# Towards Graph-based Explainable Recommender Systems

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

Doctor of Philosophy

*by*

**Yicong Li**

*under the supervision of*

Guandong Xu
Angela Huo

*to*

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

Jan 2024

# CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Yicong Li 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.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:

SIGNATURE: Signature removed prior to publication.

DATE: 19th Jan, 2024

PLACE: Sydney, Australia

# Abstract

Explainable recommender systems, which aim to provide accurate recommendations and reliable explanations, have attracted significant research interest due to their ability to enhance the recommender system's transparency, build user trust, and bring additional benefits to the system. In this thesis, I study a critical yet not well-explored question of explainable recommendation in graph structure. Graph-based explainable recommendations aim to predict users' preferences in the recommendation graph for accurate recommendations and reliable explanations by learning users' historical behaviors. The research delves into the cutting-edge work in graph-based explainable recommendations and reveals that the current explainable recommendations overlook the issue of imbalanced data structures, and hampers the effective modeling of users' historical behaviors on the massive information, leading to less precise recommendation outcomes and explanations. Furthermore, most existing work fails to provide reliable explanations for their recommendation results due to the controversy of attention mechanisms. I have studied the above research problem in deep depth and addressed the following three technical challenges: 1) The imbalance of graph data distribution leads to imbalanced learning results and poor generalization performance to sparse but majority structures. 2) Recommendation graphs, constructed from heterogeneous, massive, and temporal data, pose a fundamental challenge in capturing users' historical behaviors for explanations due to their complex structure. 3) The black-box nature of graph neural networks makes it a difficult challenge to understand, reason the graph structure, and further generate meaningful explanations. Specifically, I proposed three research works to achieve satisfactory recommendation results and explainability. For the first challenge, I first study the imbalanced graph data distribution problem in a dynamic graph scenario and propose a novel fair dynamic graph embedding to close the gap between the active and inactive items to achieve fair recommendations. For the second challenge, I propose a novel reinforcement learning framework for explainable paths exploration, and then model the user's historical behavior for accurate recommendation via the explored meaningful paths. For the third challenge, I propose a post-hoc explainable module leveraging counterfactual learning to generate reliable explanations. Qualitative and quantitative experiments are conducted to validate the state-of-the-art and effective recommendation performance and explainability. I am confident that this research will significantly advance graph-based explainable recommender systems, setting the stage for the creation of more reliable explainable systems in the future.

# Acknowledgments

During my Ph.D. study, my peers and I have suffered from a severe COVID-19 period and cannot act as usual to study and live in Sydney. I also have fallen into deep confusion and psychological shadow for a long time at home because of experiences I never imagined. However, I have been knowing that I have to struggle away from them and find a way to get through them. Ultimately, time and geography prevailed over the sorrowful period.

First, I really appreciate my supervisor, Prof. Guandong Xu. He is very nice and has been my support during my Ph.D. period all the time. His congrats when I accepted my paper, his encouragement when I had less progress, and his support when I encountered difficulties have always been of great help to me. I would not finish my Ph.D. without his guidance and help. He will be my role model for life.

I also thank Dr. Hongxu Chen, Dr. Xiangguo Sun, Dr. Yu Yang and Dr. Angela Huo for their guidance. During my Ph.D. period, Dr. Hongxu Chen was the first co-supervisor to guide me and really helped me a lot in my research. Without his careful guidance, I would not have been able to start my research work so quickly. I met Dr. Xiangguo Sun and Dr. Yu Yang in the latter stages, and they also helped me in my research. I give my thanks to Dr. Yu Yang again for his kind guidance of my thesis and his excellent character. Thanks Dr. Angela Huo for her kind help. I also want to give my thanks to Prof. Lin Li for providing me with a quiet research environment in her research group in Wuhan.

Moreover, I give my thanks to my parents for their generous financial support and emotional support. I am so lucky that I was born into such a family with unlimited happiness. I also thank my careful little brother for his cooked meal each time at home. Without them, I cannot overcome various difficulties and live as I am now.

Last but not least, I want to give my thanks to my colleagues at DSMI, including Dr. Li He, Haoran Yang, Xiangmeng Wang, Sirui Huang, Haoran Tang, Kangzheng Liu, and my previous colleague, Sixiao Zhang at Nanyang Technological University. I especially express my appreciation to Dr. Shuaiqi Liu for his companionship and thank god I am so lucky to meet him.

# LIST OF PUBLICATIONS

## CONFERENCE PAPERS :

1. Hongxu Chen⋆, **Yicong Li**⋆, Xiangguo Sun, Guandong Xu, and Hongzhi Yin. "Temporal meta-path guided explainable recommendation." In Proceedings of the 14th ACM international conference on web search and data mining, pp. 1056-1064. 2021. (⋆ means equal contribution.)

2. **Yicong, Li**, Hongxu Chen, Xiangguo Sun, Zhenchao Sun, Lin Li, Lizhen Cui, Philip S. Yu, and Guandong Xu. "Hyperbolic hypergraphs for sequential recommendation." In Proceedings of the 30th ACM international conference on information & knowledge management, pp. 988-997. 2021.

3. **Yicong Li**, Yu Yang, Jiannong Cao, Shuaiqi Liu, Haoran Tang, Guandong Xu. "Toward Structure Fairness in Dynamic Graph Embedding: A Trend-aware Dual Debiasing Approach." KDD24. (Under review)

## JOURNAL PAPERS :

1. **Yicong Li**, Hongxu Chen, Yile Li, Lin Li, S. Yu Philip, and Guandong Xu. "Reinforcement Learning based Path Exploration for Sequential Explainable Recommendation." IEEE Transactions on Knowledge and Data Engineering (2023).

2. **Yicong Li**, Xiangguo Sun, Hongxu Chen, Sixiao Zhang, Yu Yang, Guandong Xu. "Attention Is Not the Only Choice: Counterfactual Reasoning for Path-Based Explainable Recommendation." IEEE Transactions on Knowledge and Data Engineering (2024).

3. **Yicong Li**, Sirui Huang, Yu Yang, Haoran Yang, Qing Li, Jiannong Cao, Guandong Xu. "Dynamic-aware Hypergraphs Neural Network for Explainable Sequential Recommendation." (Submitted to IEEE Transactions on Neural Networks and Learning Systems)

**OTHER CO-AUTHER PAPERS :**

1. Haoran Yang, Xiangyu Zhao, **Yicong Li**, Hongxu Chen, and Guandong Xu. November. "An Empirical Study Towards Prompt-Tuning for Graph Contrastive Pre-Training in Recommendations." In Thirty-seventh Conference on Neural Information Processing Systems. 2023.

2. Kangzheng Liu, Feng Zhao, Hongxu Chen, **Yicong Li**, Guandong Xu, and Hai Jin. "Da-net: Distributed attention network for temporal knowledge graph reasoning." In Proceedings of the 31st ACM International Conference on Information Knowledge Management, pp. 1289-1298. 2022.

3. Sixiao Zhang, Hongxu Chen, Xiangguo Sun, **Yicong Li**, and Guandong Xu. "Unsupervised graph poisoning attack via contrastive loss back-propagation." In Proceedings of the ACM Web Conference 2022, pp. 1322-1330. 2022.

4. Hongxu Chen, **Yicong Li**, and Haoran Yang. "Graph Data Mining in Recommender Systems." Web Information Systems Engineering-WISE 2021: 22nd International Conference on Web Information Systems Engineering, WISE 2021, Melbourne, VIC, Australia, October 26-29, 2021, Proceedings, Part II 22. Springer International Publishing, 2021.

# TABLE OF CONTENTS

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, I give a brief introduction of the thesis, including the development of recommendations, the background of graph-based explainable recommendations, challenges, contributions, and the thesis organization.

## 1.1 Background

With the development of e-commerce and redundant online product information, it is increasingly difficult for users to choose preferred items or places, like interesting products [25, 83, 115], favorite music [1, 47, 107], desired videos [38, 78, 85], locations to go [34, 185, 187] and so on. To solve this problem, the recommendation system, as a significant tool, has a strong ability to help users choose suitable items or places among the mess of information online. Figure 1.1 illustrates the recommendation process for the user, which not only involves users and items but also sometimes corporates user profiles, item attributes, and other external knowledge. The recommender system learns the latent relations among the above factors for users' preference modeling and then provides the prediction to users.

Early recommender systems rely on content for modeling [10, 73, 80, 103, 104, 135], which depends on the item's content to learn users' preferences. This kind of recommender system tends to predict tightly related users' past tastes and causes boring predictions due to too many similar tastes. Collaborative filtering-based recommender systems [71, 98, 113, 117, 118, 134] alleviate this problem by recommending the user with a

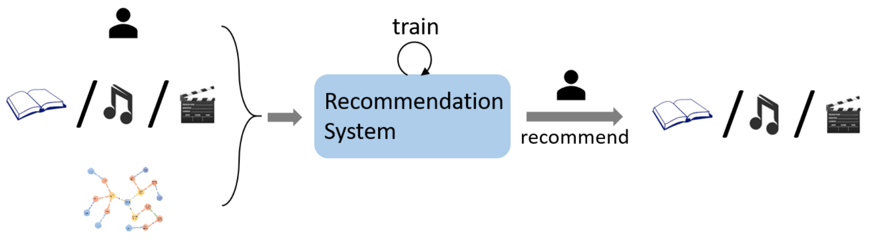


Figure 1.1: A general process of recommendation.

co-purchased user's interacted items. Specifically, they adopt the hypothesis that users who have bought the same item share the same interests and tend to buy similar items in the future. This method learns the underlying co-purchase relations among users who have purchased the same items for recommendations. With the development of neural networks, neural networks have been widely applied to recommendation systems [35, 95, 177] to model complex user-item interactions for user preferences learning. These methods have achieved remarkable performance in various recommendation tasks.

In recent years, thanks to the success of graph neural networks (GNNs), recommendation systems have a new paradigm, graph-based recommendation [31, 52, 66]. It leverages the power of graph neural networks to model complex and diverse user-item interactions and preferences. GNNs capture the high-order connectivity, the structural property, and the enhanced supervision signal of graph data, which are difficult to reveal by the above traditional recommender systems.

In the contemporary digital landscape, recommender systems are no longer sufficient to merely provide suitable recommendations to users. This is due to the inherent uncertainty regarding the perceived reliability and credibility of the recommended items from the users' perspective. Consequently, the current research focus extends beyond enhancing the effectiveness and performance of these recommender systems. It also encompasses efforts to augment the transparency and trustworthiness of the recommendations, which necessitates an exploration into the explainability of these recommender systems. Therefore, it is necessary to present users with recommended items and provide comprehensive and comprehensible explanations for these recommendations.

Moreover, explainability in recommender systems also enhances the transparency, persuasiveness, effectiveness, trustworthiness, and user satisfaction of recommender systems [179]. Furthermore, these explanations provide a valuable tool for system designers, facilitating the diagnosis, debugging, and refinement of the recommendation algorithm. From the end-user's perspective, providing explanations will significantly bolster the perceived credibility of the recommendations. Taking a music recommender system as an example, it learns a user's preference from the historical favorite music list and subsequently recommends a new song. The user, unfamiliar with the new song, initially tends to choose not to listen to it. However, if the recommender system provides a rationale for the recommendation - such as the singer being the same as one of the user's favorite singers - the user is more likely to engage with the recommended song.

Early work on the recommendations' explainability was studied before 2000, but the concept was introduced in 2014 [182]. In these early studies, traditional collaborative-filtering explainable recommendations exhibited commendable explainability due to the collaborative relationships learned within the recommendation model. However, the advent of deep learning, while enhancing recommendation performance, also introduced the 'black-box' feature to the recommendation model, thereby complicating its explainability. In recent years, the successful application of GNNs has allowed recommender systems to generate more accurate recommendation outputs compared to traditional methods. However, the use of graphs to structure the recommendation data has further abstracted the model, making it challenging to extract explanations.

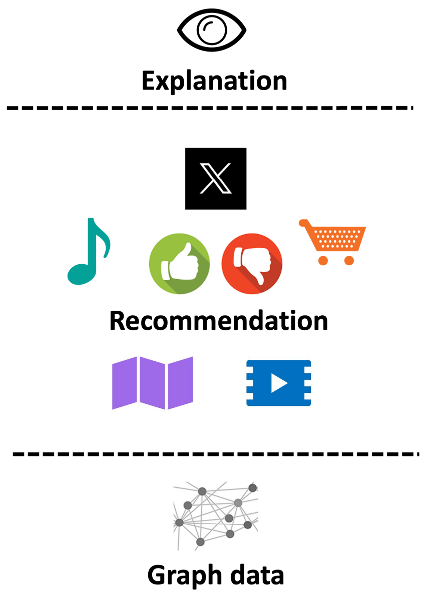


Figure 1.2: Graph-based explainable recommendation framework.

In this thesis, the research topic is graph-based explainable recommender systems. It aims to model users' historical behaviors for accurate recommendations and provide reliable explanations on the graph constructed by the recommendation data. I summarize the three aspects of research problems of graph-based explainable recommender systems in Figure 1.2, including graph data, the recommendation and the explanation. The detailed research problems are analyzed as follows.

**Graph data**: Existing work [88, 169, 170, 186] points out the imbalanced graph structure issue in the recommendation, which means the majority of users/items have little interactions, but the minority have sufficient interactions for training. This leads to imbalanced graph training for both recommendations and explanations. Especially the recommendation data is temporal, which makes the research problem more complex.

**Recommendation modeling**: A fundamental task of a graph-based explainable recommender system is to model each user's historical behaviors for preference prediction. [14, 63, 79, 124] Therefore, how to learn users' historical behavior on the recommendation graph also becomes a primary research problem.

**Explanation generation**: Current approaches often fail to provide valid explanations for their recommendation results [40, 61, 145, 147]. The black-box nature of GNNs makes it difficult to generate meaningful explanations, necessitating sophisticated algorithms capable of understanding and reasoning about the graph structures.

## 1.2 Challenges

To address the above issues, there are the following three challenges to tackle, including imbalanced data structure, users' sequential historical modeling, and explainability.

Firstly, according to the **graph data** observation, recommendation data is often imbalanced, with a majority of users lacking sufficient user-item interaction records, as depicted in Figure 1.3. The x-axis denotes the node (user/item) id, and the y-axis means the degree of the node (the number of interactions), which conforms to the long-tail distribution. This imbalance will result in users with fewer interactions disproportionately influencing the performance and explainability quality of the recommendation model, leading to disparate treatment of different user groups. For active users, the explainable recommendation model will emphasize historical data more, thereby achieving superior performance and explainability. Conversely, for inactive users, the training is inadequate, resulting in subpar performance and explainability. This disparity engenders a degree of unfairness in explainable recommendations. While some studies have explored fair graph modeling [27, 36, 39, 74, 123], they often overlook the issue of unfairness in dynamic scenarios. Given the evolving nature of users' preferences, modeling in dynamic scenarios is an inherent challenge in recommendation systems. Consequently, addressing the issue of unfairness within dynamic recommendation scenarios presents a significant challenge.

Furthermore, from the **recommendation modeling** aspect, current methodologies

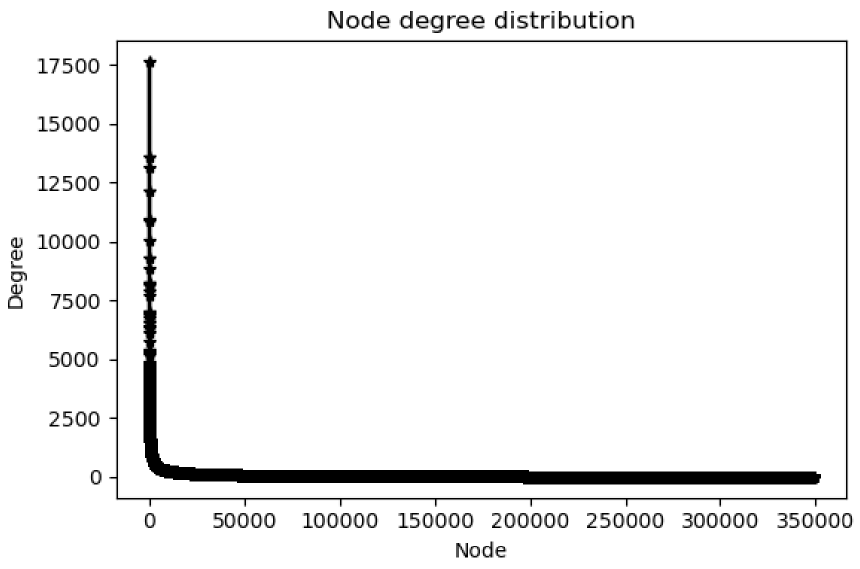


Figure 1.3: The distribution of users' behavior records in the Amazon Book dataset.

struggle to comprehend users' historical behaviors, resulting in less accurate recommendation outcomes and explanations. Most graph-based explainable recommendations depend on related paths within the graph to provide explanations for recommendations [2, 61, 143, 149, 159, 160]. However, the volume of paths is considerably large. These methods typically rely on random path selection to guide the generation of explainable paths [18, 44, 59, 120, 122, 144], and they lack sequential modeling for the evolution of users' historical behavior. It not only introduces randomness but also overlooks the sequential evolution of users' preferences, which compromises the quality of explainable paths and the performance of recommendations. In a large, heterogeneous recommendation graph, the sequential exploration of meaningful and explainable paths to enhance recommendation effectiveness poses a significant challenge.

For the **explanation generation**, the controversy of attention mechanism-based explainability is another issue. As mentioned before, most of the existing graph-based recommendations use paths for the explanation. Unfortunately, in the heterogeneous recommendation graph, there are usually a lot of underlying paths to impact the final decision, which brings a huge challenge to select more explainable paths from the large candidate space. Recently, some works [18, 90, 147] have integrated the attention mechanisms into their models and learned the path weights for further explanation. These weights are usually reflected in an item's one-hop neighbors [147], users' purchase records [18], and some external knowledge [90]. However, the attention mechanism is widely questioned for its interpretability because many studies have found the weak reliability that the attention mechanism performs [119, 154]. As shown in Figure 1.4, I run an attention-based model on 16 underlying paths three times and draw the path-level attention weights in the heat map where each block denotes a specific path and the darker orange color means higher attention weight. It is obvious that the attention-based model does not ensure stable weight distributions via different independent running, which is far-fetched to convince customers to accept the recommended results. Moreover, there are currently few widely-used quantitative evaluations due to the difficult definition of explainability from a mathematical view, which also impedes the development of explainability.

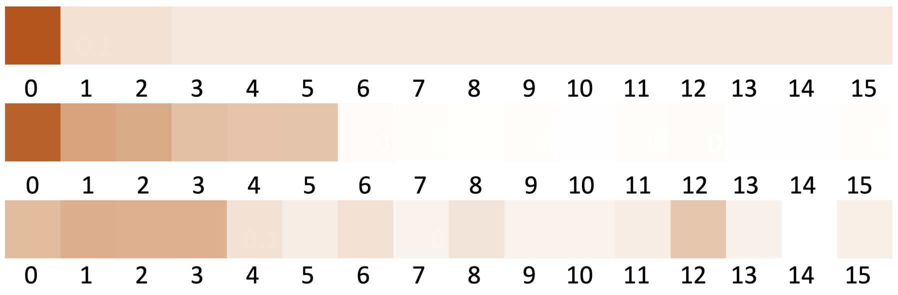


Figure 1.4: The attention weights on 16 paths via three times independent training. Each block presents a path, and darker orange means higher weight.

## 1.3 Contributions

The main contributions of the thesis are listed as follows.

- For **graph data**, I study the imbalanced graph structure issue and identify the unfairness problem in the dynamic graph, which will have a negative impact on recommendation and explanation performance. To address this issue, I propose FairDGE, a novel fair dynamic graph embedding algorithm that learns biased structural evolutions and then uses a newly devised dual debiasing method for encoding structurally fair embeddings that are highly effective in recommendation performance. Experimental results demonstrate that FairDGE achieves simultaneous improvement in effectiveness and fairness embedded in recommendations. This fair graph learning is fundamental to both recommendation modeling and explanation generation.

- For **recommendation**, I study the user's historical behavior sequential modeling, which captures the user's preference evolution for both accurate recommendation and explanation. Moreover, I also study to explore meaningful explainable paths leveraging reinforcement learning for precise user preference modeling. Specifically, I propose a novel reinforcement path exploration for diversity path mining, Temporal Meta-path Guided Explainable Recommendation leveraging Reinforcement Learning (TMER-RL). Compared to traditional path exploration, the method will explore more diverse explainable paths and achieve a better user's historical behavior modeling for recommendation. Experiments on four real-world sequential recommendation datasets demonstrate that TMER-RL outperforms several state-of-the-art baselines in recommendation performance and generating path explanations. Especially, I discuss the reinforcement learning-based paths exploration and random selection approaches through the case study. This user's historical behavior sequential modeling benefits both recommendation and explanation learning.

- For **explanation**, I point out the universal controversy of the attention mechanism to model explainability and find the superiority of counterfactual learning for explainability learning. Specifically, I propose a novel model-agnostic explainable framework for path-based recommendation via counterfactual reasoning, namely Counterfactual Path-based Explainable Recommendation, CPER. It conducts the counterfactual reasoning on both path representation and path topological structure. Moreover, I propose a package of solutions to evaluate the explainability quality of path-based recommendations. Unlike traditional attention-based explanations, mostly evaluated by case studies, the proposed measurements include qualitative and quantitative methods, which can be widely used in comparing various explanation methods. This counterfactual learning-based explainability contributes to the thesis from an explainability standpoint.

## 1.4 Thesis Organization

The thesis is organized as follows. Chapter 2 gives a literature review of the graph-based explainable recommender systems, with a focus on explanation types, models, and evaluations. In Chapter 3, I propose FairDGE, an algorithm for recommendation on the fairness issue of dynamic graph data. Chapter 4 introduces the TMER-RL model, which aims to overcome the difficulties of modeling users' historical behaviors on heterogeneous, large,

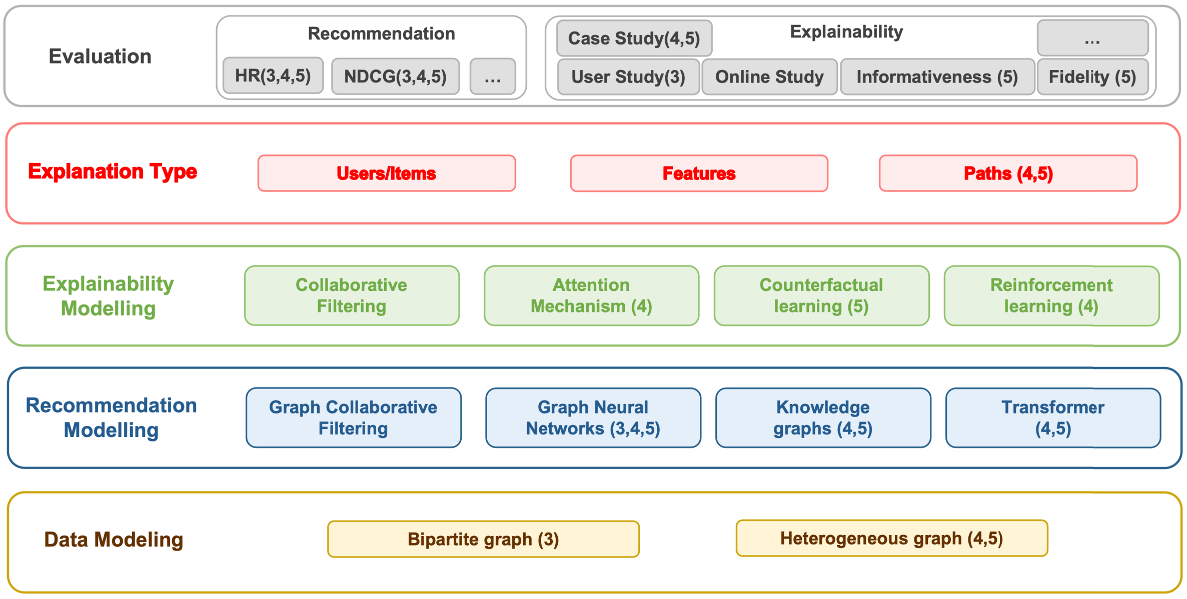


Figure 1.5: The research framework of the thesis. The numbers denote the related Chapter numbers.

and temporal data. Chapter 5 discusses the CPER model, which applies counterfactual learning for explainability modeling. This Chapter also presents a comprehensive framework to assess the quality of explainability in path-based recommendations. Finally, Chapter 6 summarizes the thesis and points out possible areas for future research. The research framework is shown in Figure 1.5.

