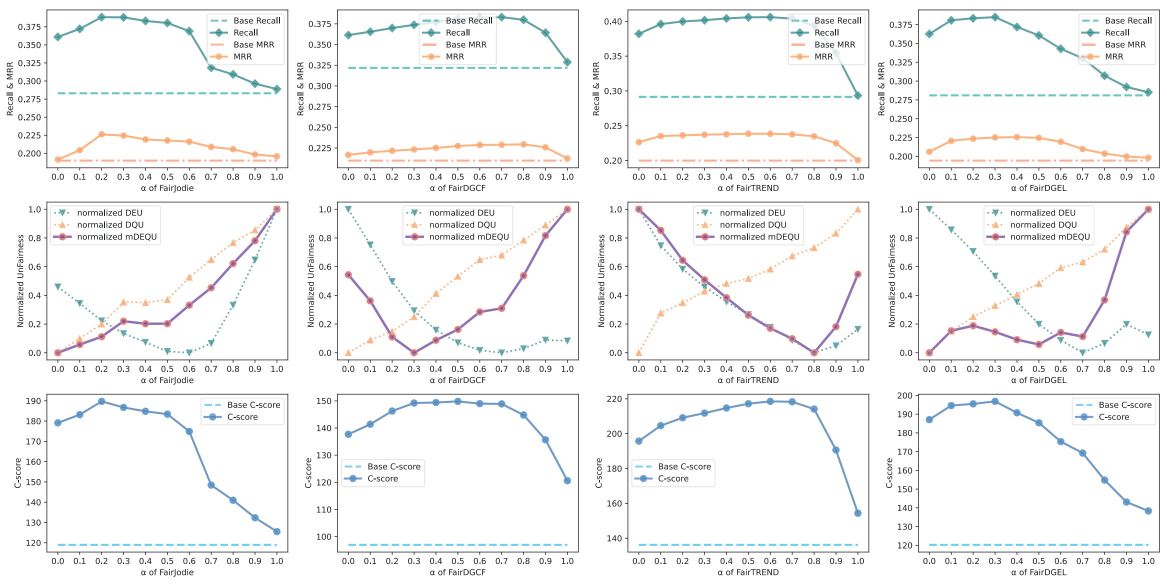
Figure 6.2: The illustration of continuous online unfairness issue during dynamic recommendation procedure.

continuously mitigating the unfairness issue for items during the dynamic recommendation procedure.

Continuous Unfairness Issue of Users. We observe that on the Wikipedia dataset, the user side is getting increasingly unfair over time, while on the Reddit dataset, the user unfairness is getting reduced constantly. This could be explained by the characteristics of the dataset. The average interaction of the Reddit dataset is considerably more than that of the Wikipedia dataset. Thus, on the Reddit dataset, the dynamic baselines are more prone to learn the disparity between users and react to them. Nevertheless, in all cases, our MDO-Fair noticeably mitigates the continuous unfairness issue from unfair baselines, attaining a fairer dynamic recommendation system.



(a). Wikipedia



(b). LastFM

Figure 6.3: The illustration of the online fairness and the performance of dynamic recommendations with the varying core hyperparameter α, which controls the multi-dimensional trade-off balance between dual-side fairness.

6.4.5 Hyperparameter Study

The core α would affect the multi-dimensional trade-off balance problem from (1) Daul-side Fairness Balance between users and items, and (2) Multi-task Balance between online fairness and dynamic recommendation. We carefully tune the α from 0.0 to 1.0 for the

MDOFair-based baselines. The results are reported in Figure 6.3. In addition, we report the original Recall@10, MRR, and C-score of baselines without our MDOFair, e.g., Base Recall. It is worth noting we normalize DEU, DQU, and mDEQU for fair comparison.

Dual-side Fairness Balance. In most cases, the item-side DEU goes down with the growth of α while the user-side DQU increases gradually, and the dual-side mDEQU first declines and then presents upper trend. However, there are some exceptions. For example, the mDEQU of FairJodie on the LastFM dataset constantly climbs upwards. Moreover, its DEU first decreases and then begins to rise. In general, the bottom of the mDEQU trend may represent the optimal fairness balance between users and items. Obviously, a large α is suitable for the Wikipedia dataset, while a relatively smaller α is an ideal choice for the LastFM dataset (except FairTREND). This is because LastFM is a user-centric music application where paying more attention to the user side (i.e., smaller α) would bring more improvement in fairness for the whole system.

Multi-task Balance for Recommendation System. It is observed that with the growth of α, the performance of recommendation (i.e., Recall and MRR) and the comprehensiveness of the system (i.e., C-score) first go up and then decrease on all datasets and all MDOFair-based baselines. In addition, selecting a smaller α for the Wikipedia dataset even causes worse performance than basic baselines without MDOFair. However, on the LastFM dataset, all determinations of α perform better than basic unfair baselines. Although it may be difficult to reach the best performance of recommendation while obtaining the fairest system, we could still achieve double-win success according to the comprehensiveness C-score. For instance, FairDGCE with $\alpha = 0.0$ is the fairest while it with $\alpha = 0.4$ reaches the best comprehensiveness due to the constraint of recommendation performance. Thus, our proposed MDOFair fulfills the trade-off balance between online fairness and dynamic recommendation, instead of excessive pursuit of fairness. Additionally, the suboptimal α for the fairness task still significantly mitigates the online unfairness issue for those unfair baselines (see Table 6.2), which also demonstrates the superiority of our MDOFair.

6.4.6 Efficiency Analysis

The recommendation models generate the basic recommendations while our MDOFair post-processes their results based on fairness learning. We report the their execution time of online inferring in Table 6.4.

Two key findings can be observed. First, our post-processing MDOFair takes much less computational cost than the basic recommendation models. Compared with the basic ranking stage, the post-processing stage is significantly more efficient. For example, over the

Table 6.4: Online inferring efficiency comparison between the basic ranking by recommendation models and the post ranking by MDOFair. The efficiency is measured in seconds.