CHAPTER 2

# LITERATURE REVIEW

In this chapter, I summarize the literature review of the existing graph-based explainable recommendations from the explanation type, models, and evaluation aspects.

## 2.1 Explanation Types for Graph-based Explainable Recommender Systems

The explanations for graph-based explainable recommender systems are the descriptions or expressions for users to understand the rationale of the recommender system's results. They usually provide why the recommender system outputs such recommendations to users. In this taxonomy, the emphasis is on the type of explanation, which encompasses users/items, features, and paths. Table 2.1 lists the summarisation of some typical work.

### 2.1.1 Users/Items for Explanation

The simplest type is to use user(s), item(s), or their combination to explain the recommendations. Most of them use attention mechanisms, designed neural networks or collaborative filtering to learn the importance of user(s), item(s), or their combination for explanations. Following introducing some typical work.

OCuLaR [55] accurately learns overlapping co-clusters of users and items in a bipartite graph according to only positive or implicit feedback, without human-based explicit negative feedback. The approach is scalable, exhibiting near-linear complexity

| Model | Explainability Modeling | Type of Explanation |
|---|---|---|
| OCuLaR [55] | Collaborative Filtering | Users/items |
| TEM [148] | Attention Mechanism | Users and Items |
| UniWalk [102] | Random Walk | Users and Items |
| RetaGNN[91] | Attention Mechanism | Items |
| ELIXIR [42] | Human Feedback | User/item |
| Trirank [50] | Collaborative Filtering | User's and Item's Features |
| DSSM [30] | Latent Weight | User's and Item's Features |
| ACF [19] | Attention Mechanism | Item's Features |
| DEAML [37] | Tree Node Selection | Item's Features |
| [2] | Breadth-first Search | Paths |
| RippleNet [143] | Breadth-first Search | Paths |
| PGPR [159] | Reinforcement Learning | Paths |
| EIUM [61] | Attention Mechanism | Paths |
| KPRN [149] | Attention Mechanism | Paths |
| ERKM [160] | Similarity | Paths |

Table 2.1: Graph-based explainable recommender systems use users/items, features, or paths for explanations.

concerning the number of input examples and co-clusters, making it suitable for large datasets. Even though it provides a template-based sentence for the explanation, the core idea is still to use user/item for explanation.

TEM [148] uses a tree structure to capture hierarchical user-item interactions. This tree structure is built using a clustering algorithm, which groups similar user-item interactions together. The model then uses a dual-attention mechanism to weigh the importance of different interactions and clusters. By examining the tree structure and the weights assigned by the dual-attention mechanism, users will understand why certain items were recommended to them. For example, an item might be recommended because it is similar to other items that the user has interacted with, or because it is popular among users who have similar preferences.

UniWalk [102] uses a unified random walk model to generate topological and attribute-aware node sequences. These sequences are then fed into a recurrent neural network to learn the node embeddings. The recommendation explanation is provided by tracing the random walk paths contributing to the prediction. The paths are ranked by their contributions to the final prediction, and the top-ranked paths are selected as the explanation. This allows to explain the recommendation by showing the sequence of items along the paths that led to the recommendation.

RetaGNN[91] captures dynamic user-item interactions over time and across different relation types. The attention mechanism in the model assigns different weights to different interactions, indicating their importance in the recommendation. The model will explain its recommendations by showing the user-item interactions with the highest weights, indicating that these interactions are the most influential in the recommendation. This allows users to understand why certain items were recommended based on their past interactions and the temporal dynamics of these interactions.

ELIXIR [42] uses user feedback on explanations to improve recommender models. The model generates explanations for its recommendations and then collects user feedback on these explanations. This feedback is used to update the model and improve its future recommendations and explanations. The model will explain its recommendations by showing the user the factors that influenced the recommendation and how user feedback has been used to adjust these factors.

However, using users/items for explanations ignores the abundant information in graphs for explanations, like the item's attributes and so on, resulting in less convincing to the users. Moreover, involving too much user profile information also leads to privacy problems.

### 2.1.2 Features for Explanation

Some researches rely on using the features of users or items to provide explanations. For instance, by utilizing the features of an item, an explanation is offered to a user who has made a similar purchase in the same category. Several studies have employed this approach to provide explanations, as described below.

Trirank [50] uses user reviews to generate explainable recommendations. The model identifies key aspects in the reviews and ranks them based on their importance to the user's preferences. The model then uses these ranked aspects to generate recommendations. The explanations for the recommendations are provided by showing the users the key aspects from the reviews that influenced the recommendation. This allows users to understand why certain items were recommended based on the aspects of the items that they have shown a preference for in their reviews.

DSSM [30] uses multiple views of user data from different domains to generate recommendations. The model uses a deep learning approach to learn a shared user representation across domains, which is then used to generate recommendations. The model will explain its recommendations by showing how different views of the user data contribute to the shared user representation and the final recommendation. For example, it might show that certain features from one domain are particularly important for predicting the user's preferences in another domain. This allows users to understand why certain items were recommended based on their behavior in different domains.

ACF [19] uses attention mechanisms at both the item and component levels to generate recommendations. The model assigns different weights to different items and components (e.g., visual, textual features of items), indicating their importance in the recommendation. The model will explain its recommendations by showing the users the items and components that have the highest weights, indicating that these items and components are the most influential in the recommendation. This allows users to understand why certain items were recommended based on the importance of different items and components.

DEAML [37] uses multiple views of user and item data to generate recommendations. The model uses an attention mechanism to assign different weights to different views, indicating their importance in the recommendation. The model will explain its recommendations by showing the users the views that have the highest weights, indicating that these views are the most influential in the recommendation. This allows users to understand why certain items were recommended based on the importance of different views of the user and item data.

However, using features for explanation lacks interpretability, especially when they are high-dimensional or complex. Users will have trouble understanding what each feature represents and how it influences the recommendation. Therefore, most of the state-of-the-art explainable recommendations use paths for explicit explanation.

### 2.1.3 Paths for Explanation

Many existing graph-based explainable recommendations are based on meta-paths. A meta-path is a sequence of node and edge types defined within a network schema, describing a composite relation between different types of nodes. For example, the meta-path "Author-Paper-Author" signifies the co-author relationship between two authors, while "Author-Paper-Venue-Paper-Author" relates to two authors who published papers in the same venue. While such meta-paths are significant in explaining node relations, they are high-level and general, presenting a schema rather than specific paths. This limitation renders meta-paths less effective in providing precise and detailed explanations.

Additionally, meta-path-based explanations cannot capture the full context of the recommendation. A recommendation is based on multiple meta-paths, and the importance of the meta-paths is different in determining the recommendation. Therefore, a more comprehensive approach that considers multiple meta-paths and their relative importance is be necessary to provide a more accurate and detailed explanation of the recommendation. Although this method has its benefits, it also has some limitations. One disadvantage of utilizing meta-paths is that they often present sets of generalized paths, providing a high-level explanation. Additionally, it tends to overlook the sequential patterns of users' purchase behaviors, which is crucial in generating explanations. Analyzing the sequence of interactions between a user and the recommended items makes it possible to infer patterns and trends that are not apparent in individual interactions. This will be used to create more personalized and relevant recommendations that are tailored to a user's unique preferences and interests.

To avoid the general and high-level explanation, there is another type that uses specific paths to explain. [2] constructs a knowledge graph with entities, including users, items, and items' attributes. It uses a naive breadth-first search (BFS) to explore the shortest path and then generates the textual sentences according to the pre-defined textual templates, but the core of the explanation is still to use the specific path to explain. However, the path exploration strategy of this work is too naive to use BFS, which ignores the sequential information. Later, [149] utilized Long Short-Term Memory (LSTM) to encode the entities sequentially in order to capture their compositional semantics conditioned on relations. However, this approach only considers path-based sequential information between users and items, neglecting the significance of users' clicked history sequences in reflecting their preferences. This is one of the common limitations of existing path-based explainable recommendation methods.

These recent years, most of the graph-based explainable recommendations are path-based explainable recommendations, which means they use paths to explore the user's preference and provide explanations. Among them, RippleNet [143] uses BFS to explore paths between users and items. PGPR [159] uses reinforcement learning to explore paths between users and items. These two works do not consider sequential information. EIUM [61], KPRN [149] and ERKM [160] all generate paths within a pre-defined length and use these paths to explain recommendations. Most of them overlook the users' historical behavior evolution modeling.

## 2.2 Models for Graph-based Explainable Recommender Systems

According to the methodology to learn the explainable weights, I categorize the method as attention-based explainable recommender systems [18, 20, 84, 128, 147, 147], causality-based explainable recommender systems [39, 41, 129, 139, 152, 156, 162, 191], and others [149, 160]. The last category mainly contains pooling-based [149], well-designed score calculations [160], and so on.

### 2.2.1 Attention-based Explainability Modeling

The attention mechanism, proposed in [165], was initially developed for computer vision to enhance the focus on critical areas in image modeling and reduce the focus on trivial areas. As a result of its success in computer vision, the attention mechanism was later integrated into natural language processing for the task of machine translation [3]. Following this breakthrough, the attention mechanism is increasingly being recognized as having a greater potential role in neural networks.

Many graph-based explainable recommender systems in data mining rely on the attention mechanism to learn explainable weights. For instance, a recommendation graph has multiple paths related to the recommendation item for a specific user, which results in redundant information learning and high computing consumption. Thus, most state-of-the-art graph-based explainable recommender systems use the attention mechanism to downsample significant explanations.

This category involves implicit explanations to use latent attention representations and explicit explanations to use realistic entities or relations. AFM [20] adopts the Latent Factor Models (LFMs) to learn the user and item features for recommendation. It proposed the Attention-driven Factor Model (AFM) to learn the item feature's latent and integrate the user's historical interacted items as the user's attention distribution on different item features by Gated Attention Units (GAUs). The weights of different item features for the specific user are regarded as explainable weights. However, AFM only focuses on the item's features as the implicit explanation. Later, DARIA-SARAH [128] proposed a Dual Attention Recommender with Items and Attributes (DARIA) and Self-Attention Recommender based on Attributes and History (SARAH) for both the user's and item's feature-based explainability learning. DARIA relies on users' historical behaviors and learns the dependency between the target item and the historical items. SARAH is similar to DARIA and uses the users' historical behaviors and items' features to train historical items' contribution distribution to the target item. While such methods achieve good performance using attention mechanisms and visible distribution weights, the explanations will still be difficult for humans to understand straightforwardly. Therefore, more work chooses to go a further step to use realistic vertices or relations to explain.

Ante-RNN [84] adopts the attention mechanism to learn both marked images with highlighted areas and sentences with highlighted words for explanations. Specifically, it proposes a novel Attentive Recurrent Neural Network (Ante-RNN) to model users' historical interactions from image latent representation and textual perspectives. KGAT [147] leverages the structure and semantics of knowledge graphs and uses attention mechanisms to model the importance of different entities and relations in the graph for path exploration when making recommendations. KGIN [147] is similar to KGAT which encodes the user intents and item relations into the representations of users and items and then applies a graph attention mechanism to aggregate the information from different users and items. The final output is a recommendation score that reflects the user's preference for each item. The paper also provides explanations for predictions by identifying influential intents and relational paths using attention maps.

However, there are some discussions about the shortcomings of attention-based explainability [13, 43, 119, 154], and we also summarize the points as follows. The attention mechanism is primarily designed for performance enhancement rather than explainability, which means that the quality of attention-based explainability is not guaranteed. Additionally, since the attention mechanism is closely tied to the recommendation process, different attention-based recommendations will result in varying attention weight distributions.

### 2.2.2 Causality-based Explainability Modeling

According to Pearl's Ladder of Causation [106], the causal hierarchy includes associational, interventional, and counterfactual. Among them, the interventional and counterfactual are more popular for explainability learning.

Interventional reasoning relies on the causal graph to perform interventions and observe how they impact the behaviors of users. For instance, CGSR [156] utilizes causal and correlation models to explain recommendations in a session. Specifically, the causality relationship is modeled by a directed graph, where the direction of the edge represents the sequence of item interactions. This helps to capture the transition patterns of items in a session. The correlation relationship is modeled by an undirected graph, where the edge represents the correlation between two items. This helps to capture the co-occurrence patterns of items in a session. The CGSR model integrates these two graphs into a unified framework and learns the item embeddings by preserving the structure of the CGSR. The learned item embeddings are then used to compute the recommendation scores. The method also provides explanations for the recommendations by identifying the most influential paths in the CGSR that lead to the recommended item. These paths represent the sequence of item interactions and co-occurrences that contribute to the recommendation, providing insights into why the item is recommended.

Moreover, counterfactual reasoning has also come up as an alternative explainable approach [139, 152]. It answers the ’what-if’ causal question that if some condition did not occur, the result would not happen. According to the idea, the original condition is an essential reason for the result. In the early work, CountER [129] proposes a model-agnostic explainable recommendation to learn slight perturbation on item aspects for counterfactual reasoning exploration. Later, CNR [191] also proposes a similar model-agnostic explainable recommendation to perform perturbation on user features for flipping the model decision to learn the counterfactual explanation. To make better use of rich data of recommendation, CF$^2$[162] and CEF [39] utilize review data of items to enrich the information for training counterfactual reasoning. However, the counterfactual reasoning-based recommendations mentioned above mainly focus on providing aspect or item explanations and do not take into account the use of knowledge graphs or graph neural network modeling to generate composite relations (such as paths) for the explanation. Although PRINCE [41] proposes a path-based counterfactual explainable recommendation through removing edges as a perturbation, the explainable model is model-specific, which is inflexible for explaining other black-box recommendations. This limits the method’s generalizability and requires significant modifications to be used with different models.

To deal with the aforementioned shortcomings, I propose a novel explainable framework for path-based recommendation through counterfactual reasoning. The framework is flexible to be applied to other path-based recommendation systems and generates counterfactuals by perturbing both the path representation and the path topological structure. Specifically, I introduce a novel reinforcement learning method to learn how to perturb the path topological structure through node selection and removal.

### 2.2.3 Other Explainability Modeling

There are also many well-designed neural networks or functions for explainability weights learning.

KPRN [149], a pooling-based explainability modeling, proposes to use LSTM to model the entities along the path for path representation learning. Each path from the specific user to the target item presents an explanation for purchasing the item. Finally, it proposes a pooling operation to aggregate paths' representations for prediction score learning, and the learned contribution weights are regarded as the explanation weights.

ERKM [160] first constructs a knowledge graph using information about items and their attributes to generate candidate recommendations. Then, a multi-objective optimization algorithm is used to rank these candidates based on multiple criteria, such as relevance and diversity. The algorithm aims to balance these criteria to generate a list of recommendations. The explanation, calculated from the well-designed criteria, demonstrates how the method generates high-quality recommendations. For example, if a user is recommended a movie, the method will explain this recommendation by pointing out that the movie has similar attributes to other movies the user has liked in the past, or that it features actors or directors the user likes.

## 2.3 Evaluations

The explainable recommendation requires evaluations not only on recommendation performance but also on explainability. Following, I summarize commonly used evaluations.

### 2.3.1 Evaluation for Recommendations

To evaluate the recommendation performance, usually consider top $k$ recommendation candidates for evaluation. Specifically, the following list three performance evaluation metrics: top $k$ Hit Rate (HR@$k$), top $k$ Normalized Discounted Cumulative Gain (NDCG@$k$), and top $k$ Precision (PREC@$k$).

- **HR@$k$** calculates the recall ratio of the prediction list.

  $$HR@k = \frac{\#Hits@k}{|pos|} \tag{2.1}$$

  where $|pos|$ is the number of all positive items. $\#Hits@k$ is the number of top $k$ prediction list hit to the positive items.

- **NDCG@$k$** calculates the normalized ranking of the recall ratio of the prediction list.

  $$NDCG@k = \frac{1}{|pos|} \sum_{i}^{|pos|} \frac{1}{log_2(p_i^k + 1)} \tag{2.2}$$

  where $p_i^k$ is item $i$'s ranking in the top $k$ prediction list.

- **PREC@$k$** measures the precision of the prediction list.

$$PREC@k = \frac{\sum_{u \in U} pred_k(u) \bigcap pos(u)}{\sum_{u \in U} pred_k(u)} \tag{2.3}$$

where $U$ is the user set, $pred_k(u)$ is the top $k$ prediction list for user $u$, and $pos(u)$ is $u$'s positive item list.

### 2.3.2 Evaluation for Explainability

The following introduces some typical evaluations for explainability.

- **Case Study** needs to select a specific instance or scenario where a recommendation is made by their system. They would then analyze and demonstrate how the system arrived at that particular recommendation, using the knowledge graph, multi-objective optimization, or the learned weights. For example, if the system recommends a product to a user, the case study would trace back the attributes of the product in the knowledge graph, such as the category or brand, and show how these attributes align with the user's past preferences or interactions. Through case studies, readers will assess whether the explanation provided by the system is understandable and makes sense to the users. The case study is the most popular evaluation for the explainability of recommendations.

- **User Study** needs to design an interview or report for the users to rate the quality of the explanations or indicate whether the explanations helped them understand why a particular item was recommended. The interview or the report usually includes criteria or questions for the users to answer in an online or offline way. For example, if the system provides both recommendations and explanations to a user, the user study should design a series of questions for the users to rate the quality of the provided explanations. However, user studies usually involve large human work which is inconvenient and expensive for a large amount of research work. Therefore, this evaluation is not as popular as the case studies.

There are also other evaluations for explainability for specific types of explanation. For example, when the explanation is of textual sentence type, it is suitable for BLEU [100] and ROUGE [82] evaluations. There is also an evaluation, named Model Fidelity [105], to evaluate the explainable items ratio of all the recommended items, calculated as

$$Model\ Fidelity = \frac{|explainable\ items| \cap |recommended\ items|}{|recommended\ items|} \tag{2.4}$$

This evaluation only evaluates the model's explainability but not the learned explanations' quality. Another evaluation, named Fidelity [5, 109], evaluates the quality of the learned explanations, but it requires that recommendations rely on the learned explanations. Specifically, fidelity requires feeding the learned explainable items back to the recommendation, and seeing how the recommendation score changes. The decreased score of the recommendation is the fidelity score. A larger fidelity value shows a larger contribution of the learned explanations to the recommendation, indicating better explainability.

The evaluations for explainability still need universal metric-based assessments to make it easier and more convenient. However, due to the difficult definition of explanation quality in a mathematical way, this problem is not been widely addressed. In this thesis, we push forward this area by proposing a package of solutions including both qualitative and quantitative evaluations.

# Chapter 3

# Toward Structure Fairness in Dynamic Graph Embedding

For graph data, this Chapter goes deep into the graph structure evolution in dynamic scenarios and identifies the unfairness issue, which impedes the performance of both recommendation and explanation. To deal with the issue, I propose a novel fair dynamic graph embedding algorithm to alleviate the unfairness. The fair graph modeling will benefit the downstream task, i.e., recommendation and explanation.

## 3.1 Overview

Graph embedding has achieved great success in many web applications, such as online advertising, social inference, risk assessment, etc. Many of these applications are human-centered, where fairness matters a lot [111]. For example, recommending jobs unbiasedly, ensuring every demographic has the same opportunities, ensuring algorithmic credit scoring for lending does not unintentionally favor certain ethnic or age groups, providing equitable access and connectivity in transportation networks, etc. This drives the research on fair graph embeddings.

Fair graph embedding aims to learn low-dimension vertex/edge representations that will not make disparate treatment or impacts to vertices/edges in downstream graph mining tasks based on their sensitive attributes and/or connectivity [70]. Disparate treatment, which is direct discrimination, intentionally treats vertices/edges differently based on sensitive attributes [8]. Disparate impact is indirect discrimination, where algorithms perform much worse on vertices/edges with sensitive attributes and/or connectivity than on others [8].

Existing works successfully learn fair embeddings for static graphs. Some studies employed adversarial training [27], fairness regularizations [36, 39, 74, 123], resampling [132, 183], or graph argumentation [21, 172] to make sensitive vertex attributes transparent to the embeddings, thus eliminating disparate treatment or impacts to sensitive attributes. Other studies [72, 87, 131, 146] learned static graph embeddings that are structurally fair by preventing the effectiveness disparity of high- and low-degree vertex groups (i.e., disparate impact) in downstream graph mining tasks, given that vertex degrees of real-world graphs usually follow a long-tailed power-law distribution [7].

Despite the success of learning fair embeddings for static graphs, achieving structure fairness in dynamic graph embedding remains an open problem. Naively re-training the above structurally fair static graph embedding methods from scratch at each timestamp is time-consuming and fails to deal with the bias caused by structural changes in dynamic graphs. As the graph structure evolves, a minority of lower-degree vertices (*Tail*) may become higher-degree vertices (*Head*) by actively forming edges with other vertices, while most tail vertices may change slightly. If the model prioritizes fairness by reducing head and tail vertices' effectiveness disparity without considering their evolutionary patterns, the performance of head and tail-to-head vertices will significantly drop close to generally poorer embedding performance of most slightly altered vertices in the long-tail part of the power-law distribution. As a result, the effectiveness of learned structurally fair embeddings in downstream tasks is questionable.

In this Chapter, I study a critical but overlooked problem of learning structurally fair dynamic graph embeddings that are highly effective in downstream graph mining tasks. To achieve this goal, there are two major challenges. (1) Learning biased structural evolution over time. The structural evolution of head and tail-to-head vertices leaves more linkage-changing information than fluctuation tail vertices, giving rise to the structure bias in the embeddings. Such structure bias changes as the graph evolves, and not all structural evolutions bias embeddings. It is critical to learn biased structural evolution for later debiasing. (2) Debiasing properly to achieve fairness without sacrificing or even improving embedding effectiveness. Debiasing unbiased structural evolution impairs embedding effectiveness without notably improving structure fairness. Structurally fair embedding algorithms should customize debiasing strategies for different biased structural evolutions such that the effectiveness of embedding is not greatly sacrificed.

In light of these challenges, I first identify three biased structural evolutions in a dynamic graph based on the evolutionary trend of vertex degree, i.e., *Fluctuation-at-Tail (FaT)*, *Tail-to-Head (T2H)*, and *Starting-from-Head (SfH)*, then innovatively propose **FairDGE**, a structurally **Fair D**ynamic **G**raph **E**mbedding algorithm. To learn biased structural evolution over time, FairDGE jointly embeds the connection changes among vertices and the long-short-term evolutionary trend of vertex degrees by an intermedia training task that classifies the identified biased structural evolutions. Next, I devise a dual debiasing approach for FairDGE to encode fair embeddings that are highly effective in downstream tasks. It first debiases the embeddings via contrastive learning, bringing closer the embeddings with identical biased structural evolutions and penalizing those with different ones. Thanks to the theoretical proof in [146], the embedding performance of the FaT vertices will be much improved, resulting in a smaller performance gap with T2H and SfH vertex groups. Meanwhile, I minimize the effectiveness disparity of embeddings of T2H and SfH vertex groups, which are in the head of the power-law distribution and with high performance, to boost fairness further. Consequently, FairDGE breaks the effectiveness bottleneck of embeddings with almost no fairness loss.

The contributions of this Chapter are highlighted as follows:

- **A new problem.** To the best of my knowledge, I am the first to study the structure fairness problem in dynamic graph embedding and identify three biased structural evolutions that are *Fluctuation-at-Tail (FaT)*, *Tail-to-Head (T2H)*, and *Starting-from-Head (SfH)*.

- **A new and effective approach.** I propose FairDGE, a novel dynamic graph embedding algorithm, that learns biased structural evolutions and then uses a newly devised dual debiasing method for encoding structurally fair embeddings that are highly effective in downstream graph mining tasks.

- **Extensive experiments.** The experimental results demonstrate that FairDGE achieves simultaneous improvement in the effectiveness and fairness of embeddings.

## 3.2 Preliminary

### 3.2.1 Problem Formulation

I give the definition of dynamic graphs and formulate the problem of structurally fair dynamic graph embedding.

Firstly, I define a dynamic graph as a snapshot graph sequence.

**Definition 1.** ***A Dynamic Graph.*** *A dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T)$ is a sequence of directed or undirected snapshot graphs $\mathcal{G}_t$, where $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ is a static graph at time $t \in \{1,2,...,T\}$. $\mathcal{V}_t$ is a subset of the vertex set $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$. An edge $e^t_{i,j} = (v^t_i, v^t_j) \in \mathcal{E}_t$ represents the connection between vertices $v^t_i$ and $v^t_j$ in $\mathcal{G}_t$, where $v^t_i, v^t_j \in \mathcal{V}_t$ and $\mathcal{E}_t$ is a subset of the edge set $\mathcal{E}$.*

Similar to the existing work [146], I leverage statistical parity [29] as the fairness metric, as in Definition 2. In other words, the smaller the intergroup performance disparity, the better the fairness.