Dataset	Ranking	Jodie	DGCF	TREND	DGEL
Wikipedia	**Basic**	95s	124s	116s	121s
	Post	47s	46s	45s	44s
Reddit	**Basic**	578s	746s	665s	772s
	Post	369s	370s	363s	360s
LastFM	**Basic**	697s	855s	787s	926s
	Post	324s	329s	330s	338s

Wikipedia dataset, the basic Jodie consumes 95s to generate recommendations while MDOFair consumes 47s for post-processing. Also, over the LastFM datasets, the computation cost of DGEL is 926s while that of MDOFair is only 338s. This suggests that, compared with the basic ranking stage, leveraging our MDOFair as post-processing stage would not cause heavy computational overhead to the dynamic recommendation systems. Second, no matter what recommendation model is selected as recommender, the cost of MDOFair is relatively stable, indicating the post-processing stage is model-agnostic. For instance, the post-processing cost over the Reddit dataset is around 365s across all recommenders. Thus, we could integrate it with any recommendation model, and implement it into any dynamic system.

6.5 Summary of MDOFair

In this chapter, we study the problem of dual-side online unfairness for the dynamic recommendation scenario where users tend to interact with items continuously, resulting in a dynamic online environment and rapid evolution of fairness. To this end, we propose an adaptive model-agnostic post-ranking method with dual-side online fairness learning, called MDOFair. Firstly, we carefully design two novel online fairness learning methods, dynamic item-side exposure-oriented unfairness and user-side quality-oriented unfairness, to continuously trace the evolution of online fairness for both users and items. Then, we leverage the online fairness task and dynamic recommendation task in one unified framework to achieve a double-win success for both unfairness mitigation and recommendation improvement. Finally, we propose an efficient model-agnostic post-ranking strategy to dynamically re-rank and update the recommendation list in real time. Extensive experiments on three real-world datasets demonstrate both the superiority and effectiveness of MDOFair.

Chapter 7

Conclusion and Future Work

This last chapter concludes the thesis and discusses the potential future work of dynamic recommendation. Specifically:

- Sec. 7.1 summarizes the key findings and contributions of the research in this thesis, confirming that all research challenges have been effectively addressed.
- Sec. 7.2 further discusses the potential future work of dynamic recommendation beyond the research in this thesis, with a primary focus on dynamically leveraging large language models for dynamic recommendation.

This chapter reviews the limitations/challenges of existing studies and concludes how the research in this thesis addresses them successfully, thus highlighting the contributions achieved. Furthermore, it explores the potential future research direction by investigating the possibility of incorporating large language models into the dynamic environment.

7.1 Conclusion

This thesis focuses on the dynamic recommendation where users are continuously interacting with items over time. The recommendation systems should dynamically update user/item representations and refresh recommendation results in a multi-round manner. Temporal graph learning, which continuously constructs user-item interaction graph based on temporal information, can naturally handle the update problem of dynamic RecSys. Chapter 1 discusses the dynamic recommendation in terms of the background, existing research limitations, the core unsolved challenges, and the contributions of this thesis. Chapter 2 thoroughly conducts the relevant literature review, including static/dynamic recommendation systems, deep recurrent learning, graph representation learning, and fairness in recommendation. Four major research challenges are derived and this thesis aims to address them.

Firstly, most existing TGL methods mainly utilize insufficient perspectives for representation update, which means they lack comprehensive capturing different dynamics from different perspectives. In addition to TGL, many conventional GNN methods are still limited to static one-time recommendation paradigm. Thus, the first challenge is *how to comprehensively leverage different update perspectives in one unified framework and effectively convert static graph learning into temporal graph learning for recommendation*. To this end, Chapter 3 proposes a novel unified framework DGEL which leverages multiple perspectives to update temporal representations, such as inherent interaction potential, time-decay augmentation, and symbiotic local structure learning. DGEL provides the solution to comprehensively capture dynamics and learn the evolution of users and items. Moreover, to bridge different perspectives with the whole model learning, it constructs the re-scaling network that adaptively normalizes different sub-embeddings.

Secondly, most previous studies on TGL tend to directly model the temporal evolution when new interaction is injected into the temporal graph, without carefully verifying whether the collaborative effect learned from the new interaction would truly benefit modeling the temporal evolution. Hence, the second challenge is *how well the newly-arrived interaction could match the temporal history and how much information should be learned according to the current context*. To this end, Chapter 4 designs a temporal collaboration-aware co-evolving graph model TCGC. It investigates the collaborative effect of newly-arrived interactions, and further devises the collaboration-aware indicator to guide the temporal graph learning. In addition, TCGC builds the co-evolving network, which is supported by the temporal indicator, to capture the correlation between event-level and history-level dynamics.

Thirdly, existing studies explicitly assume that all historical interactions would reflect the actual representation evolution when aggregating the historical neighbors over temporal graph by T-GNN. However, some historical interactions may be imperfect since they may not represent the users' actual long-term preferences. Therefore, the third challenge is *how to efficiently discover valuable interactions from long-term history and adaptively filter out those inappropriate interactions for improving update*. Compared with the second challenge, this challenge focuses on the long-term historical neighbors. To this end, Chapter 5 presents the CREAM, a dynamic self-correcting interaction network, which aims to accurately generate predictions over the temporal user-item interaction graph. CREAM constructs a novel adaptive dual-side corrector to dynamically correct the histories by filtering out inappropriate historical interactions in a long run. Meanwhile, CREAM presents a temporal excitement learning for new interactions to capture both positive and negative dynamics. In addition, CREAM provides a warm initialization that could flexibly initialize both temporal and static representations.

Fourth, there are the exposure disparity among items and the recommendation discrepancy among users, causing the unfairness issue to the dynamic RecSys. However, most TGL methods lack the consideration of mitigating the unfairness. As a result, the popular items would get more exposure than unpopular items and the active users would get better recommendation services than inactive users. Hence, the fourth challenge is *How to dynamically trace the dual-side unfairness over time and efficiently mitigate unfairness in a generalizable way without being restricted by models*. To this end, Chapter 6 proposes the MDOFair, an adaptive model-agnostic post-ranking method with dual-side online fairness learning. It devises dynamic item-side exposure-oriented unfairness and user-side quality-oriented unfairness, to trace the evolving state of the dynamic RecSys over time. Then it refreshes the recommendation results based on the learned dual-side unfairness for users and items. Moreover, MDOFair leverages the online fairness task and dynamic recommendation task in one unified framework, achieving a double-win success for both unfairness mitigation and recommendation performance.