**Definition 2.** ***Fairness Metric.*** *Given $q$ mutually exclusive vertex groups $\{G_1, ..., G_q\}$ and $G_1 \bigcup ... \bigcup G_q = \mathcal{V}$, fairness is achieved when $\mathcal{P}(h_v | v \in G_1) = ... = \mathcal{P}(h_v | v \in G_q)$, where $\mathcal{P}(\cdot)$ indicates the performance of $v$'s embedding $h_v$ in a downstream task.*

Dynamic graph embedding aims to learn low-dimensional representations (i.e., embeddings) for every vertex that preserves its dynamic connectivity changes. The learned embedding should be highly effective in downstream graph mining tasks such as high classification accuracy, small regression errors, high hit rate in recommendation, etc. Achieving structure fairness in dynamic graph embeddings requires the embeddings learned from different structural evolution groups to perform similarly in downstream graph mining tasks. To this end, I give the formal problem formulation of structurally fair dynamic graph embedding in Definition 3.

**Definition 3.** ***Structurally Fair Dynamic Graph Embedding.*** *Given a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T)$ and $\mathcal{V}$ is divided into $q$ mutually exclusive vertex groups $\{G_1, ..., G_q\}$ based on their structural evolution patterns, and $G_1 \bigcup ... \bigcup G_q = \mathcal{V}$, the objective is to learn a mapping function $f : v \mapsto h_v \in \mathbb{R}^{dim}$ for $\forall v \in \mathcal{V}$ such that the representation $h_v$ preserves the structural evolution of $v$ in terms of dynamic connectivity changes between $v$ and other vertices in $\mathcal{V}$ over time, achieves high performance in downstream graph mining tasks, and satisfies the fairness requirement defined in Definition 2, where $dim$ is a positive integer indicating the dimension of $h_v$.*

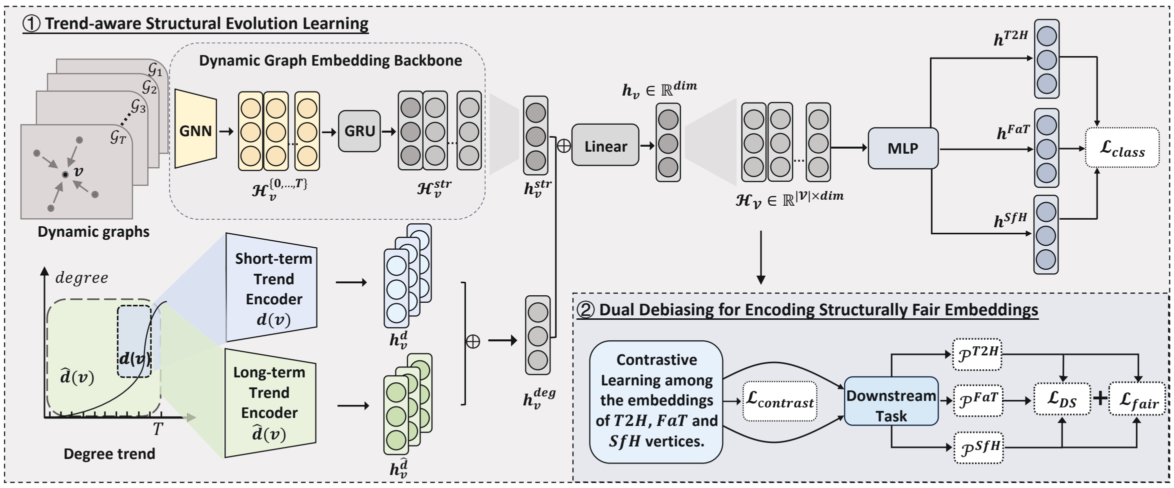

① Trend-aware Structural Evolution Learning
Dynamic Graph Embedding Backbone
GNN
GRU
Linear
MLP
Dynamic graphs
degree
Degree trend
Short-term Trend Encoder d(v)
Long-term Trend Encoder d̂(v)
② Dual Debiasing for Encoding Structurally Fair Embeddings
Contrastive Learning among the embeddings of T2H, FaT and SfH vertices.
Downstream Task

Figure 3.1: Overall framework of FairDGE.

### 3.2.2 Identifying Biased Structure Evolutions of Dynamic Graphs

Following, I explain intuitions for identifying the biased structural evolution of dynamic graphs and explain the fairness implications for dynamic graph embeddings.

The degree of a vertex represents the number of other vertices connected to it, which indicates its neighborhood structures. As the graph structure evolves, the connections among vertices change accordingly, causing vertices' degrees to vary. Therefore, degree change is capable of approximating the vertex's neighborhood structure evolution over time.

Vertex degrees of real-world graphs usually follow a long-tailed power-law distribution [7]. High- and low-degree vertices are in the head and tail parts of the distribution, respectively. When the vertex degree changes as the graph structure evolves, several evolution patterns will be identified based on the evolving trend of degrees, including *Fluctuation-at-Tail (FaT)*, *Tail-to-Head (T2H)*, *Head-to-Tail (H2T)*, and *Fluctuation-at-Head (FaH)*.

| | FaT | | T2H | | H2T | | FaH | |
|---|---|---|---|---|---|---|---|---|
| Snapshots | $\mathcal{G}_1$ | $\mathcal{G}_2$ | $\mathcal{G}_1$ | $\mathcal{G}_2$ | $\mathcal{G}_1$ | $\mathcal{G}_2$ | $\mathcal{G}_1$ | $\mathcal{G}_2$ |
| Avg. Degree | 1.171 | 0.936 | 0.869 | 9.345 | 16.362 | 3.447 | 29.594 | 23.237 |
| Std. Degree | 1.414 | 1.000 | 1.732 | 13.153 | 9.849 | 2.646 | 23.979 | 20.396 |
| Vertex Ratio | 90.67% | | 7.02% | | 1.45% | | 0.86% | |

Table 3.1: Statistics on the evolutionary trend of vertex degrees.

I conduct a statistical analysis on six months of data in the Amazon Books dataset containing 64,425 vertices and 217,163 edges. I construct two snapshot graphs using data from the first three months and the last three months. The head and tail groups are divided by a degree threshold of 10 to identify the evolving trend of degrees across snapshot graphs. The results in Table 3.1 show that the above-identified evolving trend of degrees approximately follows the long-tailed power-law distribution as 90.67% vertices fluctuate at the tail with slight connection changes while only 9.33% vertices' neighborhood structures vary a lot. This makes dynamic graph embedding algorithms exhibit structure unfairness [87, 131, 146], favoring highly active structural evolutions (i.e., T2H, H2T, and FaH) too much yet discriminating inactive ones (i.e., FaT). Thus, I call them the biased structural evolutions of the dynamic graph.

Since the number of vertices in H2T and FaH is small, I combine them in a single group called *Start-from-Head (SfH)* in the rest of this Chapter. As for complex structural evolution patterns such as first rising and then falling, first falling and then rising, etc., I leave it to future work.

## 3.3 Methodology

Following, I present the details of FairDGE to learn structurally fair dynamic graph embeddings. The symbol notations used in the remainder of this Chapter are listed in Table 3.2.

### 3.3.1 Overall Framework of FairDGE

The overall framework of FairDGE is presented in Figure 3.1. FairDGE consists of two modules that are trained jointly. I design a trend-aware structural evolution learning module to embed the connection changes among vertices as well as the evolving trend of the corresponding degrees. An intermedia training task that classifies the identified biased structural evolutions (i.e., FaT, T2H, and SfH) is employed to supervise the learning of structural evolutions.

Another dual debiasing module is devised to encode structurally fair embeddings. It customizes debiasing strategies for FaT, T2H, and SfH vertex groups, respectively. In particular, I first debias the embeddings via contrastive learning, bringing closer the embeddings with identical biased structural evolution and penalizing those with different ones. Hence, the embedding performance of FaT vertices will be significantly improved and close to that of T2H and SfH. Then, I minimize the effectiveness disparity

| Symbols | Descriptions |
|---|---|
| $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ | Static snapshot graph, vertex set, edge set at time $t$ |
| $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ | Dynamic graph, vertex set, edge set |
| $h \in \mathcal{H}$ | A representation vector (embedding) |
| $\mathcal{H}_{\mathcal{V}}^{T2H}$ | Tail-to-head vertex representations |
| $\mathcal{H}_{\mathcal{V}}^{SfH}$ | Starting-from-head vertex representations |
| $t \in T$ | Timestamps |
| $q$ | The number of vertex groups |
| $dim$ | Representation dimensions |
| $\oplus$ | Stack operator |
| $\varpi$ | Head/tail ratio threshold |
| $d(\cdot)$ | Short-term trend encoder |
| $\hat{d}(\cdot)$ | Long-term trend encoder |
| $deg(\cdot)$ | Degree of vertex |
| $y \in Y$ | Labels of biased structural evolution |
| $(v, v^+)$ | Positive vertex pair from same structural evolution group |
| $(v, v^-)$ | Negative vertex pair from different structural evolution groups |
| $\mathcal{L}_{class}$ | Classification loss |
| $\mathcal{L}_{contrast}$ | Contrastive loss |
| $\mathcal{L}_{DS}$ | Downstream task loss |
| $\mathcal{L}_{fair}$ | Fairness loss |
| $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ | Loss hyperparameters |
| $\|\|\Theta\|\|_1, \|\|\Theta\|\|_2$ | $\ell_1$, $\ell_2$ norm of model parameters |
| $p_e$ | Probability of edge $e$ existing |
| $k$ | Top $k$ prediction list |

Table 3.2: Symbol descriptions.

of T2H and SfH vertices' embeddings in downstream tasks to boost fairness further and simultaneously avoid dragging their embedding effectiveness down by FaT vertices with relatively lower performance.

## 3.3.2 Embedding Trend-aware Structural Evolutions in Dynamic Graphs

I present the details of the trend-aware structural evolution learning module to embed the FaT, T2H, and SfH vertices in a dynamic graph, solving the first challenges identified in Chapter 3.1.

#### 3.3.2.1 Learning Evolving Trend of Degrees

Calculating the degree difference between the first and last timestamp to measure the trend will lose a lot of information about short-term changes. I first model the degree of vertex $v$ across the snapshot graph sequence $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T)$ as a time series, denoting as $deg(v) \in \mathbb{R}^T$. Then, I devise a long-short-term trend encoder to learn the comprehensive varying patterns of $deg(v)$, including long-term evolving trends and short-term fluctuation patterns.

Specifically, I employ a 3-layer Gate Recurrent Unit (GRU) [24] as the long-term trend encoder, denoting as $\hat{d}(v)$, to learn the long-term evolving patterns of $deg(v)$ as follows:

$$h_v^{\hat{d}} = \hat{d}(v) = GRU(expand(deg(v))) \tag{3.1}$$

where $expand(deg(v))$ expands the dimension of $deg(v)$ from $T \times 1$ to $T \times dim$ by duplication.

To capture and embed the short-term fluctuation patterns of degrees, I employ a 1-d convolutional neural network (CNN) as the short-term trend encoder $d(v)$.

$$h_v^{d} = d(v) = CNN(deg(v)) \tag{3.2}$$

I set the kernel's size and padding to 3 and 1, respectively. Therefore, $h_v^d$ keeps the same dimension as $h_v^{\hat{d}}$.

Lastly, I fuse $h_v^d$ and $h_v^{\hat{d}}$ to obtain the trend embedding of $deg(v)$ by a stack-mean operation as shown below.

$$h_v^{deg} = m([h_v^d \oplus h_v^{\hat{d}}]) \tag{3.3}$$

where $h_v^{deg}$ is the trend embedding of $deg(v)$, $\oplus$ is a stack operator for two embeddings, and $m(\cdot)$ is the mean operation.

#### 3.3.2.2 Embedding Structural Evolutions

Embedding the connection changes among vertices, noting as the structural evolutions, in the snapshot graph sequence $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T)$ is a typical dynamic graph embedding problem, which has been well studied [9, 101, 116, 167, 168, 171, 176]. Since it is not the main focus of this study and many existing algorithms [9, 101, 116, 167, 168, 171, 176] is capable to be directly applied, I use a graph neural network (GNN) to embed the snapshot graph at each timestamp and then employ a GRU to learn the sequential changing pattern across the snapshot graph sequence and obtain structural evolution embeddings.

$$\mathcal{H}_{\mathcal{V}}^{str} = GRU(GNN(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T)) \tag{3.4}$$

where $\mathscr{H}_{\mathcal{V}}^{str}$ is the embedding of all vertices in the dynamic graph.

I treat this as a dynamic graph embedding backbone in which any existing dynamic graph embedding algorithms are able to plug in to learn the structural evolution embeddings.

#### 3.3.2.3 Learning Trend-aware Structural Evolutions via An Intermedia Trend Classification Task

To embed the trend-aware structural evolutions, I fuse the degree-trend embedding of $h_v^{deg}$ and the structural evolution embedding $h_v^{str}$ for every vertex $v$, as shown in Eq. (3.5) where $h_v^{str}$ comes from $\mathscr{H}_{\mathcal{V}}^{str}$ for $v \in \mathcal{V}$, $[.;.]$ is the concatenation operation and $g(.)$ is a linear layer.

$$h_v = g([h_v^{deg}; h_v^{str}]) \tag{3.5}$$

I leverage an intermedia training task to better train $h_v$ for retaining the biased structural evolutions FaT, T2H, and SfH. A two-layer Multi-Layer Perceptron (MLP) is employed to encode the representation, followed by a softmax function to forecast the probability of vertex $v$ belonging to one of the biased structural evolutions, i.e., FaT, T2H, or SfH.

$$h_v^{SfH,T2H,FaT} = MLP(h_v) \tag{3.6}$$

$$h_{v,y} = softmax(ReLU(h_v^{SfH,T2H,FaT})) \tag{3.7}$$

where $h_{v,y}$ indicates the probability of vertex $v$ that belongs to a class $y \in Y$ = {SfH, T2H, FaT}. Lastly, I employ a cross-entropy loss to train $h_v$ and $h_v^{SfH,T2H,FaT}$, which is

$$\mathscr{L}_{class} = -\sum_{v \in \mathcal{V}} log \frac{\exp(h_{v,y_v})}{\sum_{y=1}^{Y} \exp(h_{v,y})} \tag{3.8}$$

$h_{v,y_v}$ is a one-hot vector indicating the ground truth biased structural evolutions of $v$, which is manually labeled from the historical data for training. Due to the ambiguous degree boundary of head and tail vertices, finding a golden standard to determine the threshold of head and tail degrees is difficult. Therefore, I devise a slope-based method to identify the degree evolving trend of SfH, T2H, and FaT. Details are in the Algorithm 1. The threshold of degree changing $\rho$ is set to 0.

**Algorithm 1** Degree evolving trend labeling.

```
Input: Vertex set $\mathcal{V}$, timestamp range $T$, head/tail ratio threshold $\varpi$, degree variation threshold $\rho$
Output: Label set $Y$ for $\mathcal{V}$, where each vertex $v$'s bias label $y \in \{FaT, T2H, SfH\}$
 1: head_num = $|\mathcal{V}| \times \varpi$
 2: head_group = sort(deg($\mathcal{V}$))[:head_num]
 3: for vertex $v$ in $\mathcal{V}$ do
 4:   if $v_t$ in head_group then
 5:     $y_v = SfH$
 6:   else
 7:     $argmax(deg(v_T))$ //Find the timestamp of $v$'s largest degree
 8:     $argmin(deg(v_T))$ //Find the timestamp of $v$'s least degree
 9:   end if
10:   if $argmax(deg(v_T)) - argmin(deg(v_T)) > \rho$ then
11:     $y_v = T2H$
12:   else
13:     $y_v = FaT$
14:   end if
15: end for
16: return $Y$
```

### 3.3.3 Dual Debiasing for Encoding Structurally Fair Embeddings

I devise a dual debiasing module that customizes debiasing strategies for FaT, T2H, and SfH vertices to encode structurally fair embeddings, overcoming the second challenge. The idea is to make the embedding performance of SfH and T2H vertices high and close while improving the embedding performance of FaT vertices as much as possible to narrow the gap with that of SfH and T2H, eventually achieving structure fairness without sacrificing or even improving embedding effectiveness.

Wang et al. [146] had theoretically and empirically proved that contrastive learning will improve the tail vertex's performance and alleviate the performance gap between the head and tail vertex groups, eventually improving fairness. Inspired by this, I employ contrastive learning to debias the embedding of FaT, T2H, and SfH vertices. Specifically, for each vertex $v$, I randomly select two vertices $v^+$ and $v^-$ from $\mathcal{V}$ to form a positive pair $(v, v^+)$ and a negative one $(v, v^-)$ in which $v$ and $v^+$ are in the same structural evolution group but $v$ and $v^-$ are not. The objective is to make the embeddings of the positive pairs as close as possible and the representations of the negative pairs as far as possible, bringing closer the embeddings with identical structural evolutions and penalizing those with different ones. A contrastive loss is proposed in Eq. (3.9) to achieve this goal, where

$h_v$, $h_{v^+}$, $h_{v^-}$ are the embeddings of vertex $v$, $v^+$ and $v^-$, respectively.

$$\mathscr{L}_{contrast} = -log \frac{exp(sim(h_v, h_{v^+}))/\tau}{exp(\frac{sim(h_v,h_{v^+})}{\tau}) + \sum_{v^- \in \mathcal{V}^-} exp(\frac{sim(h_v,h_{v^-})}{\tau})} \tag{3.9}$$

$sim(h_v, h_{v^+})$ is measuring the cosine similarity between $h_v$ and $h_{v^+}$. $\tau$ is a scaling parameter. $\mathcal{V}^-$ is a set of negative vertices with different structural evolutions from $v$.

Additionally, I alleviate the effectiveness disparity of T2H and SfH vertices' embeddings in downstream tasks for a second debiasing, facilitated by a fairness loss for the embeddings of T2H and SfH vertices in Eq. (3.10).

$$\mathscr{L}_{fair} = ||\mathscr{L}_{DS}(H_{\mathcal{V}}^{T2H}) - \mathscr{L}_{DS}(H_{\mathcal{V}}^{SfH})||_2 \tag{3.10}$$

where $H_{\mathcal{V}}^{T2H}$ and $H_{\mathcal{V}}^{SfH}$ are the final learned embeddings of T2H and SfH vertices, respectively. $||\cdot||_2$ denotes the $\ell_2$ norm. $\mathscr{L}_{DS}(\cdot)$ is a loss function of the downstream task, and I use the recommendation task in this Chapter, mathematically,

$$\mathscr{L}_{DS} = -\sum_{v}^{|\mathcal{V}|}\sum_{e}^{|\mathscr{E}_v|} x_e \cdot logp_e + (1-x_e)\cdot log(1-p_e) \tag{3.11}$$

$\mathscr{E}_v$ is an edge set of $v$ for training, which contains the positive interactions and negative interactions, and the ratio is the same. $p_e$ is a learned interaction probability computed by inputting the final embeddings to a Fermi-Dirac decoder [97].

Lastly, I jointly minimize all the loss items mentioned above with $\ell_2$ and $\ell_1$ regularization of model parameters $\Theta$.

$$\mathscr{L} = \gamma_1\mathscr{L}_{DS} + \gamma_2\mathscr{L}_{class} + \gamma_3\mathscr{L}_{contrast} + \gamma_4\mathscr{L}_{fair} + ||\Theta||_2 + ||\Theta||_1 \tag{3.12}$$

where $\gamma_1$, $\gamma_2$, $\gamma_3$, and $\gamma_4$ are the weight hyper-parameters to control the proportion of different components in the loss function.

## 3.4 Experiments

I conduct extensive experiments to validate the embedding effectiveness of FairDGE in terms of performance and fairness.

### 3.4.1 Experiment Settings

#### 3.4.1.1 Datasets

The experiments are respectively conducted on small-, medium- and large-scale real-world datasets including Amazon Books, MovieLens and GoodReads. In the Amazon

| Dataset | # User | # Item | # Vertex | # Edges | Dens. | # Snapshots |
|---|---|---|---|---|---|---|
| Amazon Books | 4,054 | 17,651 | 21,705 | 72,317 | 0.1% | 5 |
| MovieLens | 2,342 | 5,579 | 7,921 | 477,419 | 3.65% | 15 |
| GoodReads | 16,701 | 20,823 | 37,524 | 1,084,781 | 0.3% | 15 |

Table 3.3: Dataset details of FairDGE.

Books dataset, the user-book interactions from 09 Aug 2006 to 12 Aug 2007 are selected. For the MovieLens dataset, I use the 10M movie ratings from 07 Jan 2001 to 07 Jan 2003. The GoodReads dataset is from an online book community website, and I select the Children's books from 1 Jan 2013 to 30 Dec 2015. The statistics of the three datasets are shown in Table 3.3.

#### 3.4.1.2 Baseline Methods

To comprehensively benchmark FairDGE, I carefully select eight baseline methods for performance comparison, including static graph learning methods GCN [69] and GAT [138], dynamic graph embedding methods MPNN-LSTM [99] and EvolveGCN [101], and fair graph learning methods FairGNN [27], FairVGNN [151], DegFairGNN [87], and TailGNN [86].

<u>Graph learning methods:</u>

- GCN [69]: GCN adopts the graph convolutions to learn the graph embeddings.
- GAT [138]: GAT leverages masked self-attentional layers to specify different weights to different vertices in a neighborhood.

<u>Dynamic graph learning methods:</u>

- MPNN-LSTM [99]: MPNN-LSTM proposes an LSTM-based message passing mechanism to model the vertices evolution in the dynamic graphs.
- EvolveGCN [101]: EvolveGCN combines the graph convolutional network model with an RNN to capture temporal dynamics in a graph sequence.

<u>Fair graph learning methods:</u>

- FairGNN [27]: FairGNN effectively mitigates the impact of sensitive attributes in graph data by incorporating fairness constraints during training.

- FairVGNN [151]: FairVGNN is an extension of the FairGNN model that includes a variational graph autoencoder to learn latent representations and generate fair and diverse outputs in graph data.
- DegFairGNN [87]: DegFairGNN is a fair graph neural network to alleviate the structure bias problem using an equal opportunity fair loss.
- TailGNN [86]: TailGNN proposes transferable neighborhood translation aiming to improve the tail vertices' modeling to close the gap between the performance of head and tail vertices.

#### 3.4.1.3 Evalution Metrics of Performance

I adopt three performance evaluation metrics of recommendation tasks in the experiments: top $k$ Hit Rate (HR@$k$), top $k$ Normalized Discounted Cumulative Gain (NDCG@$k$), and top $k$ Precision (PREC@$k$). The details and mathematic equations are introduced in Chapter 2.3.

#### 3.4.1.4 Evaluation Metrics of Fairness

I adopt the commonly used ranking discrepancy metrics to quantitatively evaluate fairness. In particular, I set $k \in K = \{1,5,10,15,20,40,60,80,100\}$ and calculate the average Hit Ratio Difference (rHR) and Normalized Discounted Difference (rND) as fairness metric [166].

- **Hit Ratio Difference (rHR)** calculates the difference between the T2H items' ratio in the top $k \in K$ candidates and that in the overall ranking. A similar ratio, leading to less rHR, denotes more fairness. Mathematically,

$$rHR = \frac{1}{Z}\sum_{k}^{K}\left|\frac{|T2H_k|}{k} - \frac{|T2H|}{|\mathcal{V}|}\right| \tag{3.13}$$

  where $|T2H_k|$ denotes the number of T2H vertices in the top $k$ candidate list and $|\mathcal{V}|$ presents the total number of vertices in the dynamic graph. Given $|\mathcal{V}|$ and $|T2H|$, $Z$ is the highest possible value of rHR for normalization.

- **Normalized Discounted Difference (rND)** calculates the difference between the T2H items' ranking order in the top $k \in K$ candidates and that in the overall ranking. The smaller the rND, the better the fairness.

$$rND = \frac{1}{Z}\sum_{k}^{K}\frac{1}{log_2 k}\left|\frac{|T2H_k|}{k} - \frac{|T2H|}{|\mathcal{V}|}\right| \tag{3.14}$$

| Datasets | Amazon Books | | | | | MovieLens | | | | | GoodReads | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | Performance (↑) | | | Fairness (↓) | | Performance (↑) | | | Fairness (↓) | | Performance (↑) | | | Fairness (↓) | |
| Baselines | HR@20 | NDCG@20 | PREC@20 | rHR | rND | HR@20 | NDCG@20 | PREC@20 | rHR | rND | HR@20 | NDCG@20 | PREC@20 | rHR | rND |
| GCN | 0.0738 | 0.0289 | 0.0037 | 0.1559 | 0.2627 | 0.0961 | 0.0432 | 0.0048 | 0.3259 | 0.5615 | 0.0683 | 0.0273 | 0.0034 | 0.3710 | 0.6839 |
| w/ $\mathscr{L}_{fair}$ | 0.0858 | 0.0344 | 0.0043 | 0.1547 | 0.2551 | 0.0880 | 0.0405 | 0.0044 | 0.3244 | 0.5525 | 0.0766 | 0.0310 | 0.0038 | 0.3703 | 0.6639 |
| GAT | 0.0646 | 0.0262 | 0.0032 | 0.1569 | 0.2765 | 0.1106 | 0.0492 | 0.0055 | 0.3567 | 0.6136 | 0.0788 | 0.0301 | 0.0039 | 0.3843 | 0.6391 |
| w/ $\mathscr{L}_{fair}$ | 0.0651 | 0.0255 | 0.0033 | 0.1550 | 0.2649 | 0.0807 | 0.0360 | 0.0040 | 0.3538 | 0.6085 | 0.0787 | 0.0302 | 0.0039 | 0.3838 | 0.6387 |
| MPNN-LSTM | 0.1297 | 0.0499 | 0.0065 | 0.1557 | 0.2671 | 0.0376 | 0.0129 | 0.0019 | 0.3440 | 0.5931 | 0.0343 | 0.0125 | 0.0017 | 0.3713 | 0.6713 |
| w/ $\mathscr{L}_{fair}$ | 0.1354 | 0.0541 | 0.0068 | 0.1546 | 0.2570 | 0.0431 | 0.0140 | 0.0022 | 0.3436 | 0.5892 | 0.0411 | 0.0155 | 0.0021 | 0.3687 | 0.6621 |
| EvolveGCN | 0.0503 | 0.0197 | 0.0025 | 0.1567 | 0.2733 | 0.1708 | 0.0593 | 0.0085 | 0.3550 | 0.5384 | 0.0593 | 0.0280 | 0.0030 | 0.3214 | 0.5567 |
| w/ $\mathscr{L}_{fair}$ | 0.0446 | 0.0169 | 0.0022 | 0.1555 | 0.2580 | 0.1699 | 0.0595 | 0.0085 | 0.3550 | 0.5192 | 0.0751 | 0.0273 | 0.0038 | 0.3136 | 0.5562 |
| FairGNN | 0.1798 | 0.0766 | 0.0090 | 0.1547 | 0.2526 | 0.1772 | 0.0719 | 0.0089 | 0.3868 | 0.6751 | 0.0695 | 0.0242 | 0.0034 | 0.5149 | 0.7098 |
| w/ $\mathscr{L}_{fair}$ | 0.1778 | 0.0853 | 0.0089 | 0.1544 | 0.2504 | 0.1802 | 0.0760 | 0.0090 | 0.3741 | 0.6571 | 0.0752 | 0.0264 | 0.0037 | 0.5095 | 0.7043 |
| FairVGNN | 0.1833 | 0.0851 | 0.0092 | 0.1537 | 0.2558 | 0.1354 | 0.1141 | 0.0068 | 0.3454 | 0.5576 | 0.0810 | 0.0330 | 0.0041 | 0.4342 | 0.6873 |
| w/ $\mathscr{L}_{fair}$ | 0.1843 | 0.0878 | 0.0092 | 0.1541 | 0.2521 | 0.1307 | 0.1063 | 0.0065 | 0.3356 | 0.5530 | 0.0827 | 0.0315 | 0.0041 | 0.4282 | 0.6301 |
| DegFairGNN | 0.1697 | 0.0754 | 0.0085 | 0.1588 | 0.3180 | 0.1272 | 0.0498 | 0.0064 | 0.3618 | 0.6759 | 0.0510 | 0.0194 | 0.0025 | 0.4665 | 0.6612 |
| w/ $\mathscr{L}_{fair}$ | 0.1798 | 0.0830 | 0.0090 | 0.1581 | 0.3076 | 0.1221 | 0.0468 | 0.0061 | 0.3598 | 0.6729 | 0.0538 | 0.0209 | 0.0027 | 0.4652 | 0.6594 |
| TailGNN | 0.1295 | 0.0566 | 0.0065 | 0.1549 | 0.2607 | 0.1832 | 0.0911 | 0.0092 | 0.3785 | 0.6537 | 0.0850 | 0.0344 | 0.0043 | 0.5040 | 0.6147 |
| w/ $\mathscr{L}_{fair}$ | 0.1386 | 0.0593 | 0.0069 | 0.1542 | 0.2588 | 0.1879 | 0.0941 | 0.0094 | 0.3649 | 0.5969 | 0.0782 | 0.0338 | 0.0039 | 0.5030 | 0.6097 |
| FairDGE | **0.2028** | **0.0899** | **0.0101** | **0.1489** | **0.2075** | **0.2647** | **0.1366** | **0.0132** | **0.3169** | **0.1729** | **0.1039** | **0.0390** | **0.0052** | **0.3107** | **0.3501** |
| Improvement | **10.04%** | **2.39%** | **9.78%** | **3.12%** | **17.13%** | **40.87%** | **19.72%** | **40.43%** | **2.31%** | **66.70%** | **22.24%** | **13.37%** | **20.93%** | **0.92%** | **37.06%** |

Table 3.4: FairDGE compares with 8 baselines on performance and fairness.

| | Performance(↑) | | | | | | | | | | Fairness(↓) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Variants | HR@{10,15,20} | | | NDCG@{10,15,20} | | | PREC@{10,15,20} | | | Dec. | rHR | rND | Inc. |
| FairDGE | **0.1256** | **0.1690** | **0.2025** | **0.0704** | **0.0823** | **0.0903** | **0.0126** | **0.0113** | **0.0101** | - | **0.1489** | **0.2075** | - |
| FairDGE¬$\mathscr{L}_{fair}$ | 0.0730 | 0.1068 | 0.1396 | 0.0391 | 0.0483 | 0.0562 | 0.0073 | 0.0071 | 0.0070 | **38.13%** | 0.1551 | 0.2699 | **17.12%** |
| FairDGE¬$\mathscr{L}_{class}$ | 0.0940 | 0.1325 | 0.1672 | 0.0490 | 0.0595 | 0.0679 | 0.0094 | 0.0088 | 0.0084 | **23.49%** | 0.1536 | 0.2605 | **14.35%** |
| FairDGE¬$\mathscr{L}_{contrast}$ | 0.0646 | 0.1024 | 0.1342 | 0.0334 | 0.0437 | 0.0513 | 0.0065 | 0.0068 | 0.0067 | **42.92%** | 0.1552 | 0.2728 | **17.85%** |
| FairDGE¬ Deg | 0.0681 | 0.0932 | 0.1199 | 0.0361 | 0.0429 | 0.0493 | 0.0068 | 0.0062 | 0.0060 | **45.02%** | 0.1524 | 0.2282 | **6.16%** |
| FairDGE¬ Seq | 0.0456 | 0.0681 | 0.0856 | 0.0254 | 0.0315 | 0.0357 | 0.0046 | 0.0045 | 0.0043 | **60.93%** | 0.1568 | 0.2792 | **19.93%** |

Table 3.5: Ablation study of variants on both performance and fairness. ¬ means the variant without the following module.

| | Performance(↑) | | | | | | | | | | Fairness(↓) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Variants | HR@{10,15,20} | | | NDCG@{10,15,20} | | | PREC@{10,15,20} | | | Dec. | rHR | rND | Inc. |
| FairDGE | **0.1819** | **0.2301** | **0.2647** | **0.1152** | **0.1283** | **0.1366** | **0.0182** | **0.0153** | **0.0132** | - | **0.3169** | **0.1729** | - |
| FairDGE¬$\mathscr{L}_{fair}$ | 0.1644 | 0.2122 | 0.2506 | 0.1050 | 0.1181 | 0.1273 | 0.0164 | 0.0141 | 0.0125 | **7.71%** | 0.3176 | 0.2126 | **11.59%** |
| FairDGE¬$\mathscr{L}_{class}$ | 0.1482 | 0.2011 | 0.2515 | 0.0797 | 0.0941 | 0.1063 | 0.0148 | 0.0134 | 0.0126 | **16.82%** | 0.3077 | 0.3365 | **50.40%** |
| FairDGE¬$\mathscr{L}_{contrast}$ | 0.1648 | 0.2208 | 0.2643 | 0.0908 | 0.1061 | 0.1165 | 0.0165 | 0.0147 | 0.0132 | **8.89%** | 0.3012 | 0.3881 | **59.76%** |
| FairDGE¬ Deg | 0.0243 | 0.0346 | 0.0482 | 0.0124 | 0.0152 | 0.0184 | 0.0024 | 0.0023 | 0.0024 | **85.66%** | 0.3190 | 0.4825 | **89.86%** |
| FairDGE¬ Seq | 0.1144 | 0.1772 | 0.2430 | 0.0501 | 0.0673 | 0.0831 | 0.0114 | 0.0118 | 0.0121 | **31.12%** | 0.5759 | 0.1990 | **48.41%** |

Table 3.6: Ablation study of variants on both performance and fairness on MovieLens dataset.

## 3.4.2 Performance and Fairness Comparison

To validate the effectiveness of our proposed FairDGE, we split the Amazon Books, MovieLens, and GoodReads datasets into 5, 15, and 15 snapshots for training and testing, respectively. We regard each user's last interacted item as the test data.

The experimental results are shown in Table 3.4. FairDGE performs best on all three datasets, outperforming baseline methods of 2.39% - 40.87% in recommendations, and achieves the smallest fairness scores, especially on the rND metric. The smaller the rHR and rND scores, the better the fairness. GCN and GAT always obtain the worst performance among these baselines because they are simple static graph neural networks and fail to embed the structural evolutions. Although EvolveGCN also gets worse performance on the Amazon Books dataset, it performs well on the MovieLens dataset because the Amazon Books dataset is too sparse for EvolveGCN to learn abundant evolution information via the recurrent architecture. Thanks to generative adversarial network structures, FairGNN and FairVGNN are the best baselines in both recommendation performance and fairness, but they still perform much worse than our FairDGE.

Another observation is that better fairness performance is obtained when equipping the baseline methods with our fairness loss $\mathscr{L}_{fair}$, especially rND. This demonstrates that our fairness loss effectively pushes the *T2H* vertices to get more exposure in the recommendation candidate list, especially to display in the high ranking. At the same time, their recommended performance only dropped slightly, and some even increased. This demonstrates that the dual debiasing approach makes FairDGE successfully overcome the second challenge.

### 3.4.3 Ablation Study of FairDGE

To study the contributions of each module to performance and fairness, I conducted the ablation study on the Amazon Books and Movielens datasets, as shown in Table 3.5 and Table 3.6, respectively. I will take the results on the Amazon Books dataset as an example to explain the contributions of each module. I compare the {HR, NDCG, PREC}@{10,15,20} for link prediction performance and compare rHR and rND scores to reveal fairness on different variants of FairDGE. The Dec. and Inc. columns denote the comparison with FairDGE about the average decreasing percentage performance scores and the average increasing percentage fairness scores, respectively. It should be noted that the increased fairness scores mean more discrepancy between the disadvantaged and advantaged vertices, leading to more unfairness issues.

The first row of variants is FairDGE, achieving the best performance and fairness scores. The next row is FairDGE without the fairness loss $\mathscr{L}_{fair}$. The performance and fairness decrease by 38.13% and increase by 17.12%, respectively. This indicates that both performance and fairness become worse when removing the fairness loss. This also validates that fairness loss breaks the effectiveness bottleneck and results in better performance. The third row is FairDGE without the intermedia training tasks of classifying the biased structural evolutions, which differentiates the *FaT*, *T2H*, and *SfH* vertex groups. This module influences the least performance of the downstream task, but it contributes much to fairness because this classification module aims to separate disadvantaged vertices from all vertices, which will contribute to the following fairness treatment. FairDGE$\neg\mathscr{L}_{contrast}$ denotes a variant without contrastive learning in the dual debiasing. The results indicate this module contributes almost the most to both downstream task performance and fairness because it assists in bringing closer the embeddings with identical biased structural evolution and penalizing those with different ones, further proving the effectiveness of contrastive learning on structural fairness [123]. The next variant is to remove the degree evolving trend learning module introduced in Chapter 3.3.2.1, which contributes to the second-best performance but slightly influences fairness. It is because the supervised classification of biased structural evolutions takes more effort in bias detection, but degree evolving trend learning is more associated with the dynamic graph embedding backbone for biased structural evolution learning, which limits its influence on performance and fairness. The last variant is FairDGE without dynamic structural evolution modeling, which is the GRU layers. This variant models the data statically and cannot differentiate the biased structural evolutions because the proposed fairness definition is based on dynamic scenarios, so the dynamic structural evolution modeling is very significant to both performance and fairness. To sum up, each module of FairDGE is indispensable and contributes to performance and fairness to varying degrees.

## 3.4.4 Hyperparameter Study

To study different hyperparameters' influence on performance and fairness, I conducted the hyperparameter studies on the Amazon Books dataset to test different numbers of snapshots and different model dimensions. For each hyperparameter study, I present the average scores after five runs for a fair comparison. The x-axis of Figure 3.2 denotes different hyperparameters. The left y-axis displays HR@10 and HR@20 to illustrate the performance trend, while the right y-axis exhibits the inverse function of the fairness scores (rHR and rND) for convenient visualization. This maintains a consistent interpretation for both sides of metrics, where a larger value indicates better effectiveness.

### 3.4.4.1 Different Number of Snapshots

To reveal the influence of the number of snapshots on performance and fairness, the left figure in Figure 3.2 shows the trend of performance and fairness. The greater the number of snapshots, the shorter the time period within each snapshot. There is a slight decrease in the performance trend as the number of snapshots increases. This is because the increased number of snapshots results in sparser snapshot graphs, as each snapshot contains less information. Consequently, the performance of each individual snapshot deteriorates. Additionally, the advanced ability of long temporal evolution

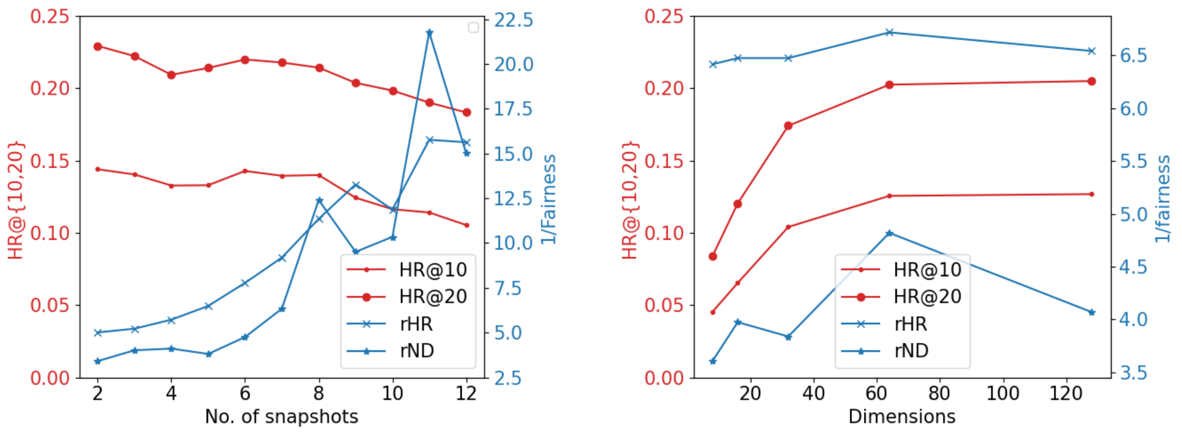


Figure 3.2: HR@{10,20} & {1/rHR, 1/rND} w.r.t different number of snapshots and model dimensions.

modeling is necessary to model longer snapshot evolution. However, the GRU model in dynamic graph embedding backbone restricts its ability to learn this evolution, resulting in lower performance. On the other hand, from a fairness perspective, a higher number of snapshots allows for a more refined evolution of the data. This enhanced granularity aids in accurately training the degree evolving trend model to capture change across the evolution snapshots. Consequently, this benefits FairDGE in effectively identifying different biased vertex groups, contributing to improved fairness. In addition, the trends of performance and fairness reverse as the number of snapshots increases, which aligns with the widely recognized notion that performance and fairness often conflict with each other.

#### 3.4.4.2 Different Dimensions

The right figure in Figure 3.2 illustrates the variations in performance and fairness as the FairDGE dimension varies from 8 to 128. The performance metrics (HR@{10,20}) exhibit notably low values at dimension 8 but experience a steep increase after dimension 32. Subsequently, the performance metrics reach their peak at dimension 64. Regarding the fairness scores, it initially performed poorly at the lowest dimension, surpassing other dimensions at dimension 64. However, at dimension 128, the fairness scores decrease and become the second-best due to over-training. As a result, I select a default dimension setting of 64 for all the experiments.

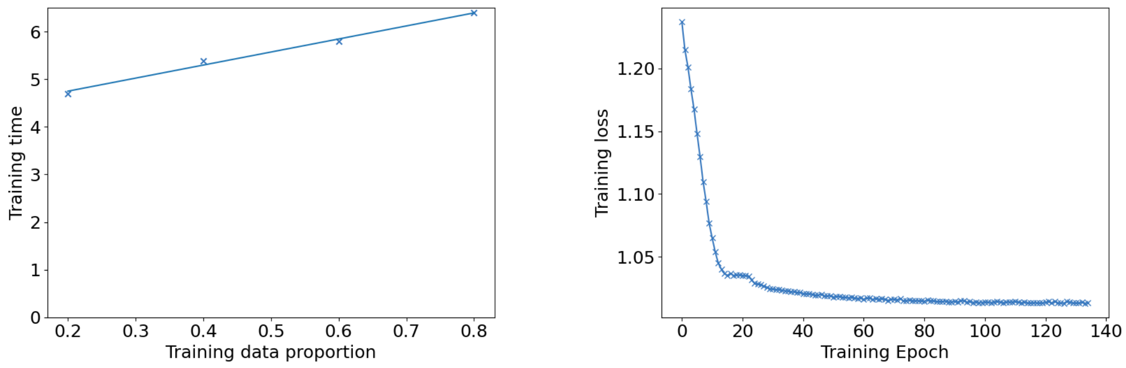


Figure 3.3: Training efficiency analysis and convergence.

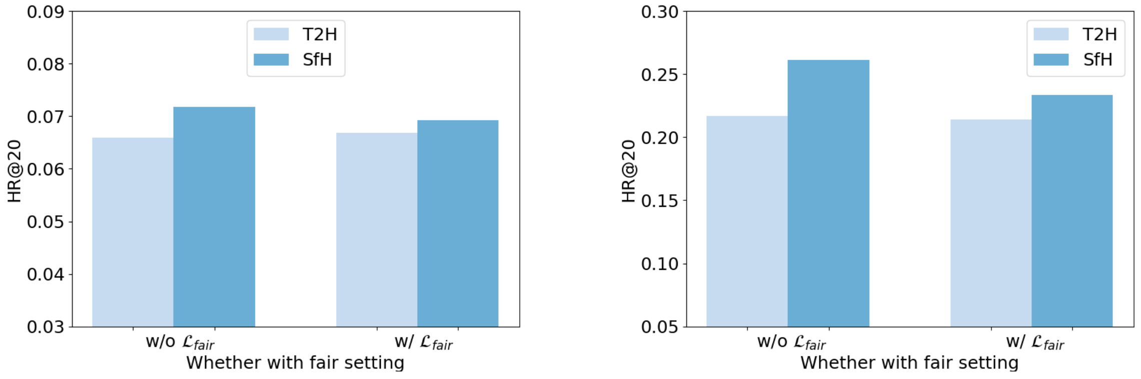
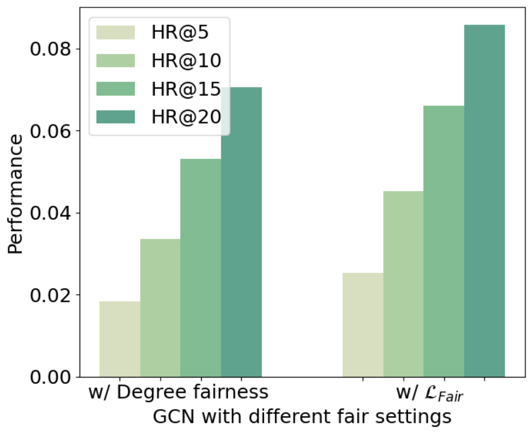


Figure 3.4: Effectiveness of fairness loss $\mathscr{L}_{fair}$.

### 3.4.5 Training Efficiency and Convergence

I conduct an empirical test on the training efficiency of FairDGE by randomly selecting different percentages of the training data {20%, 40%, 60%, 80%} from the Amazon Books dataset. The growth of the training time at each epoch is shown in the left figure in Figure 3.3. As the volume of training data increases, the training time increases linearly. This demonstrates the efficiency of FairDGE when dealing with extensive training data. Additionally, I provide the convergence curve of the training process on the entire Amazon Books dataset in the right figure in Figure 3.3. The x-axis represents the number of training epochs, while the y-axis represents the training loss. The model converges after approximately 60 epochs.



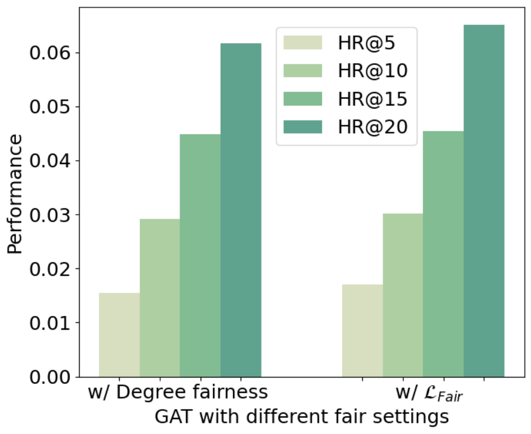


Figure 3.5: Effectiveness of debiasing structural evolutions.

### 3.4.6 Effectiveness Analysis of Fairness

Following, I first discuss the effectiveness of fairness loss $\mathscr{L}_{fair}$ on the T2H and SfH vertex groups and then reveal the superiority of capturing structural evolution for debiasing against the conventional high- and low-degree-based fairness.

#### 3.4.6.1 Effectiveness of Fairness Loss $\mathscr{L}_{fair}$ on The T2H and SfH Vertex Groups.

To demonstrate the effectiveness of the fairness loss $\mathscr{L}_{fair}$ on the disadvantage group (*T2*) and advantage group (*SfH*), I analyze the embedding performance in downstream tasks. The closer the performance of T2H and SfH vertex groups, the better the fairness. Specifically, I compare the HR@20 scores of all the biased vertex groups with and without the fairness loss $\mathscr{L}_{fair}$ on GCN and FairDGE. The results are shown in Figure 3.4, where the x-axis denotes whether the fairness loss is used and the y-axis represents the HR@20 scores for performance comparison.

In each result set, the two bars represent the HR@20 scores of *T2H* and *SfH*, respectively. I observe that, without $\mathscr{L}_{fair}$, GCN exhibits a significant performance gap between *T2H* and *SfH*, and the performance is much close after equipping $\mathscr{L}_{fair}$. A similar phenomenon is observed in FairDGE. This indicates that the fairness loss $\mathscr{L}_{fair}$ effectively narrows the performance gap between the *T2H* and *SfH* vertex groups, thereby improving fairness.

| | rHR (↓) | rND (↓) |
|---|---|---|
| GCN w/o Fairness | 0.1559 | 0.2627 |
| w/ Degree Fairness | 0.1553 (0.4% ↓) | 0.2578 (2% ↓) |
| w/ $\mathscr{L}_{fair}$ | **0.1547 (0.8% ↓)** | **0.2551 (3% ↓)** |
| GAT w/o Fairness | 0.1569 | 0.2765 |
| w/ Degree Fairness | 0.1557 (0.8% ↓) | 0.2711 (2% ↓) |
| w/ $\mathscr{L}_{fair}$ | **0.1550 (1.2% ↓)** | **0.2649 (4% ↓)** |

Table 3.7: Fairness of debiasing structural evolutions.