In Chapter 3, Chapter 4, Chapter 5, and Chapter 6, extensive experiments are conducted over various real-world datasets. The experiments include overall performance comparison, ablation study, variant exploration, hyperparameter sensitivity, dynamic visualization, efficiency comparison, case study and so on. The experimental results demonstrate the superiority and effectiveness of the DGEL, TCGC, CREAM and MDOFair, fulfilling the aims of the research in this thesis.

7.2 Future Work

Recently, the Large Language Models (LLMs), which achieve superior performance across various Natural Language Processing (NLP) tasks, are devoted to the area of recommendation systems [42, 144, 36, 2]. In particular, LLMs facilitate recommendation learning by constructing personalized textual prompts to harness the common reasoning capabilities of LLMs. In this way, RecSys can naturally incorporate the recommendation task into natural language understanding, unlocking new opportunities for recommendation research.

However, one research gap remains to be solved. Most LLM-based recommendation methods still follow one-time recommendation-evaluation paradigm. This thesis focuses on the dynamic multi-round recommendation. The burst interactions between users and items are continuously forming and the system needs to dynamically update user preferences and refresh recommendation results over time. Obviously, the potential of LLMs in dynamic environment is still unclear, which motivates the future work of this thesis.

To fulfill the gap, some research questions naturally arise regarding *how to dynamically utilize LLMs to conduct multi-round recommendations for dynamic RecSys and how well the LLMs can perform when they continuously refresh recommendations over time*. The future objective of the research in this thesis is to incorporate LLMs into dynamic RecSys, conduct multi-round LLM-based recommendation, and verify their performance in dynamic environment, as shown in 7.1. On the one hand, the dynamic recommendation models (the proposed TGL models in this thesis) learn the temporal representations from user-item interaction space. On the other hand, the large language models perform semantic reasoning from textual prompts. Integrating the two aspects into a one framework aims to enhance the dynamic RecSys, shedding light on a promising direction for LLM-based recommendation research.

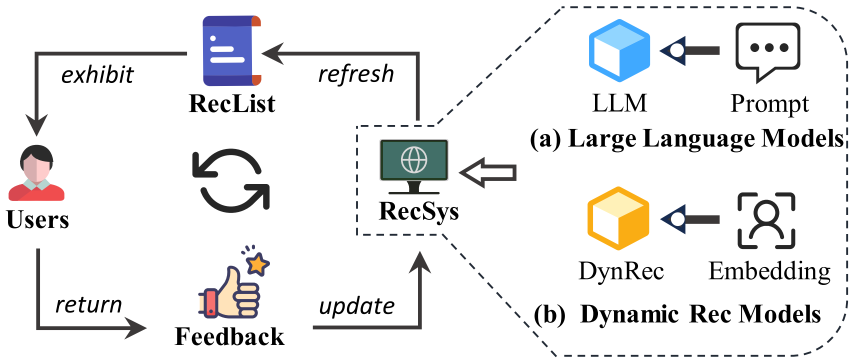


Figure 7.1: A simple illustration of the dynamic multi-round recommendation based on (a) large language models and (b) representation learning models.

Bibliography

[1] Q. Bai, C. Nie, H. Zhang, D. Zhao, and X. Yuan. "HGWaveNet: A Hyperbolic Graph Neural Network for Temporal Link Prediction." In: *Proceedings of the ACM Web Conference 2023*. WWW '23. ACM, 2023, pp. 523–532.

[2] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He. "TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation." In: *Proceedings of the 17th ACM Conference on Recommender Systems*. RecSys '23. ACM, 2023, pp. 1007–1014.

[3] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi. "Latent Cross: Making Use of Context in Recurrent Recommender Systems." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. ACM, 2018, pp. 46–54.

[4] S. Brody, U. Alon, and E. Yahav. "How Attentive are Graph Attention Networks?" In: *ICLR' 22*. 2022.

[5] G. Cai, J. Zhu, Q. Dai, Z. Dong, X. He, R. Tang, and R. Zhang. "ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. ACM, 2022, pp. 2692–2697.

[6] Z. Chai, Z. Chen, C. Li, R. Xiao, H. Li, J. Wu, J. Chen, and H. Tang. "User-Aware Multi-Interest Learning for Candidate Matching in Recommenders." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. ACM, 2022, pp. 1326–1335.

[7] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, and Y. Li. "Sequential Recommendation with Graph Neural Networks." In: *Proceedings of the 44th Inter-*

national ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '21. ACM, 2021, pp. 378–387.

[8] H. Chen, Y. Xiong, Y. Zhu, and P. S. Yu. "Highly Liquid Temporal Interaction Graph Embeddings." In: *Proceedings of the Web Conference 2021.* WWW '21. ACM, 2021, pp. 1639–1648.

[9] H. Chen, L. Wang, Y. Lin, C.-C. M. Yeh, F. Wang, and H. Yang. "Structured Graph Convolutional Networks with Stochastic Masks for Recommender Systems." In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '21. ACM, 2021, pp. 614–623.

[10] H. Chen, C.-C. M. Yeh, F. Wang, and H. Yang. "Graph Neural Transport Networks with Non-local Attentions for Recommender Systems." In: *Proceedings of the ACM Web Conference 2022.* WWW '22. ACM, 2022, pp. 1955–1964.

[11] K. Cheng, P. Linzhi, J. Ye, L. Sun, and B. Du. "Co-Neighbor Encoding Schema: A Light-cost Structure Encoding Method for Dynamic Link Prediction." In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* KDD '24. ACM, 2024, pp. 421–432.

[12] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. "An experimental comparison of click position-bias models." In: *Proceedings of the 2008 International Conference on Web Search and Data Mining.* WSDM '08. ACM, 2008, pp. 87–94.

[13] H. Dai, Y. Wang, R. Trivedi, and L. Song. "Deep coevolutionary network: Embedding user and item features for recommendation." In: *arXiv preprint arXiv:1609.03675* (2016).

[14] A. Dash, A. Chakraborty, S. Ghosh, A. Mukherjee, and K. P. Gummadi. "FaiRIR: Mitigating Exposure Bias From Related Item Recommendations in Two-Sided Platforms." In: *IEEE Transactions on Computational Social Systems* 10.3 (2023), pp. 1301–1313.

[15] Y. Deldjoo, V. W. Anelli, H. Zamani, A. Bellogin, and T. Di Noia. "A Flexible Framework for Evaluating User and Item Fairness in Recommender Systems." In: *User Modeling and User-Adapted Interaction* (2021), pp. 1–55.

[16] Y. Deng, Y. Li, B. Ding, and W. Lam. "Leveraging Long Short-Term User Preference in Conversational Recommendation via Multi-agent Reinforcement Learning." In:

IEEE Transactions on Knowledge and Data Engineering (TKDE) 35.11 (2023), pp. 11541–11555.

[17] V. Do, S. Corbett-Davies, J. Atif, and N. Usunier. "Online certification of preference-based fairness for personalized recommender systems." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 6. 2022, pp. 6532–6540.

[18] J. Du, Z. Ye, L. Yao, B. Guo, and Z. Yu. "Socially-aware Dual Contrastive Learning for Cold-Start Recommendation." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. ACM, 2022, pp. 1927–1932.

[19] Y. Du, X. Zhu, L. Chen, B. Zheng, and Y. Gao. "HAKG: Hierarchy-Aware Knowledge Gated Network for Recommendation." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. ACM, 2022, pp. 1390–1400.

[20] T. Ebesu, B. Shen, and Y. Fang. "Collaborative Memory Network for Recommendation Systems." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. ACM, 2018, pp. 515–524.

[21] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li. "Graph Trend Filtering Networks for Recommendation." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. ACM, 2022, pp. 112–121.

[22] Z. Fan, Z. Liu, J. Zhang, Y. Xiong, L. Zheng, and P. S. Yu. "Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer." In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. ACM, 2021, pp. 433–442.

[23] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, and Y. Li. "A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions." In: *ACM Trans. Recomm. Syst.* 1.1 (2023).

[24] Y. Ge, S. Liu, R. Gao, Y. Xian, Y. Li, X. Zhao, C. Pei, F. Sun, J. Ge, W. Ou, and Y. Zhang. "Towards Long-term Fairness in Recommendation." In: *Proceedings of the 14th ACM*

International Conference on Web Search and Data Mining. WSDM ’21. ACM, 2021, pp. 445–453.

[25] Y. Ge, J. Tan, Y. Zhu, Y. Xia, J. Luo, S. Liu, Z. Fu, S. Geng, Z. Li, and Y. Zhang. “Explainable Fairness in Recommendation.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’22. ACM, 2022, pp. 681–691.

[26] Y. Ge, X. Zhao, L. Yu, S. Paul, D. Hu, C.-C. Hsieh, and Y. Zhang. “Toward Pareto Efficient Fairness-Utility Trade-off in Recommendation through Reinforcement Learning.” In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. WSDM ’22. ACM, 2022, pp. 316–324.

[27] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. “DeepFM: a factorization-machine based neural network for CTR prediction.” In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. 2017, pp. 1725–1731.

[28] J. Guo, L. Du, X. Chen, X. Ma, Q. Fu, S. Han, D. Zhang, and Y. Zhang. “On Manipulating Signals of User-Item Graph: A Jacobi Polynomial-based Graph Collaborative Filtering.” In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’23. ACM, 2023, pp. 602–613.

[29] J. Guo, P. Zhang, C. Li, X. Xie, Y. Zhang, and S. Kim. “Evolutionary Preference Learning via Graph Nested GRU ODE for Session-based Recommendation.” In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM ’22. ACM, 2022, pp. 624–634.

[30] B. He, X. He, Y. Zhang, R. Tang, and C. Ma. “Dynamically Expandable Graph Convolution for Streaming Recommendation.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 1457–1467.

[31] W. He, G. Sun, J. Lu, and X. S. Fang. “Candidate-aware Graph Contrastive Learning for Recommendation.” In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’23. ACM, 2023, pp. 1670–1679.

[32] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. “LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation.” In: *Proceedings of the*

43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '20. ACM, 2020, pp. 639–648.

[33] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. "Neural Collaborative Filtering." In: *Proceedings of the 26th International Conference on World Wide Web.* WWW '17. 2017, pp. 173–182.

[34] M. Heuss, F. Sarvi, and M. de Rijke. "Fairness of Exposure in Light of Incomplete Exposure Estimation." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '22. ACM, 2022, pp. 759–769.

[35] C. Hsu and C.-T. Li. "RetaGNN: Relational Temporal Attentive Graph Neural Networks for Holistic Sequential Recommendation." In: *Proceedings of the Web Conference 2021.* WWW '21. ACM, 2021, pp. 2968–2979.

[36] J. Hu, W. Xia, X. Zhang, C. Fu, W. Wu, Z. Huan, A. Li, Z. Tang, and J. Zhou. "Enhancing Sequential Recommendation via LLM-based Semantic Embedding Learning." In: *Companion Proceedings of the ACM Web Conference 2024.* WWW '24. ACM, 2024, pp. 103–111.

[37] C. Huang, S. Wang, X. Wang, and L. Yao. "Modeling Temporal Positive and Negative Excitation for Sequential Recommendation." In: *Proceedings of the ACM Web Conference 2023.* WWW '23. ACM, 2023, pp. 1252–1263.

[38] J. Jiang, P. Zhang, Y. Luo, C. Li, J. B. Kim, K. Zhang, S. Wang, X. Xie, and S. Kim. "AdaMCT: Adaptive Mixture of CNN-Transformer for Sequential Recommendation." In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management.* CIKM '23. ACM, 2023, pp. 976–986.

[39] D. Kang, D. Han, N. Park, S. Kim, U. Kang, and S. Lee. "Eventera: Real-Time Event Recommendation System from Massive Heterogeneous Online Media." In: *2014 IEEE International Conference on Data Mining Workshop.* 2014, pp. 1211–1214.

[40] S. Kumar, X. Zhang, and J. Leskovec. "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD '19. ACM, 2019, pp. 1269–1278.

[41] S. Lai, E. Meng, F. Zhang, C. Li, B. Wang, and A. Sun. “An Attribute-Driven Mirror Graph Network for Session-based Recommendation.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’22. ACM, 2022, pp. 1674–1683.