#### 3.4.6.2 Effectiveness of Capturing Structural Evolution for Debiasing against Degree Fairness.

Degree fairness prevents the performance disparity of head- and tail-degree vertex groups. To further validate that capturing structural evolution for debiasing is better than static degree fairness, I divide vertices into head- and tail-degree vertex groups to train GCN and GAT with fairness loss $\mathscr{L}_{fair}$ and compare the performance with GCN and GAT trained on the T2H and SfH vertex groups. The results in Figure 3.5 and Table 3.7 validate that capturing structural evolution to debias dynamic graph embedding is capable of achieving fairness without sacrificing and even improving embedding performance, which degree fairness cannot. Furthermore, these results also demonstrate that the debiasing method can be used as a general fairness plug-in, enabling dynamic graph embedding algorithms to achieve structural fairness.

## 3.5 Summary

This chapter investigates the dynamic fairness problem, and proposes a novel structurally Fair Dynamic Graph Embedding algorithm, namely FairDGE, which first learns trend-aware structural evolutions and then encodes structurally fair embeddings by a novel dual debiasing approach. I have conducted extensive experiments and achieved simultaneous performance and fairness improvements. This work effectively models the imbalanced distribution of dynamic graphs and establishes a solid foundation for graph modeling aspects of graph-based explainable recommendation algorithms.

CHAPTER 4

# Temporal Meta-path Guided Explainable Recommendation leveraging Reinforcement Learning

For recommendation modeling, this Chapter studies the sequential modeling of the user's historical behavior, which accurately captures the evolution of the user's preference. Additionally, I also propose a reinforcement learning-based path exploration for meaningful explainable paths mining, which also benefits the user's preference modeling. This model contributes to the user's historical behavior modeling for both accurate performance of recommendations and meaningful explanation generation.

## 4.1 Overview

Reasoning with paths over user-item associated Knowledge Graphs (KGs) has been becoming a popular means for explainable recommendations [149, 159, 184]. The path-based recommendation systems have successfully achieved promising recommendation performance, as well as appealing explanations via searching the connectivity information between users and items across KGs. One category of existing works on path-based explainable recommendations seeks auxiliary meta-paths (pre-defined higher-order relational compositions between various types of entities in KGs) as similarity measures and evidence for possible explanations between users and items.

However, most existing path-based methods for explainable recommendation simply

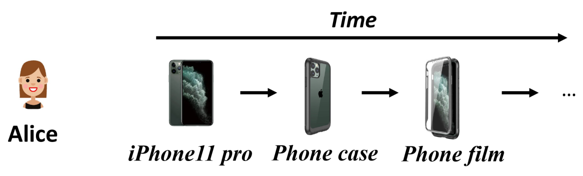


Figure 4.1: Intuitive sequential modeling.

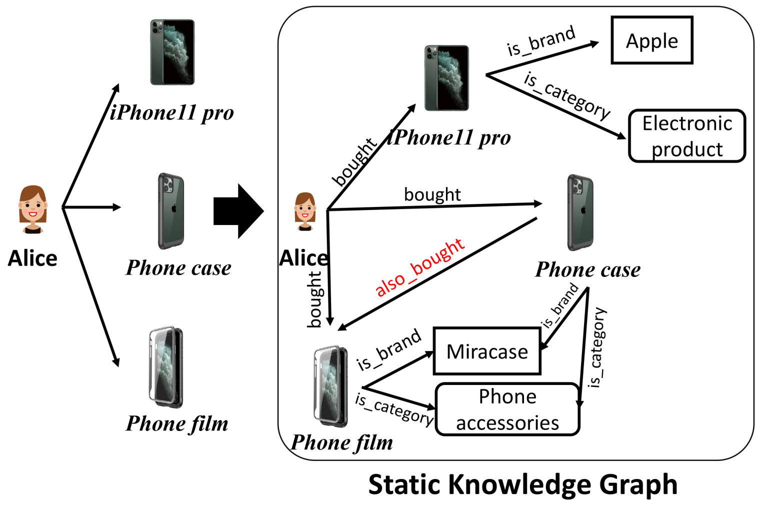


Figure 4.2: Existing static knowledge-graph based modeling.

treat the underlying KGs as static graphs, ignoring the dynamic and evolving nature of user-item interactions in real-world recommendation scenarios. The dynamics and evolutions of users' interactions with items play a pivotal role in both recommendation precision and explanations for a user's real intent. Taking the scenario in Figure 4.1 as an example, Alice purchased a phone case and a phone film after a recent buy of a new phone. If ignore the sequential information between each purchase (in Figure 4.1) and treat the whole information as a static graph (in Figure 4.2), the system probably gives an explanation of buying the phone case is because a similar customer also bought this phone case by exploring co-purchasing relationships. Whilst the explanation, in this case, might be valid and the underlying reasoning mechanism (collaborative filtering signal) can be used for recommendation, it is still sub-optimal as a more appealing motivation for buying a phone case is due to the recent purchase of a new phone. For this reason, in this example, it is important for a system to be capable of considering temporal and sequential user-item interactions and disentangling the importance of various reasons when generating possible explanations.

Although some prior works have considered some extent of the sequential information for knowledge-aware path-based explainable recommendation problems, they still fail to explicitly model the dynamics of users' activities. To allow effective reasoning on paths to infer the underlying rationale of a user-item interaction, the method proposed in [149] takes advantage of path connectivity and leverages the sequential dependencies of entities and sophisticated relations of a path connecting a user-item pair. Nevertheless, the methods only consider the sequential information within specific paths and did not consider the importance of the user's historical sequences in reflecting the user's dynamic interactions with items. To improve, the KARN model [192] fuses the user's clicked history sequence and path connectivity between users and items in KGs for recommendation. However, the method models the user's sequential behaviors and user-item interactions separately and in a coarse-grained manner (treating a user's click history as a whole), which may restrict the expressiveness of users' temporal dynamics on recommendation explainability.

In light of this, in this Chapter, I challenge the problem of exploring users' temporal sequential dynamics in the context of path-based knowledge-aware recommendation. Different from existing works that either only consider the sequential information within a path or treat the user's sequential interactions as a whole and separately, I aim to 1) explicitly model and integrate the dynamic user-item interactions over time into the path-based knowledge-aware recommendation, and 2) leverage the captured dynamics of user-item interactions to improve the performance and explainability of the recommendation.

It is worth noting that modeling users' temporal sequential behavior with path-based knowledge-aware explainable recommendation is a non-trial and challenging task. First, path-based knowledge-aware recommender systems are built upon the well-constructed KGs, and their expected accuracy and explainability are highly related to the underlying KGs and distilled paths. If temporal information between users and items is considered, the original underlying static KGs will be cast into multiple snapshots w.r.t. the timestamps. Compared with the static KGs that consist of all users' full timelines, each snapshot at a certain timestamp only has partial observations of the global knowledge, which will result in inferior recommendation performance. To deal with the issue, I devise a general framework for temporal knowledge aware reinforcement path-based explainable recommender systems, namely Temporal Meta-path Guided Explainable Recommendation leveraging Reinforcement Learning (TMER-RL). TMER-RL provides a solution that can explicitly depict users' sequential behavior while being able to be aware of global knowledge of the entire underlying KGs. Specifically, to model the temporal dynamics between users and items, TMER-RL naturally models the task as a sequential recommendation problem and takes as input users and their corresponding sequential purchase history. To learn the sequential dependencies between consecutive items purchased by a user, TMER-RL novelly explores item-item paths between consecutive items and embeds the paths as context with elaborated designed attention mechanisms to model the dynamics between user and items. Compared to existing path-based explainable recommendation systems that only consider user-item paths as evidence and support for the recommendation decision, TMER-RL contributes another creamy layer the top of existing works, which makes use of the powerful expressiveness of temporal information between users and items.

In addition, when taking paths into consideration for the above purpose, another challenge lies in the existence of a large number of possible paths. It is time-consuming for a model to select several paths that are meaningful, expressive, and have positive impacts on both recommendation performance and explainability. To this end, inspired by prior work [58, 193], my previous work TMER [18] leverages the concept of meta-path [127] and explores diverse meta-path schemas to characterize the context of dynamic interactions between users and items. However, pre-defined meta-paths and random path instances selection from all generated ones involves human and random factors in the following recommendation module. Moreover, the diversity of the explanations is also restricted by the pre-defined meta-path schemas. To solve the above shortcomings, some work adopts reinforcement learning to retrieve diverse paths to improve recommendation. For example, PGPR [159] uses a novel policy-gradient approach to navigate users' potential interests and form explainable paths, and TPGR [17] regards the candidate nodes as a balanced clustering tree and proposes a tree-structured policy gradient approach to guide the path exploration. However, the above path-finding methods usually suffer from poor convergence because of a lack of supervision and using the brute-force method. The intuitive rationale is that supervision could lead the training process to quickly approach the ground truth and explore a reliable strategy to find paths. Afterwards, ADAC [184] introduces a weakly supervised reinforcement framework to first derive sets of paths according to pre-defined meta-paths and then use reinforcement learning to explore interpretable paths and predict items, whereas it still inevitably adopts the pre-defined meta-path to help fast convergence. To sum up, the above reinforcement learning-based path exploration methods need to not only explore explainable paths, but also predict items for recommendation, so they naturally lack supervision (pre-defined target items) to guide the exploration of the path. However, in the method of this Chapter, the goal of the reinforcement learning method is only to mine diverse paths between the given nodes (users and items), which means the reinforcement path exploration is supervised by the target nodes. Hence, the reinforcement learning path exploration could better converge thanks to the target nodes. I also design a useful score function for paths to assist paths mining.

With respect to the powerful transformer model for sequential modeling, I also elaborate on item-item path attention and user-item path attention units to learn combinational features of multiple paths to further characterize users' temporal purchasing motivations and their general shopping tastes, respectively. The rationale for developing such path-based attention units is that a user's motivation toward buying a certain product is complex and consists of multiple factors. For example, when buying a new phone, a customer may consider several factors including a phone's intrinsic features such as functionality, display, camera, etc., as well as other external factors such as the choices of their close friends, and their previous purchase of certain related electronics using the same operating system. With the help of the designed path-based attention units, the proposed TMER-RL framework is able to learn different weights for various possible paths which will be then used as explanations for recommendations (I show the effectiveness of using the proposed path-based attention units for the explainable recommendation in the experiments by running case studies).

The main contributions of this Chapter are summarized as follows:

- I analyze the meta-path instances' context relation learning ability for the recommendation, and differentiate the meta-path instances into user-item meta-path instances and item-item meta-path instances because of their different potential meaning. I introduce, to the best of my knowledge, the first study to model users' temporal sequential behavior with the item-item path-based knowledge-aware explainable recommendation.

- I propose a novel reinforcement path exploration for diversity path mining. Compared to the pre-defined meta-path derivation, the method could explore more diverse explainable paths, and therefore, achieve a better recommendation performance than the random selection from meta-path derived paths.

- I propose Temporal Meta-path Guided Explainable Recommendation leveraging Reinforcement Learning (TMER-RL), which considers users' dynamic behaviors on top of the global knowledge graph for sequential-aware recommendation and explores both user-item and item-item meta-path paths with well-designed reinforcement framework and attention mechanisms for the explainable recommendation.

- Experiments on four real-world sequential recommendation datasets demonstrate that TMER-RL outperforms several state-of-the-art baselines in recommendation performance and generating path explanations. Especially, I discuss the reinforcement learning-based paths exploration and random selection approaches through the case study.

## 4.2 Preliminaries

Following, I give some essential definitions and define the problem.

**Definition 4.** ***Heterogeneous Information Network.*** *In Heterogeneous information network (HIN) $G = (\mathcal{V}, E)$, each edge $e \in E$ represents a particular relation $r \in R$, each entity $v \in \mathcal{V}$ belongs a particular type $\chi \in X$, where $|R| > 1, |X| > 1$.*

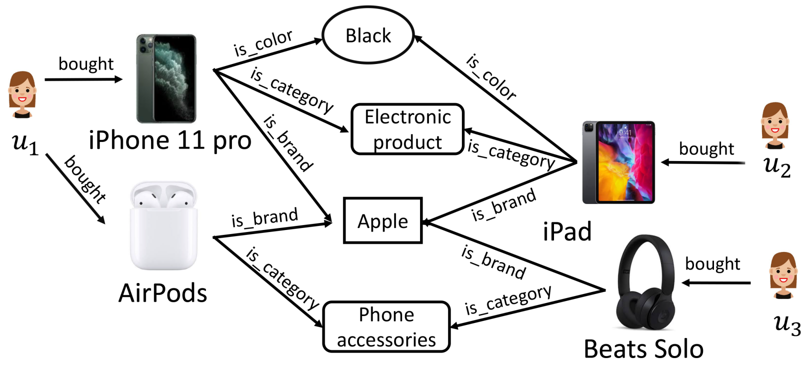


Figure 4.3: An example of a heterogeneous information network: product recommendation network.

**Definition 5.** ***Meta-path.*** *A meta-path [127] P in a network from entity $v_0$ to $v_k$, is denoted as $v_0 \xrightarrow{r_0} v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \ldots \xrightarrow{r_{k-1}} v_k$, where composite relation from $v_0$ to $v_k$ is $r = r_0 \circ r_1 \circ r_2 \circ \ldots \circ r_{k-1}$. $\circ$ represents the composition operator on relations.*

**Example 1.** *As shown in Figure 4.3, the product recommendation network $G_P = (\mathcal{V}, E)$ contains 5 types of nodes, including items, users, categories, colors and brands. Edges between users and items denote the buy relation, edges between items and brands denote the is brand of relation, and etc. There are many meta-paths; one of the meta-paths* *is $UIBI$, which means user $\rightarrow$ item $\rightarrow$ brand $\rightarrow$ item. In this Chapter, I explore the meta-path instances instead of meta-paths. A meta-path instance is a specific path, like $u_1 \rightarrow AirPods \rightarrow Apple \rightarrow BeatsSolo$ is a meta-path instance of meta-path $UIBI$.*

**Problem 1.** ***Sequential knowledge-aware explainable recommendation.*** *For a user $u_i \in U$, given the item set $I$, the user transaction sequence $S_u$ and the item associated network $G$, the target of knowledge-aware explainable recommendation is to predict top $k$ items that $u_i$ will interact with, as well as the possible reasoning of recommended items.*

## 4.3 Methodology

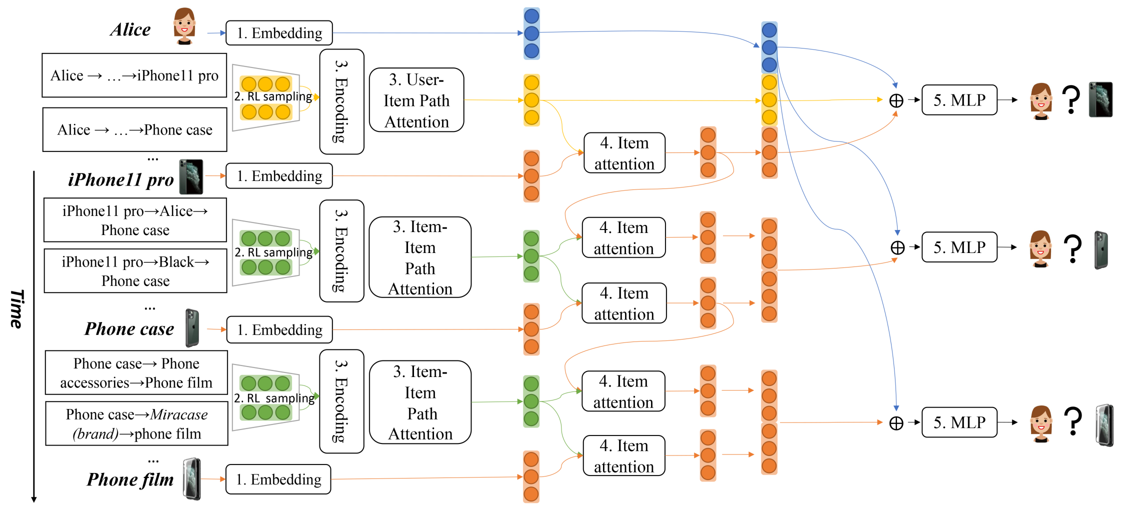

Alice
1. Embedding
Alice → …→iPhone11 pro
Alice → …→Phone case
2. RL sampling
3. Encoding
3. User-Item Path Attention
4. Item attention
5. MLP
iPhone11 pro
iPhone11 pro→Alice→Phone case
iPhone11 pro→Black→Phone case
3. Item-Item Path Attention
Time
Phone case
Phone case→ Phone accessories→Phone film
Phone case→Miracase (brand)→phone film
Phone film
?

Figure 4.4: It is the architecture of Temporal Meta-path Guided Explainable Recommendation leveraging Reinforcement Learning (TMER-RL). Here is an example: User *Alice* bought *iPhone 11 pro*, *Phone case* and *Phone film* in sequential order and the training process for *Alice*. The index in each network block is the step order of the model.

Following, I introduce the proposed model Temporal Meta-path Guided Explainable Recommendation leveraging Reinforcement Learning (TMER-RL). In the remaining of the Chapter, I use the notation summarized in Table 4.1 to refer to the variables and parameters used throughout the Chapter.

### 4.3.1 Overview of TMER-RL Architecture

The overall architecture of the proposed TMER-RL model is shown in Figure 4.4. It mainly consists of five components. First, to initialize users and items, I use DeepWalk

| Symbol | Description |
|---|---|
| $G = (\mathcal{V}, E)$ | Heterogeneous information network (HIN) |
| $R$ | Type of edges (relations) |
| $T$ | Type of nodes (entities) |
| $\mathcal{U}$ | User entity set |
| $\mathcal{I}$ | Item entity set |
| $\mathcal{B}$ | Brand entity set |
| $\mathcal{C}$ | Category entity set |
| $\mathcal{S}$ | State set |
| $\mathcal{A}$ | Action set |
| $p \in P$ | A meta-path in an HIN |
| $u \in U$ | User $u$ |
| $i \in I$ | Item $i$ |
| $W$ | Weight matrix |
| $b$ | Bias vector |
| $\phi \in \Phi$ | Path instance in a HIN |
| $\mathcal{N}$ | Neighbour function |
| $h$ | Embedding |
| $\alpha$ | Attention mechanism parameter |
| $\hat{r}$ | User-item rating |

Table 4.1: Description of major notations used in this Chapter.

[108] to pre-train user and item entities. Secondly, I utilize reinforcement learning with a Markov Decision Process (MDP) environment to explore useful and meaningful sequential (temporal) and non-sequential paths to improve the recommendation performance and personalization. In this step, I obtain item-item instance paths between consecutive items using a reinforcement learning framework. In the third step, after embedding instances, I use multi-head attention to learn the weights of instances as the weights of reasoning paths for the specific user. Next, for the item updating, I employ a two-layer attention to make items contain reasoning information. This step also models the users' sequential purchased information, feeding the previous item's feature to the next one. Finally, I feed item embeddings, user embeddings, and instances to the recommendation network to make recommendations. The specific steps are elaborated on as follows.

## 4.3.2 Initialize User and Item Representations

I firstly learn latent representations for involved users and items by treating random truncated walks in a user-item bipartite network as an equivalent of sentences in DeepWalk [108], which optimizes the co-occurrence probability among the entities in a walk by using skip-gram based on word2vec [92]. Other recent advanced graph-based embedding initialization methods can also be applied, like GraphSage [45], GAT [137], and so on. These recent works can usually outperform DeepWalk. However, through an extensive comparison of these methods, DeepWalk is the best choice. The possible reason behind this is that DeepWalk pays more attention to the embedding of nodes in a path, while GCNs, such as GraphSage, GAT, etc., learn the embedding of each entity with the aggregated feature information from its local neighborhood, which means that these approaches pay more attention to local relationships. However, if only considering the recommendation task itself, although the local relationships are significant, the global higher-order relationships learned by walks on a graph also play a notable role. For example, the transitivity of co-purchasing relationships between friends and the substitute relationship of items can be discovered by modeling higher-order relations in a bipartite user-item graph.

### 4.3.3 Incorporating Meta-path based Context

In recommendation tasks, external features related to users and items, such as product attributes, user social networks, and user demographic information are usually considered as additional auxiliary information to complement traditional collaborative filtering methods. However, how to utilize the heterogeneous additional information efficiently is an open problem. Some prior works [65, 89, 174] attempt to consider social relations as the user-side information to boost the recommendation performance. To seek help from injecting more complex additional information, recent works [46, 173] introduce meta-paths into recommendation methods to describe relational compositions between various types of entities in heterogeneous information networks. In [173], the authors propose to diffuse user preferences along different meta-paths in information networks to generate latent features of users and items. Related work [46] firstly extracts different-aspect features with meta-paths from a HIN, and then fuses aspect-level latent factors to the recommendation systems. However, these methods largely rely on the latent factors obtained from constructed meta-path based similarity matrices, which are too general and can only reflect mutual interaction between different types of entities in a graph but cannot capture the specific information along with particular path instances. Therefore, inspired by existing work [58], my prior work TMER and the proposed model TMER-RL explore improving both recommendation performance and explainability by modeling more specific meta-path instances.

Different from existing works, I differentiate meta-path instances into two different categories based on the involved entities in a recommendation scenario (i.e., user-item and item-item meta-path instances). Through modeling these path instances, I learn a more detailed meta-path based context to further characterize the motivations, reasons as well as leading factors between each pair of user-item interactions. While previous works such as [58] mainly focus on modeling meta-path instances between a user and an item, the work proposed in this Chapter highlights the item-item meta-path instances, which I think are beneficial in multiple aspects of sequential explainable recommender systems. Firstly, only considering user-item paths is restrictive for recommendation explainability as user-item paths only represent a user's general shopping interests. In comparison, item-item paths are more expressive and can reflect diverse reasons by exploring higher-order relations among items, such as complementary products (e.g., phone and a phone case), substitutable items of known items (e.g., iPhone - phone film - Huawei Phone), co-purchased products with other people, etc. In addition, item-item paths sometimes also serve as sequential modeling signals that naturally capture the temporal dependencies between each consecutive item purchased by users, which will be of great impact for sequential explainable recommendations.

Despite the powerful expressiveness of meta-paths in exploring HIN-based knowledge-aware recommendation, it is still challenging mainly because the number of meta-paths is too large to handle (i.e., the amount of edges is cubic to the number of entities). Taking the electric product recommendation scenario for an example, for $IUIBI$ meta-path schema, if I fix the starting node ($iPhone11\ pro$), there are many instances: $iPhone11\ pro \rightarrow Alice \rightarrow iPad \rightarrow Apple \rightarrow AirPods$, $iPhone11\ pro \rightarrow Amy \rightarrow Phone\ case \rightarrow Miracase \rightarrow Phone\ film$, and so on. Therefore, it is necessary to explore useful path instances while limiting the total amount to simplify later calculations. The path instances exploration is introduced in Chapter 4.3.4.

### 4.3.4 Reinforcement Learning for Paths Exploration

My previous work [18] pre-defines use-item and item-item meta-paths on the recommendation knowledge graph, and randomly selects meta-path instances from all existing ones. The hand-crafted meta-paths not only need human efforts but also are difficult to be determined when dealing with large recommendation knowledge graphs. Therefore, I propose a reinforcement learning module to explore potential *useful* meta-path instances on the recommendation knowledge graph. The definition of *usefulness* in my work is to have a contribution to the recommendation module learning. The process of meta-path instances exploration can be defined as a Markov Decision Process (MDP)[121], and I use reinforcement learning to train the exploration policy.

- *State.* It is the status of each step and the state set is defined as $\mathscr{S}$. At step $t$, the state $\mathscr{S}_t$ is $(e_t, his_t)$, where $e_t \in (\mathscr{U} \cup \mathscr{I} \cup \mathscr{B} \cup \mathscr{C})$ is the current visiting entity and $his_t = \{e_0, e_1, ..., e_t\}$ is the history visited entities including the current one. Inspired by [28, 114, 159, 163], I add self-loop and inverse edges on knowledge graph $G$ to facilitate graph traversal.

- *Action.* The action set $\mathscr{A}_t$ at step $t$ means all candidate entities to go. I define $\mathscr{A}_t = \mathscr{N}(e_t) \cup e_t$, which means the candidate entities are the neighbors of the current visiting entity $e_t$ and itself. Moreover, I allocate each action a weight to present the probability of choosing each action. The score function of each action at step $t$ is as follows.

$$s_{e_i} = softmax(\frac{h_{e_t} \cdot h_{e_i}^T}{||h_{e_t}||||h_{e_i}||}),\ where\ e_i \in \mathscr{A}_t \tag{4.1}$$

  where $h_{e_t}$ and $h_{e_i}$ mean the presentation of the current visiting entity and one of the actions, respectively. I believe the cosine function could indicate the similarity of two entities and it is more likely to choose similar entities to form a path on the knowledge graph. However, on the knowledge graph, some entities have dense structures, leading to large action sets. Thus, pruning unimportant candidate entities is necessary to improve efficiency. I rank each action's score descendingly and obtain the top $k$ as the new action set.