[42] L. Li, Y. Zhang, and L. Chen. “Prompt Distillation for Efficient LLM-based Recommendation.” In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM ’23. ACM, 2023, pp. 1348–1357.

[43] M. Li, Z. Zhang, X. Zhao, W. Wang, M. Zhao, R. Wu, and R. Guo. “AutoMLP: Automated MLP for Sequential Recommendations.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 1190–1198.

[44] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang, and P. S. Yu. “Dynamic Graph Collaborative Filtering.” In: *2020 IEEE International Conference on Data Mining (ICDM)*. 2020, pp. 322–331.

[45] X. Li, Z. Wang, X. Chen, B. Guo, and Z. Yu. “A Hybrid Continuous-Time Dynamic Graph Representation Learning Model by Exploring Both Temporal and Repetitive Information.” In: *ACM Trans. Knowl. Discov. Data* 17.9 (2023).

[46] Y. Li, H. Chen, Z. Fu, Y. Ge, and Y. Zhang. “User-oriented Fairness in Recommendation.” In: *Proceedings of the Web Conference 2021*. WWW ’21. ACM, 2021, pp. 624–632.

[47] Y. Li, H. Chen, S. Xu, Y. Ge, and Y. Zhang. “Towards Personalized Fairness based on Causal Notion.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. ACM, 2021, pp. 1054–1063.

[48] Z. Li, W. Cheng, H. Xiao, W. Yu, H. Chen, and W. Wang. “You Are What and Where You Are: Graph Enhanced Attention Network for Explainable POI Recommendation.” In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM ’21. ACM, 2021, pp. 3945–3954.

[49] Y. Liang. “FAGR: Fairness-aware Group Recommendation in Event-based Social Networks.” In: *2022 IEEE International Conference on Big Data (Big Data)*. 2022, pp. 4926–4935.

[50] N. Lim, B. Hooi, S.-K. Ng, X. Wang, Y. L. Goh, R. Weng, and J. Varadarajan. “STP-UDGAT: Spatial-Temporal-Preference User Dimensional Graph Attention Network for Next POI Recommendation.” In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20. ACM, 2020, pp. 845–854.

[51] W. Lin, X. Zhao, Y. Wang, Y. Zhu, and W. Wang. “AutoDenoise: Automatic Data Instance Denoising for Recommendations.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 1003–1011.

[52] Y. Lin, C. Wang, Z. Chen, Z. Ren, X. Xin, Q. Yan, M. de Rijke, X. Cheng, and P. Ren. “A Self-Correcting Sequential Recommender.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 1283–1293.

[53] H. Liu. “LightSGCN: Powering Signed Graph Convolution Network for Link Sign Prediction with Simplified Architecture Design.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’22. ACM, 2022, pp. 2680–2685.

[54] J. Liu, D. Li, H. Gu, T. Lu, P. Zhang, L. Shang, and N. Gu. “Personalized Graph Signal Processing for Collaborative Filtering.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 1264–1272.

[55] L. Liu, Y. Guan, Z. Wang, R. Shen, G. Zheng, X. Fu, X. Yu, and J. Jiang. “An interactive food recommendation system using reinforcement learning.” In: *Expert Systems with Applications* 254 (2024), p. 124313.

[56] I. Loshchilov and F. Hutter. “Decoupled weight decay regularization.” In: *arXiv preprint arXiv:1711.05101* (2017).

[57] X. Luo, J. Yuan, Z. Huang, H. Jiang, Y. Qin, W. Ju, M. Zhang, and Y. Sun. “HOPE: High-order Graph ODE For Modeling Interacting Dynamics.” In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 23124–23139.

[58] M. Mansoury and B. Mobasher. “Fairness of exposure in dynamic recommendation.” In: *arXiv preprint arXiv:2309.02322* (2023).

[59] M. Morik, A. Singh, J. Hong, and T. Joachims. “Controlling Fairness and Bias in Dynamic Learning-to-Rank.” In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’20. ACM, 2020, pp. 429–438.

[60] M. Naghiaei, H. A. Rahmani, and Y. Deldjoo. “CPFair: Personalized Consumer and Producer Fairness Re-ranking for Recommender Systems.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’22. ACM, 2022, pp. 770–779.

[61] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim. “Continuous-Time Dynamic Network Embeddings.” In: *Companion Proceedings of the The Web Conference 2018.* WWW ’18. 2018, pp. 969–976.

[62] Z. Pan, F. Cai, W. Chen, and H. Chen. “Graph Co-Attentive Session-based Recommendation.” In: *ACM Trans. Inf. Syst.* 40.4 (2021).

[63] C. M. Parra, M. Gupta, and D. Dennehy. “Likelihood of Questioning AI-Based Recommendations Due to Perceived Racial/Gender Bias.” In: *IEEE Transactions on Technology and Society* 3.1 (2022), pp. 41–45.

[64] H. Qian, H. Zhou, Q. Zhao, H. Chen, H. Yao, J. Wang, Z. Liu, F. Yu, Z. Zhang, and J. Zhou. “Mdgnn: Multi-relational dynamic graph neural network for comprehensive and dynamic stock investment prediction.” In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 38. 13. 2024, pp. 14642–14650.

[65] X. Qin, H. Yuan, P. Zhao, G. Liu, F. Zhuang, and V. S. Sheng. “Intent Contrastive Learning with Cross Subsequences for Sequential Recommendation.” In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining.* WSDM ’24. ACM, 2024, pp. 548–556.

[66] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. “GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training.” In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD ’20. ACM, 2020, pp. 1150–1160.

[67] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. M. Bronstein. “Temporal Graph Networks for Deep Learning on Dynamic Graphs.” In: *CoRR* abs/2006.10637 (2020). arXiv: `2006.10637`.

[68] S. Sahebi, M. Yao, S. Zhao, and R. Feyzi Behnagh. “MoMENt: Marked Point Processes with Memory-Enhanced Neural Networks for User Activity Modeling.” In: *ACM Trans. Knowl. Discov. Data* 18.6 (2024).

[69] J. Sanz-Cruzado and P. Castells. “RELISON: A Framework for Link Recommendation in Social Networks.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’22. ACM, 2022, pp. 2992–3002.

[70] Q. Shen, W. Tao, J. Zhang, H. Wen, Z. Chen, and Q. Lu. “SAR-Net: A Scenario-Aware Ranking Network for Personalized Fair Recommendation in Hundreds of Travel Scenarios.” In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management.* CIKM ’21. ACM, 2021, pp. 4094–4103.

[71] T. Silva, N. Silva, H. Werneck, C. Mito, A. C. Pereira, and L. Rocha. “iRec: An Interactive Recommendation Framework.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’22. ACM, 2022, pp. 3165–3175.

[72] A. Singh and T. Joachims. “Fairness of Exposure in Rankings.” In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD ’18. ACM, 2018, pp. 2219–2228.

[73] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang. “Session-Based Social Recommendation via Dynamic Graph Attention Networks.” In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.* WSDM ’19. ACM, 2019, pp. 555–563.

[74] X. Song, J. Lian, H. Huang, M. Wu, H. Jin, and X. Xie. “Friend Recommendations with Self-Rescaling Graph Neural Networks.” In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* KDD ’22. ACM, 2022, pp. 3909–3919.

[75] Y. Su, R. Zhang, S. M. Erfani, and J. Gan. “Neural Graph Matching based Collaborative Filtering.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. ACM, 2021, pp. 849–858.

[76] S. Suresh, M. Shrivastava, A. Mukherjee, J. Neville, and P. Li. “Expressive and Efficient Representation Learning for Ranking Links in Temporal Graphs.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 567–577.

[77] H. Tang, S. Wu, X. Sun, J. Zeng, G. Xu, and Q. Li. “TCGC: Temporal Collaboration-Aware Graph Co-Evolution Learning for Dynamic Recommendation.” In: *ACM Transactions on Information Systems (TOIS)* 43.1 (2024), pp. 1–27.

[78] H. Tang, S. Wu, G. Xu, and Q. Li. “Dynamic Graph Evolution Learning for Recommendation.” In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’23. ACM, 2023, pp. 1589–1598.

[79] Y. Tian, Y. Yang, X. Ren, P. Wang, F. Wu, Q. Wang, and C. Li. “Joint Knowledge Pruning and Recurrent Graph Convolution for News Recommendation.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. ACM, 2021, pp. 51–60.

[80] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. “Graph Attention Networks.” In: *ICLR*. 2018.

[81] B. Wang, Y. Zhang, X. Wang, P. Wang, Z. Zhou, L. Bai, and Y. Wang. “Pattern Expansion and Consolidation on Evolving Graphs for Continual Traffic Prediction.” In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’23. ACM, 2023, pp. 2223–2232.

[82] C. Wang, Y. Ye, L. Ma, D. Li, and L. Zhuang. “Dual disentanglement of user–item interaction for recommendation with causal embedding.” In: *Information Processing and Management* 60.5 (2023), p. 103456.

[83] G. Wang, R. Ying, J. Huang, and J. Leskovec. “Improving graph attention networks with large margin-based constraints.” In: *arXiv preprint arXiv:1910.11945* (2019).

[84] J. Wang, G. Song, Y. Wu, and L. Wang. "Streaming Graph Neural Networks via Continual Learning." In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* CIKM '20. ACM, 2020, pp. 1515–1524.

[85] J. Wang, W. Zhu, G. Song, and L. Wang. "Streaming Graph Neural Networks with Generative Replay." In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* KDD '22. ACM, 2022, pp. 1878–1888.

[86] L. Wang, E.-P. Lim, Z. Liu, and T. Zhao. "Explanation Guided Contrastive Learning for Sequential Recommendation." In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* CIKM '22. ACM, 2022, pp. 2017–2027.

[87] L. Wang and T. Joachims. "User Fairness, Item Fairness, and Diversity for Rankings in Two-Sided Markets." In: *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval.* ICTIR '21. ACM, 2021, pp. 23–41.

[88] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen. "Streaming Ranking Based Recommender Systems." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* SIGIR '18. ACM, 2018, pp. 525–534.

[89] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua. "Neural Graph Collaborative Filtering." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR'19. ACM, 2019, pp. 165–174.

[90] Y. Wang, H. Guo, R. Tang, Z. Liu, and X. He. "A Practical Incremental Method to Train Deep CTR Models." In: *CoRR* abs/2009.02147 (2020). arXiv: 2009.02147.

[91] Y. Wang, W. Ma, M. Zhang, Y. Liu, and S. Ma. "A Survey on The Fairness of Recommender Systems." In: *ACM Transactions on Information Systems* 41.3 (2023), pp. 1–43.

[92] Y. Wang, Y. Zhao, Y. Zhang, and T. Derr. "Collaboration-Aware Graph Convolutional Network for Recommender Systems." In: *Proceedings of the ACM Web Conference 2023.* WWW '23. ACM, 2023, pp. 91–101.

[93] Y. Wang, Y. Zhao, Y. Zhang, and T. Derr. “Collaboration-Aware Graph Convolutional Network for Recommender Systems.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 91–101.

[94] Y. Wang and C. Mendis. “TGLite: A Lightweight Programming Framework for Continuous-Time Temporal Graph Neural Networks.” In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS ’24. ACM, 2024, pp. 1183–1199.

[95] Z. Wang, P. Yang, X. Fan, X. Yan, Z. Wu, S. Pan, L. Chen, Y. Zang, C. Wang, and R. Yu. “ConTIG: Continuous representation learning on temporal interaction graphs.” In: *Neural Networks* 172 (2024), p. 106151.

[96] Z. Wen and Y. Fang. “TREND: TempoRal Event and Node Dynamics for Graph Representation Learning.” In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. ACM, 2022, pp. 1159–1169.

[97] C. Wu, F. Wu, X. Wang, Y. Huang, and X. Xie. “Fairness-aware news recommendation with decomposed adversarial learning.” In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 5. 2021, pp. 4462–4469.

[98] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie. “Self-supervised Graph Learning for Recommendation.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. ACM, 2021, pp. 726–735.

[99] Y. Wu, J. Cao, G. Xu, and Y. Tan. “TFROM: A Two-sided Fairness-Aware Recommendation Model for Both Customers and Providers.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. ACM, 2021, pp. 1013–1022.