- *Transition.* Given a current state $\mathscr{S}_t = (e_t, his_t)$ and an action $e_i \in \mathscr{A}_t$, the transition probability to the next state $\mathscr{S}_{t+1} = (e_{t+1}, his_{t+1})$ is

$$p(\mathscr{S}_{t+1} | \mathscr{S}_t, e_i) = 1 \tag{4.2}$$

- *Reward.* The target of reinforcement learning is to find user-item meta-path instances and item-item meta-path instances, so the target entity is known for each user. Therefore, if I get to the target entity before the pre-defined maximum step $T$, the reward is 1, otherwise 0, mathematically,

$$\mathscr{R} = \begin{cases} 1, & if\ get\ to\ the\ target\ entity\ before\ step\ \mathrm{T}, \\ 0, & otherwise \end{cases} \tag{4.3}$$

- *Optimization.* The whole reinforcement learning module maximizes the final reward to learn policy $\pi$ to find useful user-item and item-item meta-path instances, mathematically,

$$J(\theta) = \mathbb{E}_{e_0 \in (\mathscr{U} \cup \mathscr{I})}(\mathbb{E}_{\pi}(\mathscr{R})) \tag{4.4}$$

where $e_0$ is the initial entity. I use REINFORCE [155] algorithm to train the objective function.

According to the above MDP framework, I can get candidate user-item and item-item meta-path instances. Taking user-item meta-path instances from $u_n$ to $i_m$ for an example, there should be $p$ candidate paths. I use a score function to calculate each path's score, mathematically,

$$c_{u_n \to i_m} = \frac{\sum_{t=1}^{T} s_{e_t}}{T} \tag{4.5}$$

where $s_{e_t}$ is each step's action score. The rationale for choosing the average step's score as the path's score is that I do not want the length of the path to influence the path's score. I rank the score of paths descendingly and get the top $q$ candidate paths. The pseudo code of exploring paths from user $u_i$ to the item $i_m$ is shown as Algorithm 2.

**Algorithm 2** Reinforcement learning for paths exploration.

```
Input: $S_t = (e_t, his_t)$ is the state at step $t$, $R_t$ is the reward associated with the $t$-th
    transition, $T$ is the maximum step, $C$ is the array storing the candidate paths' scores,
    $u_n$ is the start node, $i_m$ is the end node
Output: Paths $P$ and Scores $C$
 1: for each $t \in [1, T]$ do
 2:   if $e_t \neq i_m$ then
 3:     calculate the score $s_{e_i}$ of the candidate nodes according to Eq. 4.1, where $e_i \in A_t$
 4:     obtain top $k$ candidate nodes as pruned action set $A_t$ according to candidate
        nodes scores $s$
 5:   else
 6:     Break
 7:   end if
 8: end for
 9: calculate $R$ according to Eq. 4.3
10: if $R == 1$ then
11:   calculate path's score $c[u_n \to i_m]$ according to Eq. 4.5
12: end if
13: return $P, C$
```

### 4.3.5 Parameterizing Combinational Features of Meta-paths as Recommendation Context

#### 4.3.5.1 Learning Combinational Path-based Features with Self-Attention

After obtaining candidate user-item meta-path instances and item-item meta-path instances, I first regard paths as sentences, and nodes as tokens in sentences, using Word2Vec [92] method and $Mean(\cdot)$ operations to learn path embedding. Then, I employ multi-head self-attention units to learn the meta-path based context (the User-Item and Item-Item Path Attention modules shown in Figure 4.4). The rationale for deploying such self-attention units is that after reinforcement learning paths exploration, there are still multiple paths between each pair of item-item (or user-item) representing particular distinct reasons (reasoning paths); and I observe that the reasons for buying two consecutive items are not simply unique; rather, the reasons are more complex and likely a mixture of multiple different reasons. For example, the reasons for a customer to buy a phone case right after his/her previous purchase of a new phone are probably a mixture of 1) his/her friends who own a similar phone and bought this particular phone case, 2) the phone case is the most popular match for the purchased new phone, and 3) the color of the phone case matches the customer's preference. The potential reasons can be more than the listed, and again they can be represented by using various meta-paths.

Based on this observation, self-attentive properties of the Transformer model [136], I aim to learn the combinational features from multiple path instances to better characterize the complex reasons between each connected pair of entities in the KG.

$$\begin{aligned} &Attention(Q_\phi, K_\phi, V_\phi) = f\left(\frac{Q_\phi K_\phi^T}{\sqrt{d_k}}\right)V_\phi, \\ &MultiHead(Q_\phi, K_\phi, V_\phi) = Concat(head_1, ..., head_m)W^O, \end{aligned} \tag{4.6}$$

where $head_i$ is $Attention(W_i^Q Q_\phi, W_i^K K_\phi, W_i^V V_\phi)$. $f(.)$ is a softmax function. Query $Q$, key $K$ and value $V$ are self-attention variables associated with path $\phi$, and $W$ is the weight. $d_k$ is the dimensionality (here $d_k = 100$). $Concat(.)$ is the concatenation operation.

#### 4.3.5.2 Modeling Temporal Dependencies with Item-item Meta-path Instances

To learn the temporal dynamics of each user, the proposed TMER-RL framework artfully resorts to the above-mentioned item-item meta-path instances together with the well-designed architecture to capture the sequential dependencies between two consecutive items. Compared with most existing works on sequential recommendation [149, 164, 192] that utilize recurrent neural networks to encode the temporal effects between items in a user's interacted sequence, the proposed TMER-RL bypasses the de-facto default deployment of RNNs or LSTMs that sometimes make the model even heavier. Specifically, the proposed framework novelly models the temporal dependencies between two items by capturing 1) the information passed from the previous item through an item-attention unit, and 2) item-item connectivity through a specific candidate path instance. Notably, the information passed from the previous item is an attentive aggregation of previous item-item connectivity information. For example, in Figure 4.4, whether Alice will buy the phone film is modeled by considering 1) the information passed from the phone case (which includes the connectivity between the iphone11 pro and the phone case), and 2) the paths between the phone case to the phone film. As a result, the long-range and short-term "memory" in a sequence can be captured, and the extent of the goodness of the long and short-term memory can be influenced by the length of the modeled sequence in different scenarios with different datasets.

#### 4.3.5.3 Updating Item Representations

After updating representations of user-item meta-path instances and item-item meta-path instances according to a multi-head self-attention mechanism, I employ a different kind of attention mechanism to update item representations. It is obvious that the current item is mostly related to the last one, which means it is better to add the last item's information to the current one to contain temporal information. Besides, the current item is also related to the instances that arrived at this item. Therefore, I perform a two-layer attention mechanism to update item representations. Mathematically,

$$h_i^{(1)} = g(W_{i-1}h_{i-1} + W_{\phi_{i-1\to i}}h_{\phi_{i-1\to i}} + b_i^{(1)}) \odot h_{i-1} \tag{4.7}$$

$$h_i^{(2)} = g(W_i h_i + W^{(2)}_{\phi_{i-1\to i}}h_{\phi_{i-1\to i}} + b_i^{(2)}) \odot h_i \tag{4.8}$$

where $h_i^{(1)}$ and $h_i^{(2)}$ mean the first and second layer output of the item attention module, respectively. $h_{i-1}$ is the last item's latent representation. $\phi_{i-1\to i}$ is the instance from the $(i-1)^{th}$ item to the $i^{th}$ item. $W$ and $b$ with different superscripts denote different variables' weights and bias. $g(.)$ is ReLU function. However, there is still a problem with the calculation of the first item, because there is no item before it. To solve this problem, I involve the user-item instance from user $u$ to the first item in the update of the first item, as Eq. 4.9 shows. Actually, the instance from $u$ to the first item is really important in the recommendation, since it is the first item that the user has bought and most of the other bought items have a sequential relation with the first item to some extent. That is why I embed it into the first item.

$$h_{i=1} = g(W_i h_i + W_{\phi_{u\to i}} h_{\phi_{u\to i}} + b_i) \odot h_i \tag{4.9}$$

where $\phi_{u\to i}$ represents the path from user $u$ to the first item.

### 4.3.6 The Complete Recommendation Model

Finally, I concatenate user, item and instances information (calculated in 4.3.5.2) to a vector according to Eq. 4.10, and get user-item prediction scores through Multilayer Perceptron (MLP) with explainability instances.

$$h_{u,i} = [h_u; h_i^{(1)}; h_i^{(2)}] \tag{4.10}$$

where [;] means vector concatenation. Here $h_{u,i}$ denotes the explicit mutual vector of the user, item, and implicit mutual of user-item meta-path instances and item-item meta-path instances. For the first item, the concatenation operation is different because of the dimension problem, and therefore, for the first item related to each user, the vector fed in MLP involves a user-item instance, mathematically,

$$h_{u,i=1} = [h_u; h_{\phi_{u\to 1}}; h_{i=1}] \tag{4.11}$$

After that, the final user-item rating $r_{u,i}$ calculates as follows.

$$r_{u,i} = MLP(h_{u,i}) \tag{4.12}$$

where the MLP contains a two-hidden-layer neural network with ReLU as the activation function and the sigmoid function as the output layer. According to [49, 58], neural network models can learn more abstractive features of data by using a small number of hidden units for higher layers, I employ a tower structure for the MLP component, halving the layer size for each successive higher layer.

I use implicit feedback loss with negative sampling [53, 130] as the whole loss function:

$$loss_{u,i} = -E_{j\sim P_{neg}}\left[\log\left(1 - r_{u,j}\right)\right] \tag{4.13}$$

where models the negative feedback drawn from the noise distribution $P_{neg}$, which is a uniform distribution following [58]. The whole TMER-RL process pseudo code is shown as Algorithm 3.

**Algorithm 3** The algorithm of TMER-RL.

```
Input: User set $U$, Item set $I$, user item interaction with timestamp, item meta information including Brand set $B$ and Category set $C$
Output: The converged TMER-RL model
 1: Initialize the node representations $h_U, h_I$ via pre-training DeepWalk [108] introduced in Chapter 4.3.2
 2: Explore user-item and item-item paths $\Phi$ according to reinforcement learning in Chapter 4.3.4
 3: Initialize the paths representations $h_\Phi$ using Word2Vec [92]
 4: for each $u_n \in U$ do
 5:    update user-item paths embeddings $h_{\phi_{u_n->i_m}}$ according to Eq. 4.6
 6:    for each $i_m \in I_{u_n}$ do
 7:       update item-item paths embeddings $h_{\phi_{i_m->i_{m+1}}}$ according to Eq. 4.6
 8:    end for
 9: end for
10: for each $u_n \in U$ do
11:    update item embeddings using two-layer attention according to Eq. 4.7-4.9
12:    concatenate user, item and related path embeddings to obtain the user's sequence representation according to Eq. 4.10-4.11
13: end for
14: while not converge do
15:    for each $batch \in all\_batch$ do
16:       calculate user-item ratings using MLP according to Eq. 4.12
17:       calculate loss according to Eq. 4.13
18:    end for
19: end while
20: return The converged model
```

### 4.3.7 Discussion of Explanation for Recommendation with Attention Mechanism

Attention model [4, 15] was first introduced in machine translation tasks and the attention weights were later widely used in natural language processing tasks as explanations in neural networks [94, 165]. Other than Natural Language Processing (NLP) tasks, the attention mechanism is also a near-ubiquitous method in recommendation tasks used as explanations in some works. [20, 26, 128, 150] However, there are different opinions on whether the attention mechanism could be used as a way to explain data [64, 119, 154].

In the proposed method TMER-RL, the item-item meta-path instances with attention weights learned by the Item-Item Path Attention module in Figure 4.4 are used as explanations. To be specific, for all existing item-item meta-path instances, I use reinforcement learning to explore some useful paths, because it is difficult to process all paths whose number is large. After obtaining the candidate item-item meta-path instances, I consider these paths as explanations for purchasing the target item. For example, in Figure 4.4, user *Alice* has bought *iPhone 11 pro*, *Phone case* and *Phone film* in a sequential order. For paths from *Phone case* to *Phone film*, the paths includes $Phone\ case \rightarrow Phone\ accessories \rightarrow Phone\ film$, $Phone\ case \rightarrow Miracase \rightarrow Phone\ film$ and $Phone\ case \rightarrow Abby \rightarrow Phone\ film$. In this situation, the explanations for buying *Phone film* are that *Alice* has bought other *Phone accessories*, *Alice* has bought other *Miracase* product, and that *Abby* has bought the *Phone case* and the *Phone film* together. Based on them, I use a self-attention module as Item-Item Path Attention module to learn a distribution of these paths (explanations) to further explain the recommendation. The learned weights are considered as the explainable weighted scores for the candidate item-item instances.

## 4.4 Experiments

Following, I present the experimental settings and give analysis of the evaluation results.

### 4.4.1 Experiment Settings

#### 4.4.1.1 Datasets

I use four public data collections to conduct experiments, Amazon Musical Instruments, Amazon Automotive dataset, Amazon Toys and Games and Goodreads dataset [140, 141]. The Amazon dataset [96] contains 29 categories, from which I choose thee first three datasets. Each dataset includes brand and category as the metadata. The Goodreads dataset is a public book online rating and review collection. I select mystery thriller crime genre books and choose authors and publishers as the metadata of books. More details are shown in Table 4.2.

I select the latest twelve items for each user and order these items by timestamps, and then I choose the first two items as bridge items, the next four items as training items and the rest as test items. I create the Heterogeneous Information Networks using *user*, *item*, *category* and *brand* in Amazon datasets, and using *user*, *book*, *author*, and *publisher* in Goodreads dataset, respectively. Last, user-item meta-path instances and item-item meta-path instances are explored according to Chapter 4.3.4.

| **Datasets** | **User** | **Item** | **Category** | **Brand** |
|---|---|---|---|---|
| Musical Instruments | 1450 | 11457 | 429 | 1185 |
| Automotive | 4600 | 36663 | 1592 | 3790 |
| Toys and Games | 9300 | 58743 | 820 | 5404 |
| | **User** | **Book** | **Author** | **Publisher** |
| Goodreads | 11800 | 12142 | 3633 | 1442 |

Table 4.2: Dataset information of TMER-RL.

#### 4.4.1.2 Evaluations

I leverage Top $K$ Hit Ratio (HR@$K$) and Top $K$ Normalized Discounted Cumulative Gain (NDCG@$K$) as the metrics. For each positive test instance $(u, i^+)$, I mix the ground truth item $i^+$ with $\mathcal{N}$ random negative items, rank all these $\mathcal{N}+1$ items and compute the HR@$K$ and NDCG@$K$ scores. I set $\mathcal{N} = 500$ and $K = \{1, 5, 10, 20\}$ for evaluations. For evaluation of the explainability of the recommendation, I use case studies to show the explanations in detail.

#### 4.4.1.3 Baselines

- **GRU4Rec** [56, 57]: GRU4Rec is a session-based recommendation method using GRU. For each user, I treat the training items as a session.

- **NARRE** [16]: NARRE utilizes the neural attention mechanism to build an explainability recommendation system.

- **MCRec** [58]: MCRec develops a deep neural network with the co-attention mechanism to learn rich meta-path based context information for recommendations.

- **NFM** [51]: NFM effectively combines the linear Factorization Machines (FM) and nonlinear neural networks for prediction under sparse settings.

- **Chorus** [142]: Chorus considers both relations among items and corresponding temporal dynamics to model the recommendation data in a knowledge-aware and time-aware way. The enhanced item representations improve the performance.

- **S$^3$Rec** [190]: S$^3$Rec utilizes the intrinsic data correlation to employ self-supervised learning tasks to learn data representations for sequential recommendation enhancement.

### 4.4.2 Implementation Details

For GRU4Rec, I use the ReChorus package [142] to implement the algorithm. For others, I directly run the provided code by each paper. To fairly compare the evaluation results, I modify each baseline's evaluation code as the same as TMER-RL. Especially, NARRE is a rating prediction model, I turn it into a link prediction model by rating 1 positive item and 500 negative items and ranking them.

### 4.4.3 Parameter Settings

I choose the first and second items for each user as the bridge items, the training or testing process, the last 6 items for each user as the testing positive items, and the remaining 4 are training items. The hyperparameters are carefully tuned to achieve optimal results in all experiments. I implement GRU4Rec, NARRE and NFM based on the details in papers by the PyTorch package. The meta-paths and settings in MCRec are the same as the original paper. The meta-paths in FMG are *User-Item(UI), User-Item-Brand-Item(UIBI)* and *User-Item-Category-Item(UICI)*.

In terms of the TMER compared in the following experiments, which removes the path generation via reinforcement learning module, I use *UIBI*, *UICI*, *UIBICI* and *UICIBI* as user-item meta-paths and *ICIBI*, *IBICI*, *ICICI*, *IBIBI*, *IUIUI*, *ICIUI* and *IBIUI* as item-item meta-paths, where U, I, B and C denote user, item, brand and category, respectively. For the proposed TMER-RL, the learning rate for the Amazon Musical Instruments dataset is $1e-5$, for the Amazon Automotive dataset is $5e-5$ and for Amazon Toys and Games is $1e-4$. The parameters for other baselines are searched for their best results. I set the maximum length of path instance exploration as 6 because I set 6 as the max length of meta-path in my previous work [18] and it is convenient to compare the performance.

### 4.4.4 Effectiveness Analysis on Recommendation Results

I compare TMER-RL with 6 other popular and recent baselines, including four sequential recommendations (GRU4Rec, Chorus, $S^3$Rec and my former proposed TMER), an explainable recommendation (NARRE), a path-based recommendation (MCRec) and a factorization machines-based recommendation (NFM) on 4 datasets with 500 negative samplings.

As shown in Table 4.3, the proposed TMER-RL achieves the best results and in most situations, TMER-RL could always give the correct items among 500 negative items,

| Datasets | Metrics | GRU4Rec | NARRE | MCRec | NFM | Chorus | S3Rec | TMER | TMER-RL | Improv. |
|---|---|---|---|---|---|---|---|---|---|---|
| | | NEG=500 | NEG=500 | NEG=500 | NEG=500 | NEG=500 | NEG=500 | NEG=500 | NEG=500 | |
| Amazon Musical Instruments | HR@1 | 0.3379 | 0.6171 | 0.5838 | 0.4457 | 0.1020 | 0.3448 | 0.8620 | **0.8762** | 1.65% |
| | HR@5 | 0.6833 | 0.8215 | 0.6178 | 0.5091 | 0.5386 | 0.5483 | 0.9473 | **0.9559** | 0.91% |
| | HR@10 | 0.8244 | 0.8828 | 0.6444 | 0.5505 | 0.8190 | 0.6448 | 0.9642 | **0.9761** | 1.23% |
| | HR@20 | 0.9272 | 0.9495 | 0.6819 | 0.6169 | 0.9651 | 0.7559 | 0.9736 | **0.9880** | 1.48% |
| | NDCG@1 | 0.3379 | 0.6171 | 0.5838 | 0.4457 | 0.1020 | 0.3448 | 0.8620 | **0.8762** | 1.65% |
| | NDCG@5 | 0.5187 | 0.7272 | 0.6009 | 0.4898 | 0.3185 | 0.4505 | 0.9306 | **0.9368** | 0.67% |
| | NDCG@10 | 0.5647 | 0.7472 | 0.6095 | 0.5037 | 0.4097 | 0.4817 | 0.9364 | **0.9438** | 0.79% |
| | NDCG@20 | 0.5910 | 0.7580 | 0.6189 | 0.5208 | 0.4474 | 0.5095 | 0.9389 | **0.9470** | 0.86% |
| Amazon Automotive | HR@1 | 0.4376 | 0.6521 | 0.6899 | 0.6033 | 0.1389 | 0.4457 | 0.9070 | **0.9258** | 2.07% |
| | HR@5 | 0.7708 | 0.8229 | 0.7295 | 0.6542 | 0.5746 | 0.6843 | 0.9632 | **0.9832** | 2.08% |
| | HR@10 | 0.8813 | 0.8703 | 0.7546 | 0.6937 | 0.6937 | 0.7722 | 0.9701 | **0.9910** | 2.15% |
| | HR@20 | 0.9526 | 0.9080 | 0.7867 | 0.7430 | 0.9604 | 0.8511 | 0.9745 | **0.9954** | 2.14% |
| | NDCG@1 | 0.4376 | 0.6521 | 0.6899 | 0.6033 | 0.1389 | 0.4457 | 0.9070 | **0.9258** | 2.07% |
| | NDCG@5 | 0.6146 | 0.7450 | 0.7100 | 0.6383 | 0.3587 | 0.5737 | 0.9550 | **0.9734** | 1.93% |
| | NDCG@10 | 0.6506 | 0.7604 | 0.7180 | 0.6517 | 0.4386 | 0.6019 | 0.9574 | **0.9761** | 1.95% |
| | NDCG@20 | 0.6688 | 0.7699 | 0.7261 | 0.6644 | 0.4746 | 0.6221 | 0.9585 | **0.9773** | 1.96% |
| Amazon Toy | HR@1 | 0.3914 | 0.6698 | 0.6802 | 0.5365 | 0.1753 | 0.4741 | 0.8327 | **0.8632** | 3.66% |
| | HR@5 | 0.7385 | 0.8745 | 0.7181 | 0.5963 | 0.5734 | 0.7324 | 0.9478 | **0.9647** | 1.78% |
| | HR@10 | 0.8618 | 0.9234 | 0.7412 | 0.6412 | 0.7886 | 0.8172 | 0.9605 | **0.9811** | 2.14% |
| | HR@20 | 0.9431 | 0.9570 | 0.7661 | 0.6951 | 0.9368 | 0.8870 | 0.9680 | **0.9908** | 2.36% |
| | NDCG@1 | 0.3914 | 0.6698 | 0.6802 | 0.5365 | 0.1753 | 0.4741 | 0.8327 | **0.8632** | 3.66% |
| | NDCG@5 | 0.5745 | 0.7813 | 0.6996 | 0.5775 | 0.3773 | 0.6117 | 0.9292 | **0.9456** | 1.76% |
| | NDCG@10 | 0.6147 | 0.7973 | 0.7070 | 0.5929 | 0.4472 | 0.6400 | 0.9337 | **0.9512** | 1.87% |
| | NDCG@20 | 0.6355 | 0.8059 | 0.7133 | 0.6069 | 0.4852 | 0.6578 | 0.9356 | **0.9534** | 1.90% |
| Goodreads | HR@1 | 0.2168 | 0.5007 | 0.4679 | 0.0789 | 0.1490 | 0.1967 | 0.5991 | **0.6276** | 4.76% |
| | HR@5 | 0.5477 | 0.7766 | 0.5469 | 0.1751 | 0.5224 | 0.4381 | 0.8499 | **0.8640** | 1.66% |
| | HR@10 | 0.7014 | 0.8570 | 0.5978 | 0.2414 | 0.7419 | 0.5527 | 0.9055 | **0.9168** | 1.25% |
| | HR@20 | 0.8502 | 0.9136 | 0.6600 | 0.3350 | 0.9075 | 0.6703 | 0.9395 | **0.9569** | 1.85% |
| | NDCG@1 | 0.2168 | 0.5007 | 0.4679 | 0.0789 | 0.1490 | 0.1967 | 0.5991 | **0.6276** | 4.76% |
| | NDCG@5 | 0.3910 | 0.6494 | 0.5083 | 0.1451 | 0.3379 | 0.3232 | 0.7968 | **0.8116** | 1.86% |
| | NDCG@10 | 0.4441 | 0.6755 | 0.5247 | 0.1677 | 0.4091 | 0.3603 | 0.8162 | **0.8305** | 1.75% |
| | NDCG@20 | 0.4818 | 0.6899 | 0.5404 | 0.1919 | 0.4514 | 0.3901 | 0.8251 | **0.8395** | 1.75% |

Table 4.3: Performance comparison with baselines. The last column, Improv., shows the improvement of TMER-RL compared with the second-best method.

especially in the Amazon Automotive dataset. The advantages also hold on other datasets. These results demonstrate that TMER-RL can achieve obvious advantages through explicitly modeling each user's sequential purchased information with the temporal paths, while NARRE and NFM ignore the sequential information for each user. MCRec ignores the item-item intrinsic relations and just utilizes user-item interactions to train the recommendation. GRU4Rec utilizes RNNs to model session-based recommendation. Chorus focuses on the way to better learn items' representations by combining the complement and substitute relations among items on the temporal items. $S^3$Rec uses different ways to pre-train embeddings to improve recommendation. However, the above three sequential recommendations all overlook the relation along paths and only focus on the temporal historical sequence, leading to worse results.

| Dataset | Metrics | TMER-RL | TMER | ¬ UI | ¬ II |
|---|---|---|---|---|---|
| Amazon Musical Instruments | $H@1$ | **0.9252** | 0.9216 | 0.8797 | 0.8739 |
| | $H@5$ | **0.9862** | 0.9855 | 0.9301 | 0.9300 |
| | $H@10$ | **0.9942** | 0.9934 | 0.9366 | 0.9372 |
| | $H@20$ | **0.9979** | 0.9965 | 0.9479 | 0.9471 |
| | $N@1$ | **0.9252** | 0.9216 | 0.8797 | 0.8739 |
| | $N@5$ | **0.9741** | 0.9736 | 0.9259 | 0.9247 |
| | $N@10$ | **0.9765** | 0.9761 | 0.9281 | 0.9272 |
| | $N@20$ | **0.9775** | 0.9770 | 0.9310 | 0.9297 |
| Amazon Automotive | $H@1$ | **0.9594** | 0.9482 | 0.8933 | 0.8979 |
| | $H@5$ | **0.9946** | 0.9903 | 0.9458 | 0.9464 |
| | $H@10$ | **0.9976** | 0.9939 | 0.9515 | 0.9531 |
| | $H@20$ | **0.9989** | 0.9963 | 0.9600 | 0.9618 |
| | $N@1$ | **0.9594** | 0.9482 | 0.8933 | 0.8979 |
| | $N@5$ | **0.9897** | 0.9844 | 0.9419 | 0.9422 |
| | $N@10$ | **0.9908** | 0.9857 | 0.9439 | 0.9445 |
| | $N@20$ | **0.9911** | 0.9863 | 0.9460 | 0.9468 |

Table 4.4: Impact of User-Item and Item-Item instances self-attention mechanisms. $H@K$ and $N@K$ mean $HR@K$ and $NDCG@K$, respectively.

### 4.4.5 Effectiveness Analysis on Modules of TMER-RL

To study the impact of the modules of TMER-RL for the recommendation performance, including the reinforcement learning path generation module, user-item and item-item instances modules, I further compare TMER-RL with three variants, namely TMER, ¬ UI and ¬ II. TMER removes the reinforcement learning module and generates user-item and item-item instances according to pre-defined meta-paths as mentioned in 4.4.3. ¬ UI means that I do not consider user-item instances and remove the corresponding self-attention module. ¬ II means that I remove the self-attention module for item-item instances. I report the compared results with 100 negative sampling in Table 4.4.

TMER-RL performs best on all evaluation methods, especially on the Amazon Automotive dataset. It shows that the user-item and item-item instances self-attention modules significantly boost the recommendation method, and adaptively adjust the importance of different instance paths. The self-learned user-item and item-item instance paths contribute to improving the performance of recommendation finally. Especially, using the item-item meta-path instances self-attention module helps TMER-RL improve 5.87% and 6.85% on HR@1 on the Musical Instruments dataset and Automotive dataset, respectively.

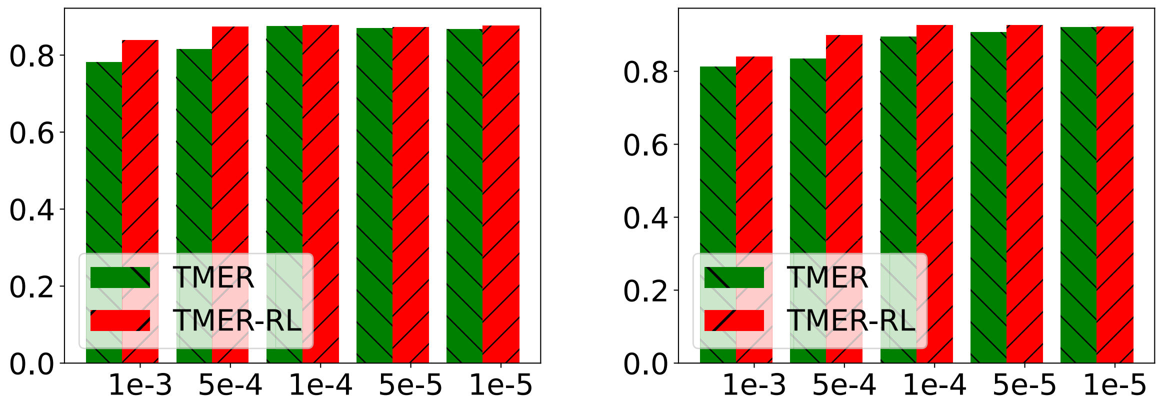


Figure 4.5: HR@1 w.r.t learning rate.

### 4.4.6 Variable Sensitivity

To test how the recommendation performs when the variable changes, I conduct a variable sensitivity test on the Amazon Musical Instruments dataset and the Amazon Automotive dataset. Figure 4.5 shows the influence of the learning rate of recommendation for HR@1. I set the learning rate of recommendation ranging among {1e-3, 5e-4, 1e-4, 5e-5, 1e-5}. The x-axis represents the learning rate of recommendation, which ranges among {1e-3, 5e-4, 1e-4, 5e-5, 1e-5}, and the y-axis means the HR@1 value.

Figure 4.5 shows that the HR@1 fluctuation of TMER-RL is minor than TMER, so I could get the conclusion that the performance of TMER-RL is more stable than TMER. Moreover, the proposed TMER-RL could always outperform TMER with different learning rates on these four datasets. This further proves the effectiveness of TMER-RL. Especially, on the larger dataset (Amazon Automotive), TMER-RL is more effective than TMER, which means the reinforcement learning path generation module could show superiority on a larger dataset, which is also required in the real-world scenario.

### 4.4.7 Impact of the Amounts of Training Data

To test whether TMER-RL could show superiority on the sparse dataset, I conduct the TMER-RL and TMER on different training proportions of the full Amazon Automotive dataset. I select {0.2, 0.4, 0.6, 0.8} as the training proportion (x-axis in Figure 4.6) and compare the NDCG@1 and NDCG@20 of TMER-RL and TMER.

It demonstrates that the more training proportion of the full dataset, the better the performance of TMER and TMER-RL. This is in line with common sense because a lower

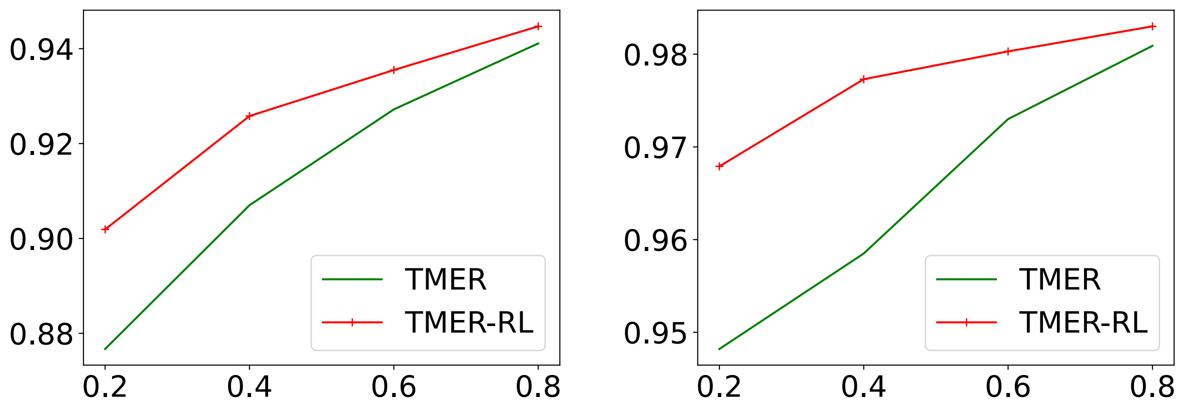


Figure 4.6: NDCG@1 and NDCG@20 w.r.t training proportion on Amazon Automotive dataset. The x-axis represents the training proportion of the full Amazon Automotive dataset.

training proportion means sparser data, which leads to fewer paths generation and captures less information among users and items, and results in lower performance of recommendation. However, in another perspective, when dealing with the least training proportion data (0.2) in Figure 4.6, the gap of TMER-RL and TMER is the most, and the proposed TMER-RL always keep the better performance. It could get the conclusion that the TMER-RL is more suitable for the sparser dataset, which means it could explore paths in a sparse data environment and make a better recommendation. Besides, the TMER-RL could always outperform TMER on NDCG@1 and NDCG@20 when the training proportion varies, further proving the effectiveness of TMER-RL.

### 4.4.8 Case Study for the Explainability

#### 4.4.8.1 Generating Explainable Paths for Recommendation via Meta-path

One of TMER-RL's contributions is to give explanations on instance paths while recommending preferable items. This is because TMER-RL generates multiple item-item instance paths according to user purchase history, and then it utilizes the attention mechanism to learn the weight of each item-item instance path and aggregate multiple item-item instance paths for each item-item pair. To demonstrate this, I show a random example drawn from TMER on the Amazon Musical Instruments dataset.

I randomly select a user whose id is $u273$ and track his item-item instance paths' parameters. In the training dataset, $u273$'s purchase history is $i2954$, $i2280$, $i4514$ and

$i11158$. I show several sampled item-item instance paths with high attention parameters in Figure 4.7 and demonstrate the explanations.

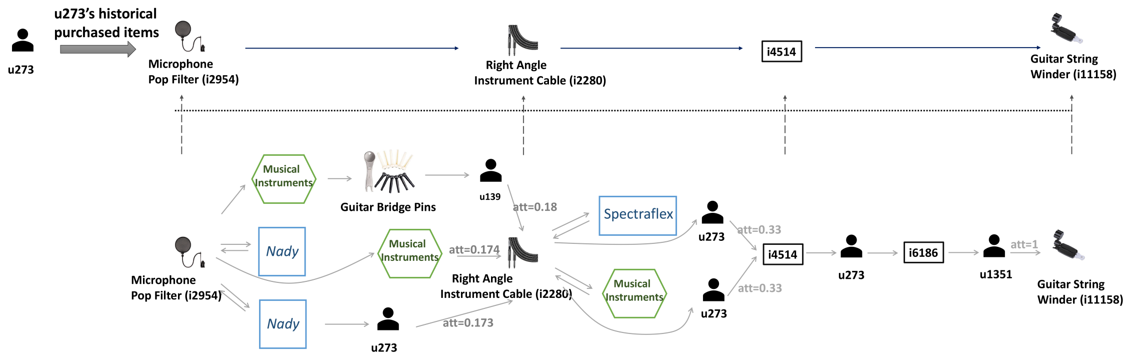

u273
u273's historical purchased items
Microphone Pop Filter (i2954)
Right Angle Instrument Cable (i2280)
i4514
Guitar String Winder (i11158)
Musical Instruments
Guitar Bridge Pins
u139
att=0.18
Spectraflex
u273
att=0.33
Nady
Musical Instruments
att=0.174
Right Angle Instrument Cable (i2280)
i4514
u273
i6186
u1351
att=1
Guitar String Winder (i11158)
Microphone Pop Filter (i2954)
Musical Instruments
att=0.33
u273
Nady
u273
att=0.173

Figure 4.7: It shows $u273$ and related historical purchased items in the upper part of the figure. In the lower part, it displays the reasoning item-item paths along historically purchased items related to $u273$. Green blocks represent the category attributes. Blue blocks represent the brand attributes. Black blocks without physical pictures do not have meta information in the dataset.

- According to Figure 4.7, there are three reasons for purchasing $i2280$. The most probable reason with the highest attention weight is that $u273$ has bought the last musical instrument category $i2954$ and $i2237$ with the attention weight 0.18. For the next item $i4514$, the reason for purchasing it is that the user $u273$ has bought $i2280$ who has the same brand and category with item $i4514$. There is also only one item-item instance path between some items because the item-brand and item-category data are sparse. Therefore, TMER-RL can model the reason through item-item instance paths with different weights.

- Besides, TMER-RL can capture sequential information according to user purchase history thanks to item-item instance paths. These item-item instance paths learn the reason path from the current item to the next item. In the whole model, these reason paths will feed to the item attention module. Therefore, TMER-RL can recommend learned sequential information.

#### 4.4.8.2 Generating Explainable Paths for Recommendation via Reinforcement Learning

Compared with randomly generating paths according to pre-defined meta-paths in my previous work [18], I design and implement a reinforcement learning framework to learn

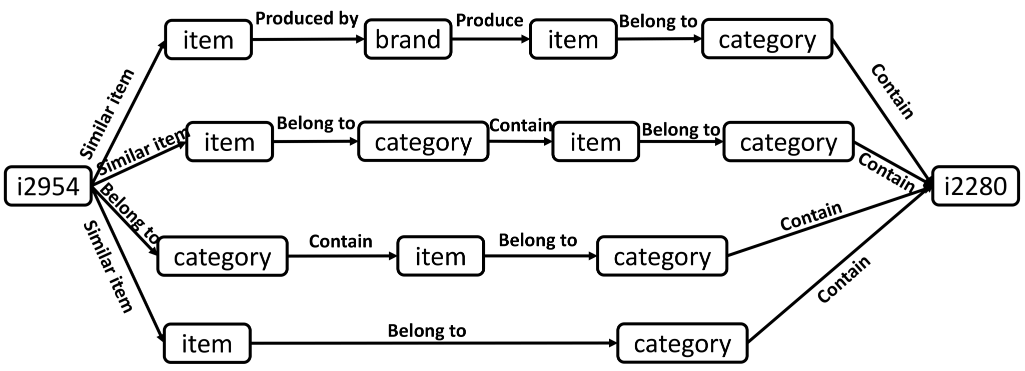


Figure 4.8: Summary of 10 generated paths via reinforcement learning from $i2954$ to $i2280$ for $u273$.

and mine instances. Also, instead of randomly choosing paths among a large number of instances, I design a rank method to sort the scores of generated paths and get the top $k$ paths to feed into the following modules. I provide a case study for path generation by reinforcement learning framework on Amazon Musical Instruments dataset. Details are shown in Figure 4.8.

To be consistent with 4.4.8.1, I also choose $u273$ as the example. $i2954$ (Microphone Pop filter) and $i2280$ (Right Angle Instrument Cable) are the first and the second items for $u273$. I summarize the generated 10 paths to schemes in Figure 4.8. The generated paths schemes contain, *IIBICI*, *IICICI*, *ICICI* and *IICI*.

Although the number of summarized schemes is less than that of pre-defined meta-paths, the generated paths via reinforcement learning contribute to recommendation more than random generation and selection, which could be proven in the above experiments. To be specific, when randomly generating, the learning paths will be exactly consistent with the pre-defined meta-paths, but it is unsure that the generated paths are useful for the recommendation. However, utilizing reinforcement learning to mine paths could at least guarantee the correlation of nodes on the paths because of the action calculation during exploration. The proposed ranking equation could obtain the $k$ most relevant paths as the explainable paths.

Moreover, the pre-defined meta-paths may not be able to define some more useful schemes. For example, in Figure 4.8, there are 4 schemes, but only *ICICI* is a pre-defined meta-path in TMER. Furthermore, if the meta-path defines all existing conditions and considers all situations, the search space would be extremely large, leading to high calculating complexity. Therefore, using reinforcement learning to explore schemes is a better choice.

### 4.4.9 User Study for the Explainability

For further studying the explainability of TMER-RL compared with random path generation with pre-defined meta-paths via TMER [18], I conduct a user study to evaluate whether humans prefer the TMER-RL generated explainable paths. Due to limited human efforts, I randomly select 100 pairs of user-item interactions from the Amazon Musical Instrument dataset and let 5 volunteers with some musical instrument knowledge and rich online purchase experience assess the ratio of the intuitive paths generated by two methods and whether the generated paths are intuitive for humans. The volunteers are required to answer the following two questions for each user-item interaction pair.

- How is the intuitive explainable paths ratio in the generated path set for each method?
- Which method can generate more intuitive explainable path sets, TMER-RL or TMER?

In the study, for the first question, the volunteers mark the reasonable paths generated by two methods for each user-item pair, and then I calculate the average ratio of the reasonable paths in all generated paths for each method. Further, for each user-item pair, volunteers select which explainable path set generated by two methods is better, and I count the ratio of the better path sets for each method. Statistical results are shown in Table 4.5: 53% of the generated paths through TMER-RL are considered intuitive by assessors, while for TMER, only 42% of paths are categorized as high-quality paths. This validates that given low-quality explainable paths take up a large proportion of all existing paths, reinforcement learning based approach (TMER-RL) still could find more explainable paths, compared with the random selection based TMER. From the whole perspective, compared to the baseline TMER, 57% paths generated by TMER-RL are considered more intuitive and explainable paths versus 43% by TMER. Specifically, path generation via TMER-RL tends to be accurate. Supposing generating reasons for purchasing *Harmonica Holder*, TMER-RL generates paths like from category *Harmonicas* to product *Harmonica* and leads to the target product. However, due to the randomness, path generation via pre-defined meta-paths is likely to generate more general information with high-frequency interaction, like the path from the widest category *Musical Instruments* to the target product, which contains less related information.

| | TMER-RL | TMER |
|---|---|---|
| #Average intuitive path w.r.t each ui pair | 53% | 42% |
| #Intuitive path set w.r.t all pairs | 57% | 43% |

Table 4.5: Explainability comparison between TMER-RL generated paths and TMER ones through the user study.

## 4.5 Summary

In this Chapter, I investigate the sequential users' historical behavior modeling on the recommendation graph and propose a novel reinforcement learning-based method, TMER-RL. Specifically, it explicitly models dynamic user-item interactions over time with path-based knowledge-aware explainable capabilities. I explore item-item paths between consecutive items with attention mechanisms for users' sequential behavior modeling via a reinforcement learning framework. It is validated as an effective graph-based explainable recommendation model, especially on the user's historical behavior modeling aspect.

CHAPTER 5

# Counterfactual Reasoning for Path-Based Explainable Recommendation

For explainability, this Chapter points out the universal controversy of attention-based explainability learning and proposes two algorithms to use counterfactual learning for explainability learning. The proposed model-agnostic explainability model is flexible to be seamlessly applied to any path-based recommendations. To contribute to the explainability research field a step further, I also propose a package of qualitative and quantitative evaluations. Experiments validate the superiority of the proposed counterfactual reasoning framework.

## 5.1 Overview

Nowadays, recommendation systems driven by knowledge graphs have achieved promising performance through various advanced graph neural networks [58, 77, 126]. However, the complexity of graph structure and rich information also brings a huge challenge to the model interpretability [180] compared with traditional recommendations such as collaborative filtering-based models.

To solve this problem, many researchers leverage graph-based models for recommendation and then explain their results via significant paths to the target items because they believe these paths can preserve real-world causality. Unfortunately, there are usually a lot of underlying paths to impact the final decision, which brings a huge challenge to select more explainable paths from the large candidate space. Recently, some work [18, 76, 90, 147, 189] have integrated the attention mechanisms into their models and learned the path weights for further explanation. These weights are usually reflected in an item's one-hop neighbors [147], users' purchase records [18, 76], and some external knowledge [90, 189], and can be evaluated by some visualization cases.

However, the concerns are mainly concluded as the following two aspects: **Firstly**, the attention mechanism is widely questioned for its interpretability because many studies have found the weak reliability that the attention mechanism performs [13, 43, 119, 154]. As shown in Figure 1.4 in Chapter 1, I run an attention-based model on 16 underlying paths three independent times and draw the path-level attention weights in the heat map where each block denotes a specific path and the darker orange color means higher attention weight. It demonstrates that the attention-based model can not ensure stable weight distributions via different independent running, which is far-fetched to convince customers to accept the explanation for the recommendation. **Secondly**, the attention mechanism used in graphs also tends to assign higher weights to those common frequent paths, which usually carry quite general and wide information. Whereas, those particular paths that carry a significant amount of explanation-specific semantics are not well captured (see further discussion on this phenomenon in Chapter 5.4.3.3).

A promising direction to overcome the above problems is counterfactual reasoning. Generally speaking, counterfactual reasoning follows the 'what-if' thinking: what will happen to the result if a condition does not hold anymore [93]. In particular, a condition that causes the final result hugely changes is regarded as an important reason. Inspired by this, in the recommendation scenario, I try to find causes by adding some slight disturbance to the candidate clues to see which clue perturbation makes the recommendation score change. If a slight disturbance of the path leads to a large decrease in the recommendation score, the current path is regarded as a significant one, and a small decrease indicates a less important one. This new direction might benefit with the following potentials: **First of all**, counterfactual reasoning mainly cares about model input and output, and goes beyond the inside mechanism differences among different models, which is perfect to support a model-agnostic explanation. This means I can use the same way on multiple recommendation models to see which one is more trustworthy; **Secondly**, the counterfactual mechanism can be more effective for those informative clues because general (wide) clues usually have lower uncertainty and they are more difficult to be disturbed to flip the recommendation score. This is promising to overcome the shortcoming of attention-based explanations that may easily overlook those informative paths.

Recent researches [67, 112, 129, 162] have started to explore the feasibility of counterfactual reasoning in recommendation explanation. However, most of them focus on the item [67] alone, the item's aspect feature [112, 129], or user feature [162]. They are not designed for the path-based explanation, which is one of the most convincing evidence forms for graph-structural data. Although [41] explores the counterfactual reasoning on the knowledge graph, the reasoning weights are tightly tailored to the proposed recommendation, which cannot be used to evaluate different recommendation backends.

To fill the gap between counterfactual reasoning and the explanation of path-based recommendation models, I propose a Counterfactual Path-based Explainable Recommendation (CPER for short). I design two effective counterfactual reasoning methods to find explainable paths from the perspective of their representations and the topological structures. In particular, I propose an optimization framework to learn appropriate perturbation factors on path embeddings (Chapter 5.3.1.1). Meanwhile, I conduct counterfactual reasoning on path topological structure by learning a path manipulation policy driven by reinforcement learning (Chapter 5.3.1.2). Besides, unlike traditional work that mostly evaluates their explanation via case studies, I propose a package of solutions to evaluate the explainability with both *qualitative* and *quantitative* ways (Chapter 5.3.1.3). I evaluate the explanation framework on three real-world datasets with a very typical path-based recommendation backend, which reveals the significant advantages of the proposed method. In summary, the main contributions of this Chapter are as follows:

- I propose a novel explainable framework for path-based recommendation via counterfactual reasoning conducted on both path representation and path topological structure.

- For the path topological structure-based counterfactual reasoning, I novelly devise a reinforcement learning method to learn a path manipulation strategy for perturbation in counterfactual learning.

- I propose a package of solutions to evaluate explainability quality for path-based recommendations. Unlike traditional attention-based explanation, which is mostly evaluated by case studies, the proposed measurements include both ***qualitative*** and ***quantitative*** methods, which can be widely used in comparing various explanation methods.

- Extensive experiments conducted on three real-world datasets further demonstrate the effectiveness of the proposed framework. I carefully compare the generated explanations with attention-based ones and the results reveal that my generated explanation method has higher stability, effectiveness, and confidence.

## 5.2 Preliminary

### 5.2.1 Problem Definition

**Definition 6.** ***Recommendation Graph.*** *Recommendation graph $G = (V, E)$ is a heterogeneous graph that contains three types of vertices including users $U = \{u_1, u_2, ..., u_m\}$, items $I = \{i_1, i_2, ..., i_n\}$ and items' attributes $A = \{a_1, a_2, ..., a_q\}$, where $V = (U, I, A)$. $e \in E$ is an edge connecting any pair of vertices in $V$.*

**Definition 7.** ***Path on Recommendation Graph.*** *Path $P_{u_m \to i_n}$ is all connected vertex sequences starting at a user $u_m$ and ending at an item $i_n$, indicating the relationship among users, items, and item attributes.*

Given the recommendation graph $G$, I generate the paths $P$ according to Algorithm 4. Specifically, for each user and item vertex, I explore the paths according to RandomWalk [158]. Furthermore, explainable paths are defined as paths with 4 to 6 lengths from users/items to items. For example, one of the explainable paths $p$ might be $u_1 \to i_1 \to a_1 \to i_2 \to a_2 \to i_3$, which is regarded as the explanation of purchasing item $i_3$.