[100] Y. Wu, Y. Fang, and L. Liao. “On the Feasibility of Simple Transformer for Dynamic Graph Modeling.” In: *Proceedings of the ACM Web Conference 2024*. WWW ’24. ACM, 2024, pp. 870–880.

[101] J. Xia, D. Li, H. Gu, J. Liu, T. Lu, and N. Gu. “FIRE: Fast Incremental Recommendation with Graph Signal Processing.” In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. ACM, 2022, pp. 2360–2369.

[102] J. Xia, D. Li, H. Gu, T. Lu, P. Zhang, and N. Gu. “Incremental Graph Convolutional Network for Collaborative Filtering.” In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM ’21. ACM, 2021, pp. 2170–2179.

[103] J. Xia, D. Li, H. Gu, T. Lu, P. Zhang, L. Shang, and N. Gu. “Neural Kalman Filtering for Robust Temporal Recommendation.” In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. WSDM ’24. ACM, 2024, pp. 836–845.

[104] L. Xia, C. Huang, Y. Xu, P. Dai, X. Zhang, H. Yang, J. Pei, and L. Bo. “Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation.” In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 5. 2021, pp. 4486–4493.

[105] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang. “Hypergraph Contrastive Collaborative Filtering.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’22. ACM, 2022, pp. 70–79.

[106] L. Xia, C. Huang, and C. Zhang. “Self-Supervised Hypergraph Transformer for Recommender Systems.” In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’22. ACM, 2022, pp. 2100–2109.

[107] C. Xiao, W. Ji, Y. Zhang, and S. Lv. “PIDKG: Propagating Interaction Influence on the Dynamic Knowledge Graph for Recommendation.” In: *ACM Trans. Web* 18.2 (2024).

[108] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, and B. Cui. “Contrastive Learning for Sequential Recommendation.” In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2022, pp. 1259–1273.

[109] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. “Inductive representation learning on temporal graphs.” In: *ICLR*. 2020.

[110] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. “How powerful are graph neural networks?” In: *arXiv preprint arXiv:1810.00826* (2018).

[111] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen. “Deep matrix factorization models for recommender systems.” In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. 2017, pp. 3203–3209.

[112] H. Yadav, Z. Du, and T. Joachims. “Policy-Gradient Training of Fair and Unbiased Ranking Functions.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’21. ACM, 2021, pp. 1044–1053. ISBN: 9781450380379.

[113] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. “Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs.” In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.1 (2014), pp. 129–142.

[114] M. Yang, M. Zhou, M. Kalander, Z. Huang, and I. King. “Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space.” In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* KDD ’21. ACM, 2021, pp. 1975–1985.

[115] T. Yang and Q. Ai. “Maximizing Marginal Fairness for Dynamic Learning to Rank.” In: *Proceedings of the Web Conference 2021.* WWW ’21. ACM, 2021, pp. 137–145.

[116] Y. Yang, J. Cao, M. Stojmenovic, S. Wang, Y. Cheng, C. Lum, and Z. Li. “Time-Capturing Dynamic Graph Embedding for Temporal Linkage Evolution.” In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2023), pp. 958–971.

[117] Y. Yang, C. Huang, L. Xia, and C. Li. “Knowledge Graph Contrastive Learning for Recommendation.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’22. ACM, 2022, pp. 1434–1443.

[118] Z. Yang, M. Ding, B. Xu, H. Yang, and J. Tang. “STAM: A Spatiotemporal Aggregation Method for Graph Neural Network-based Recommendation.” In: *Proceedings of the ACM Web Conference 2022.* WWW ’22. ACM, 2022, pp. 3217–3228.

[119] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong. “Learning the Evolutionary and Multi-scale Graph Structure for Multivariate Time Series Forecasting.” In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* KDD ’22. ACM, 2022, pp. 2296–2306.

[120] Y. Yin, Y. Wu, X. Yang, W. Zhang, and X. Yuan. “SE-GRU: Structure Embedded Gated Recurrent Unit Neural Networks for Temporal Link Prediction.” In: *IEEE Transactions on Network Science and Engineering* 9.4 (2022), pp. 2495–2509.

[121] H. Yoo, Z. Zeng, J. Kang, R. Qiu, D. Zhou, Z. Liu, F. Wang, C. Xu, E. Chan, and H. Tong. "Ensuring User-side Fairness in Dynamic Recommender Systems." In: *Proceedings of the ACM Web Conference 2024*. WWW '24. ACM, 2024, pp. 3667–3678.

[122] S. Yu, L. Ma, X. Gao, J. Guo, and G. Chen. "Attentive Hawkes Process Application for Sequential Recommendation." In: *International Conference on Database Systems for Advanced Applications*. Springer. 2023, pp. 473–488.

[123] H. Yuan, P. Zhao, X. Xian, G. Liu, Y. Liu, V. S. Sheng, and L. Zhao. "Sequential Recommendation with Probabilistic Logical Reasoning." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*. 2023, pp. 2432–2440.

[124] K. Yuan, G. Liu, J. Wu, and H. Xiong. "Semantic and Structural View Fusion Modeling for Social Recommendation." In: *IEEE Transactions on Knowledge and Data Engineering* 35.11 (2023), pp. 11872–11884.

[125] J. Zeng, H. Tang, Y. Zhao, and J. Wen. "Neu-PCM: Neural-based potential correlation mining for POI recommendation." In: *Applied Intelligence* 53.9 (2023), pp. 10685–10698.

[126] D. Zhang, Y. Zhu, Y. Dong, Y. Wang, W. Feng, E. Kharlamov, and J. Tang. "ApeGNN: Node-Wise Adaptive Aggregation in GNNs for Recommendation." In: *Proceedings of the ACM Web Conference 2023*. WWW '23. ACM, 2023, pp. 759–769.

[127] G. Zhang, T. Ye, D. Jin, and Y. Li. "An Attentional Multi-scale Co-evolving Model for Dynamic Link Prediction." In: *Proceedings of the ACM Web Conference 2023*. WWW '23. ACM, 2023, pp. 429–437.

[128] H. Zhang, Y. Sun, W. Guo, Y. Liu, H. Lu, X. Lin, and H. Xiong. "Interactive Interior Design Recommendation via Coarse-to-fine Multimodal Reinforcement Learning." In: *Proceedings of the 31st ACM International Conference on Multimedia*. MM '23. 2023, pp. 6472–6480.

[129] H. Zhang, Y. Li, B. Ding, and J. Gao. "LOKI: A Practical Data Poisoning Attack Framework Against Next Item Recommendations." In: *IEEE Transactions on Knowledge and Data Engineering* 35.5 (2023), pp. 5047–5059.

[130] J. Zhang, J. Shao, and B. Cui. “StreamE: Learning to Update Representations for Temporal Knowledge Graphs in Streaming Scenarios.” In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’23. ACM, 2023, pp. 622–631.

[131] K. Zhang, Y. Wang, X. Li, R. Tang, and R. Zhang. “IncMSR: An Incremental Learning Approach for Multi-Scenario Recommendation.” In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. WSDM ’24. ACM, 2024, pp. 939–948.

[132] M. Zhang, S. Wu, X. Yu, Q. Liu, and L. Wang. “Dynamic Graph Neural Networks for Sequential Recommendation.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 35.5 (2023), pp. 4741–4753.

[133] M. Zhang, Y. Xia, Q. Liu, S. Wu, and L. Wang. “Learning Long- and Short-term Representations for Temporal Knowledge Graph Reasoning.” In: *Proceedings of the ACM Web Conference 2023*. WWW ’23. ACM, 2023, pp. 2412–2422.

[134] S. Zhang, L. Chen, C. Wang, S. Li, and H. Xiong. “Temporal Graph Contrastive Learning for Sequential Recommendation.” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.8 (2024), pp. 9359–9367.

[135] S. Zhang, X. Chen, Y. Xiong, X. Wu, Y. Zhang, Y. Fu, Y. Zhao, and J. Zhang. “Towards Adaptive Neighborhood for Advancing Temporal Interaction Graph Modeling.” In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’24. ACM, 2024, pp. 4290–4301.

[136] S. Zhang, N. Zheng, and D. Wang. “HetGRec: Heterogeneous Graph Attention Network for Group Recommendation.” In: *IEEE Intelligent Systems* 38.1 (2023), pp. 9–18.

[137] W. Zhang, Z. Chen, H. Zha, and J. Wang. “Learning from Substitutable and Complementary Relations for Graph-based Sequential Product Recommendation.” In: *ACM Trans. Inf. Syst.* 40.2 (2021).

[138] X. Zhang, H. Lin, B. Xu, C. Li, Y. Lin, H. Liu, and F. Ma. “Dynamic intent-aware iterative denoising network for session-based recommendation.” In: *Information Processing and Management* 59.3 (2022), p. 102936.

[139] Y. Zhang, F. Feng, C. Wang, X. He, M. Wang, Y. Li, and Y. Zhang. “How to Retrain Recommender System? A Sequential Meta-Learning Method.” In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’20. ACM, 2020, pp. 1479–1488.

[140] Y. Zhang, Y. Xiong, D. Li, C. Shan, K. Ren, and Y. Zhu. “CoPE: Modeling Continuous Propagation and Evolution on Interaction Graph.” In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management.* CIKM ’21. ACM, 2021, pp. 2627–2636.

[141] Y. Zhang, Y. Xiong, Y. Liao, Y. Sun, Y. Jin, X. Zheng, and Y. Zhu. “TIGER: Temporal Interaction Graph Embedding with Restarts.” In: *Proceedings of the ACM Web Conference 2023.* WWW ’23. ACM, 2023, pp. 478–488.

[142] M. Zhao, L. Wu, Y. Liang, L. Chen, J. Zhang, Q. Deng, K. Wang, X. Shen, T. Lv, and R. Wu. “Investigating Accuracy-Novelty Performance for Graph-based Collaborative Filtering.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’22. ACM, 2022, pp. 50–59.

[143] Q. Zhao. “RESETBERT4Rec: A Pre-training Model Integrating Time And User Historical Behavior for Sequential Recommendation.” In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’22. ACM, 2022, pp. 1812–1816.

[144] Y. Zhao, J. Wu, X. Wang, W. Tang, D. Wang, and M. de Rijke. “Let Me Do It For You: Towards LLM Empowered Recommendation via Tool Learning.” In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’24. ACM, 2024, pp. 1796–1806.

[145] Z. Zhao, J. Chen, S. Zhou, X. He, X. Cao, F. Zhang, and W. Wu. “Popularity Bias is not Always Evil: Disentangling Benign and Harmful Bias for Recommendation.” In: *IEEE Transactions on Knowledge and Data Engineering* 35.10 (2023), pp. 9920–9931.

[146] Z. Zhao, X. Zhu, T. Xu, A. Lizhiyu, Y. Yu, X. Li, Z. Yin, and E. Chen. “Time-interval Aware Share Recommendation via Bi-directional Continuous Time Dynamic Graphs.” In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’23. ACM, 2023, pp. 822–831.

[147] T. Zheng, X. Wang, Z. Feng, J. Song, Y. Hao, M. Song, X. Wang, X. Wang, and C. Chen. “Temporal Aggregation and Propagation Graph Neural Networks for Dynamic Representation.” In: *IEEE Transactions on Knowledge and Data Engineering* 35.10 (2023), pp. 10151–10165.

[148] X. Zhu, Y. Wu, L. Wang, H. Su, and Z. Li. “Continuous-Time Dynamic Interaction Network Learning Based on Evolutionary Expectation.” In: *IEEE Transactions on Cognitive and Developmental Systems* 16.3 (2024), pp. 840–849.

[149] Y. Zhu, F. Cong, D. Zhang, W. Gong, Q. Lin, W. Feng, Y. Dong, and J. Tang. “WinGNN: Dynamic Graph Neural Networks with Random Gradient Aggregation Window.” In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’23. ACM, 2023, pp. 3650–3662.

[150] Y. Zhu, J. Chen, L. Chen, Y. Li, F. Zhang, and Z. Liu. “Interest Clock: Time Perception in Real-Time Streaming Recommendation System.” In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’24. ACM, 2024, pp. 2915–2919.

[151] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai. “What to do next: modeling user behaviors by time-LSTM.” In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. 2017, pp. 3602–3608.