**Problem 2.** ***(Counterfactual Path-based Explainable Recommendation)*** *Given a recommendation graph $G = (V, E)$, the path-based explainable recommendation aims to predict each user $u_m$'s preferred $l$ items $I^{u_m} = \{i_1^{u_m}, i_2^{u_m}, ..., i_l^{u_m}\} \in I$ and simultaneously generate paths $P_{\to I^{u_m}}$ where the end nodes are the predicted items $I^{u_m}$. Counterfactual-based explainable recommendation leverages counterfactual ideas for a further selection of the counterfactual paths $\mathscr{C}_{\to I^{u_m}}$ for the assistance of paths explainability enhancement. Therefore, the objective of counterfactual-based explainable recommendation is to provide recommendations for $I^{u_m}$ and learn the underlying counterfactual paths $\mathscr{C}_{\to I^{u_m}}$ for a more convincing explanation.*

For the above example explainable path, when performing counterfactual learning on path $p$, it would be slightly perturbed and change to $p'$, like $u_1 \to i_{100} \to a_1 \to i_2 \to a_2 \to i_3$, if this counterfactual path $p'$ significantly influence the recommendation prediction, it is regarded as an explainable path with high importance.

### 5.2.2 Why Counterfactual Reasoning Works?

Intuitively, most paths in the recommendation graph are very general, and only a few of them are inspiring for the explanation. For example, in the *Amazon Musical Instrument* dataset, the category *Musical Instrument* obviously holds the widest information and has the most link throughout all the category nodes. This leads to the paths related to the *Musical Instrument* category the most. However, from the perspective of information theory [12], those general paths appearing very frequently usually contain a very limited amount of information (a.k.a *lower uncertainty*), which are less helpful for the more informative explanation. Traditional attention mechanism intends to assign higher weights to those highly frequent paths but ignores those more informative paths with *higher uncertainty*. Unlike traditional attention mechanisms, counterfactual reasoning aims to add slight perturbation to these paths and see the consequence from the recommendation results [41, 67, 129]. Obviously, the general paths with *lower uncertainty* will be more difficult to be affected, but those informative paths with *higher uncertainty* can be easily disturbed, causing the results dramatically changed. Therefore, this new approach help to find more informative paths for recommendation explanation.

**Algorithm 4** Paths Exploration on Recommendation Graph

```
Input: User set $U = \{u_1, u_2, ..., u_m\}$, item set $I = \{i_1, i_2, ..., i_n\}$, item attribute set $A =$
    $\{a_1, a_2, ..., a_q\}, max\_path\_length, min\_path\_length$
Output: Path set $P$
 1: $P \leftarrow new\ List$
 2: for vertex $v \in \{U, I\}$ do
 3:     $path \leftarrow new\ List$
 4:     $path.append(v)$
 5:     while $l < max\_path\_length$ do
 6:         $v' \leftarrow Random(\mathcal{N}(path[-1]))$
 7:         $edge \leftarrow [path[-1], v']$
 8:         $path.append(v')$
 9:         $l \leftarrow len(path)$
10:         if $l > min\_path\_length$ and $path[-1] \in \mathcal{N}(v)$ then
11:             $P.append(path)$
12:         end if
13:     end while
14: end for
```

## 5.3 Methodology

### 5.3.1 Counterfactual Reasoning on Paths

To avoid the unreliability issue brought by attention-based explainable weights of paths, I learn the explainable path weights via counterfactual learning to replace the traditional attention weights. The main idea is that once performing perturbation $\gamma$ on each explainable path $p$ or the path set $P$, the decreasing score of the recommendation prediction score can be regarded as the explainable weight of the path or the path set. To achieve this, I conduct counterfactual reasoning from two perturbation perspectives, including path representation and path topological structure. Specifically, I conduct counterfactual reasoning on path representations by learning appropriate perturbation factors on path embeddings, and counterfactual reasoning on path topological structure via learning a reinforcement learning-based strategy to manipulate paths by replacing some path vertices.

#### 5.3.1.1 Counterfactual Reasoning on Path Representations

Let $x$ be the path representation and $x'$ be a disturbed representation generated by counterfactual reasoning on the same path. If feed this new representation $x'$ to the recommendation backend model $R(h_U,h_I,\{x_1,x_2,...,x_p\},\Theta)$, it can output a new recommendation prediction score $s'$ compared with the original $s$. Mathematically,

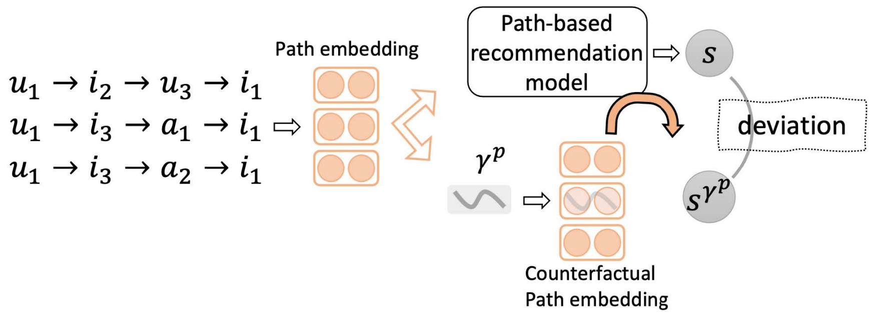
$$s = R(h_U,h_I,\{x_1,x_2,...,x_p\},\Theta) \tag{5.1}$$

$$s' = R(h_U,h_I,\{x'_1,x_2,...,x_p\},\Theta) \tag{5.2}$$

where path representation $x_1$ is disturbed to $x'$, and $R$ is introduced in Chapter 5.3.2, which can be calculated according to Eq. 5.9, 5.10, 5.11.

Unlike traditional attention-based explanation, which measures the path importance by attention weights, I evaluate the path significance from two aspects: how much is this perturbation added to the input $dis(x,x')$, and how much has the result changed $dis(s,s')$. If the perturbation on a path is very slight but causes a dramatic decrease in results, the corresponding path should be very informative and important. To this end, this method aims to learn a slight perturbation factor and at the same time find informative paths affected by this perturbation.

To realize the above two targets, I design slight perturbations on the path embeddings as shown in Figure 5.1. Specifically, the perturbation $\gamma^p$ is added to the path embeddings



Path embedding
$u_1 \to i_2 \to u_3 \to i_1$
$u_1 \to i_3 \to a_1 \to i_1$
$u_1 \to i_3 \to a_2 \to i_1$
Path-based recommendation model
$s$
deviation
$\gamma^p$
$s^{\gamma^p}$
Counterfactual Path embedding

Figure 5.1: Counterfactual reasoning on path representations.

to get the counterfactual path embeddings. After feeding them to the path-based recommendation backend, the prediction score would change to $s^{\gamma^p}$ from the original score $s$. The deviation is regarded as the explainable weight in CPER. For learning the slight perturbation, I minimize $dis(x,x')$ as follows:

$$\ell_1 = ||\gamma^p||_2^2 + \alpha||\gamma^p||_1 \tag{5.3}$$

where $\gamma^p$ is the perturbation vector on the path $p$. $||.||_2$ and $||.||_1$ are the L2 norm and L1 norm, respectively. $\alpha$ is the hyperparameter to balance the L2 norm and L1 norm. The above loss $\ell_1$ is to minimize the perturbation on the path embedding of path $p$. Besides, another target is to maximize the influence $dis(s,s')$ after perturbing the original paths, which can be ensured by minimizing the following target:

$$\ell_2 = -s_{u,i} + s_{u,i}^{\gamma^p} \tag{5.4}$$

where $s_{u,i}$ is the original recommendation score for user $u$ and item $i$; $s_{u,i}^{\gamma^p}$ is the new output of the recommendation model after changing the representation of path $p$. Considering $\ell_2$ is not differentiable when negative, I adopt a hinge loss to relax the constraint. Then the final loss from user $u$ to item $i$ is calculated as follows:

$$\mathscr{L}_{u,i} = \ell_1 + \lambda max(0, \beta + \ell_2) \tag{5.5}$$

where $\lambda$ and $\beta$ are the hyperparameters to trade off the weight of each term. Further, for learning the whole explainable model, the objective is to learn slight perturbations as follows.

$$\Gamma^\star = \underset{\Gamma}{\operatorname{argmin}} \left(\mathbb{E}_{u\in\mathscr{U}, i\in\mathscr{I}} \left[\mathscr{L}_{u,i}\right]\right) \tag{5.6}$$

where $\Gamma^\star = [\gamma_{u_1,i_1}^{p_1}, \cdots, \gamma_{u_m,i_n}^{p_k}]$ contains the optimal perturbations for each path related to user-item pairs trained according to the loss in equation 5.5. Then, perturbations are performed on each path respectively, and I choose the paths that have an impaired influence on the recommendation score ($s_{u,i} - s^{\gamma^p}_{u,i} > 0$) as explainable paths. This score also indicates the impact and explainable weight of the path quantitatively.

#### 5.3.1.2 Counterfactual Reasoning on Path Topological Structure

In addition to the high-dimensional perturbation approach, I also develop another method to perturb topological path structures. To achieve this goal, the aim is to enhance the interpretability of the perturbation, as compared to perturbation path representations. Specifically, I devise a path manipulation strategy using reinforcement learning, taking advantage of its powerful search capabilities.

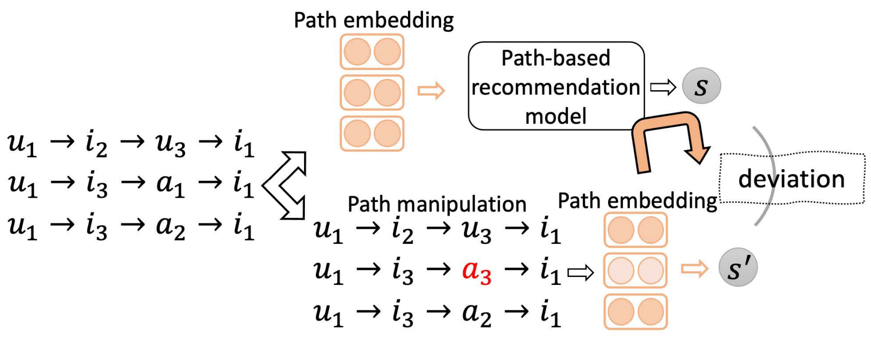

Path embedding
Path-based recommendation model
$s$
$u_1 \to i_2 \to u_3 \to i_1$
$u_1 \to i_3 \to a_1 \to i_1$
$u_1 \to i_3 \to a_2 \to i_1$
deviation
Path manipulation
Path embedding
$u_1 \to i_2 \to u_3 \to i_1$
$u_1 \to i_3 \to a_3 \to i_1$
$u_1 \to i_3 \to a_2 \to i_1$
$s'$

Figure 5.2: Counterfactual reasoning on path structure.

**Reinforcement Learning Framework.** Supposing there is a path set that will be fed into a path-based recommendation model, the desired operation is to replace some vertices on paths and see what will happen to the model result. The aim of the agent is to find a path-level operation sequence with fewer manipulations but makes the model's output score decrease as much as possible. This operation sequence can be treated as a trajectory. The agent's current action is dependent on its previously changed path set and the model output score, which can be approximated to a Markov Decision Process (MDP): $(\mathscr{S},\mathscr{A},\mathscr{T},\mathscr{W})$ where $\mathscr{S}$ is the state space, $\mathscr{A}$ is the action space, $\mathscr{T}$ is the state transition pattern, and $\mathscr{W}$ is the reward. $\mathscr{O}$ is the optimization of the model. In this framework, the recommendation system is the environment; each time the agent manipulates the path set and feeds it into the recommendation model, leading to a new path set state and a new predicted score. Therefore, the path set state, and the predicted score together serve as the environment state. The state transition $\mathscr{T}$ depends on the recommendation system.

**States.** The state $\int \in \mathscr{S}$ is defined as a combination of path set (collection) and the model output score like $(P_{\to i_n^u}, s_{u,i})$, where $P_{\to i_n^u}$ denotes all the paths to the item $i_n$ for user $u$. Taking Figure 5.2 as an example, the first path to item $i_1$ for user $u_1$ can be denoted as $p^1_{\to r_1^{u_1}} = (u_1, i_2, u_3, i_1)$. Here, the initial state $\int_0$ includes the tuple of all the paths and the original model score for the user-item pair $(u, i_n)$. Each time when the agent manipulates the path set collection, the modified collection together with the corresponding model score will be treated as a new state.

**Actions.** For a path collection $P = \{p_1, p_2, \cdots, p_K\}$, the agent tries to select several vertex positions from $P$ and replace them with some alternative vertex in $\mathscr{V}$. Therefore, the action space can be denoted by $\mathscr{A} = P \times \mathscr{V}$. I also set the constraint that the perturbed vertex should keep the same node type as the original one to keep the perturbation smallest to the greatest extent. Furthermore, to reduce the search space, I use two-hop adjacent and same-category vertices of each vertex in the knowledge graph as alternatives for each vertex on the path. The rationale for randomly selecting two-hop neighboring nodes as the candidate set is that two-hop neighboring vertex tends to have similar information to the original vertex. For example, if define $\mathscr{N} = \{\mathscr{N}_1, \mathscr{N}_2, \cdots, \mathscr{N}_{\mathscr{V}}\}$ as the alternative set for the agent, $\mathscr{N}_v \in \mathscr{N}$ contains node $v$'s two-hop neighboring nodes. Since most graph data follow the power-law distribution, the average size of $\mathscr{N}_v$ can be very small like $k \ll |\mathscr{V}|$, and then the action space can be reduced from $|\mathscr{P}| \times |\mathscr{V}|$ to $k|\mathscr{P}|$. Specifically, whether to select a vertex to be replaced is decided by a designed neural network, and the replacing vertex process from $\mathscr{N}$ is a random selection.

**Transition.** The transition $\mathscr{T}$ presents the transition probability from the current state to the next state, which can be denoted as a map function $\mathscr{T} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to [0,1]$.

**Rewards.** To calculate the reward, I first feed the paths into the fixed recommendation model to predict the positive user-item score $s_{u,i}$. Then, the reward can be calculated as follows.

$$\mathscr{W} = \begin{cases} -\zeta \cdot |C| - \epsilon \cdot |F| + \eta \cdot \Delta s & , \Delta s > 0, |F| \neq 0 \\ 0 & , \Delta s < 0 \\ -200 & , |F| = 0 \end{cases} \tag{5.7}$$

where $|C|$ is the number of selected paths of the agent, and $|F|$ is the number of the replaced vertices. These two terms ensure the least perturbation of the path collection. $\zeta$, $\epsilon$ and $\eta$ are the positive coefficients. $\Delta s = s_{u,i} - s^c_{u,i}$, where $s_{u,i}$ and $s^c_{u,i}$ are the original user-item score and the affected user-item score by counterfactual paths, both of which can be obtained from the fixed recommendation backend. The rationale is to perturb the least nodes $|F|$ and paths $|C|$ and lead to the largest recommendation score drop.

**Optimization.** To learn optimal manipulation strategy, I follow [155] to train the agent model by maximizing the accumulated rewards:

$$J(\theta) = \mathbb{E}_{\pi}(\mathscr{W}) \tag{5.8}$$

where $\mathbb{E}$ is the expectation of the rewards. The least perturbation and also more difference $\Delta s$ means the more explainable weights for the original paths. Therefore, I choose the path set with the largest reward as the explainable path set. For explainability evaluation, I combine the paths found by counterfactual reasoning on path representations and structures together and compare them with attention-based path explanation in Chapter 5.4.3 and 5.4.2. I further discuss these two proposed counterfactual reasoning methods in Chapter 5.4.2.2. Specifically, after learning the explainable paths via counterfactual reasoning on path representations and on path topological structure, I will regard the intersection as the final explainable paths for evaluation in Chapter 5.4.2.1 and 5.4.2.3.

#### 5.3.1.3 Evaluating the Explainability for Path-based Recommendation

Evaluating whether the explanation is trustworthy is very subjective and to the best of my knowledge, currently, there is seldom widely accepted measurement to evaluate path-based explainability. In this Chapter, I hope to push forward this area by proposing both qualitative and quantitative methods.

**Qualitative Evaluation.** I evaluate the ***stability*** of the explainable method by independently repeating the recommendation model learning multiple times and seeing whether the explainable path distribution is consistent. The more stable the explainable distribution is, the more reliable of the explainable method is. (see in Chapter 5.4.3.1). To evaluate the ***effectiveness*** of the proposed explainable method, I randomly add an irrelevant path to the path set to see what weight the explanation framework learns (see in Chapter 5.4.3.2). Intuitively, the irrelevant path's explainable weight should be as small as possible to make sense. Otherwise, the explainable weight is unreliable.

**Quantitative Evaluation.** I evaluate the ***confidence*** of explainability inspired by information theory, where high entropy means low certainty of the information. Therefore, I define the confidence of each explainable path as its uncertainty (a.k.a entropy), which can be calculated as follows: $-\sum_k c(p_k) log c(p_k)$, where $k$ is the number of paths, $c(p_k)$ is the weight of $k$-th path learned by attention mechanism or counterfactual method. Intuitively, a better explanation model should hold its found explainable paths more confidently and keep the path uncertainty relatively lower. I also propose to evaluate the explainability via ***informativeness***. Here, I feed the learned explainable paths back to the recommendation backend to see how much the learned explainable paths contribute to the recommendation performance. The closer to the original result, the more informative the learned explainable paths are, compared with all the rest paths. I also use a frequently used metric, ***fidelity***, to evaluate the explainability. It measures the decrease in the prediction score when removing different ratios of explainable paths from the input explainable paths. A larger fidelity value indicates stronger counterfactual weights and better explainability.

## 5.3.2 Path-based Recommendation Backend

Here, I design a lightweight path-based recommendation model as the backend. Note that this model can be seamlessly replaced by any other path-based recommendation model as long as they take paths as partial/total input.

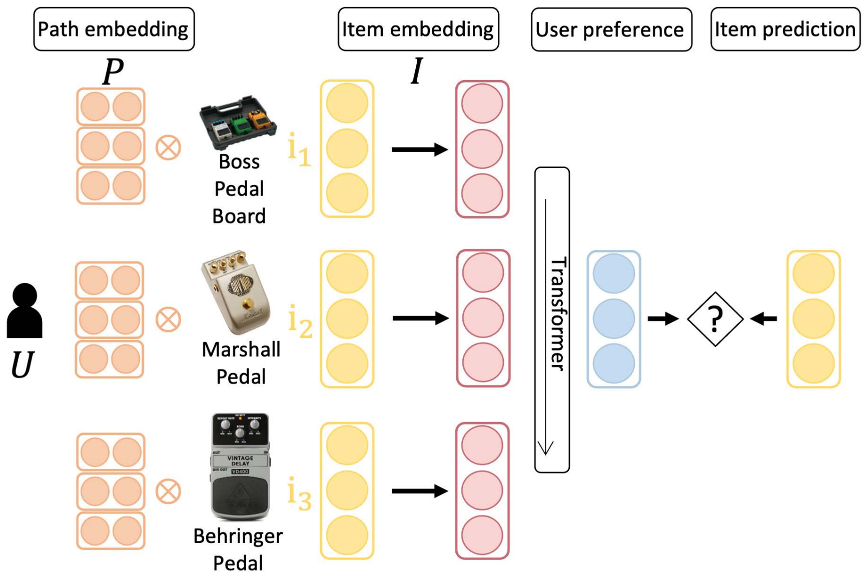


Figure 5.3: The sequential recommendation module. ⊗ represents two-layer attention for item embedding enhancement via related path embeddings.

The workflow of the recommendation backend is to model the users' behavior evolution and predict the user's preferred items. As shown in Figure 5.3, the predicted item scores $S$ can be obtained like $\mathscr{S} = (S|U, I, P, \Theta)$, which only relies on the users $U$, items $I$, paths $P$ and recommendation model parameters $\Theta$. The initial user and item embeddings could be obtained via temporal DeepWalk, which is a variation of a classical graph representation learning algorithm, DeepWalk [108]. In the traditional DeepWalk, it firstly explores paths from each node on the graph (RandomWalk) and then performs

Skip-gram [92] to obtain the nodes embeddings. However, DeepWalk only considers the static graph presentation learning due to the RandomWalk algorithm performed on the static graph, which overlooks the dynamic evolution of user-item purchase relations in recommendation data. Specifically, in the real sequential scenario [22, 33, 110], the future action (user's purchase behavior) is unknown, but in the static recommendation graph, it ignores the behavior's timestamp and regards the future behavior as the known action. This makes the presentation learning disobey real situations. Therefore, I propose a temporal RandomWalk to consider the temporal user-item purchase behavior evolution to generate temporal paths to ensure that there are no future behaviors along the path. To realize the purpose, I guarantee that the latter edges' (relations) timestamps are less than that of the former edges (relations) along each path. Then, I also adopt Skip-gram in the following step to learn the nodes embeddings. For the path embeddings, I use an average pooling on the related node embeddings to obtain. And I generate paths via the random path exploration for each item with the same settings in [18]. Specifically, assuming that user $u_m$ has some associated purchased items including $\{i_1^{u_m}, i_2^{u_m}, ..., i_o^{u_m}\}$, for each item $i^{u_m}$, I extract paths starting from user $u_m$ and ending at item $i^{u_m}$. The path embeddings can be easily obtained via the average pooling on the path node embeddings. Then the item embeddings can be updated by its representation and its associated paths as follows:

$$h'_{i_o^{u_m}} = f(W_{i_o^{u_m}} h_{i_o^{u_m}} + \sum_{z=1}^{Z} W_{p_z} h_{p_z} + b) \odot h_{i_o^{u_m}} \tag{5.9}$$

where $f(.)$ is ReLU activation function and $h_{i_o^{u_m}}$ is item $i_o^{u_m}$'s initial embedding. $W$ and $b$ are the weight matrix and bias vector, respectively. $p_z$ means the related $Z$ paths obtained in the last step, and $h'_{i_o^{u_m}}$ is the updated item embedding. With the above formula, I can infer the user's preference as follows:

$$h_{u_m}^{seq} = g(h'_{i_1^{u_m}}, h'_{i_2^{u_m}}, ..., h'_{i_o^{u_m}}) \tag{5.10}$$

where $h_{u_m}^{seq}$ represents user's preference; $g(.)$ is a sequential model such as Transformer [136] et al.. Finally, the predicted score between user $u_m$ and item $i$ can be calculated with a Multilayer Perceptron (MLP) unit as follows:

$$s_{u_m,i} = MLP(h_{u_m}^{seq}) \tag{5.11}$$

According to [58, 130], the recommendation model can be effectively trained via implicit feedback loss with negative sampling:

$$\mathscr{L}_{u,i^+} = -log\ s_{u_m,i^+} - E_{i^- \sim D_{neg}}[log(1 - s_{u,i^-})] \tag{5.12}$$

where the first term models the positive user-item score and the second term models the negative feedback. Each positive pair $(u, i^+)$ is associated with 1 negative pair $(u, i^-)$, which can be sampled from a uniform noisy distribution $D_{neg}$.

### 5.3.3 Model Optimization

For the counterfactual learning on path representations in Chapter 5.3.1.1, it optimizes the loss to learn the best perturbation embedding for each user-item pair according to Eq. 5.5, and optimizes all perturbation embeddings for all user-item pairs according to Eq. 5.6. For the counterfactual learning on path topological structure in Chapter 5.3.1.2, it optimizes the reward (Eq. 5.7) of reinforcement learning by Eq. 5.8 to optimize the best explainable path set.

In the overall learning procedure, it should first train the black-box path-based recommendation backend $R$ in Chapter 5.3.2. Then, the two counterfactual reasoning algorithms in Chapter 5.3.1.1 and 5.3.1.2 are trained respectively. Specifically, for the counterfactual reasoning on path representations (Chapter 5.3.1.1), it performs the learned perturbation on each path representation and feeds them to the recommendation backend model $R$, and paths whose perturbed score is less than the original one are regarded as explainable paths. The deviation of the recommendation score is considered as the importance weight of each path. For the counterfactual reasoning on path topological structure (Chapter 5.3.1.2), it adopts reinforcement learning to manipulate vertices on paths, and feed them to recommendation $R$. Likewise, it also uses the deviation score as the importance of paths.

## 5.4 Experiments

### 5.4.1 Experimental Settings

#### 5.4.1.1 Datasets

I evaluate the proposed recommendation on four real-world datasets, named Amazon Musical Instruments, Amazon Automotive, Amazon Toys and Games, and MovieLens. The statistics are shown in Table 5.1. They are subsets of a public product dataset, Amazon [96], which contains 29 categories of products from May 1996 to July 2014. This rich dataset also includes user-product interaction, product metadata and reviews from users. For better training, I only keep items bought more than 5 times and choose brand

and category as item attributes. The MovieLens dataset is extracted from the public MovieLens 25M Dataset. For meta information alignment, I use the genre and tag as movies' meta information.

#### 5.4.1.2 Baselines

I first compare the proposed CPER with the following 7 state-of-the-art baselines for the recommendation performance evaluation. Following are the details of them.

- **CountER**. [129] It adopts the counterfactual idea on items' aspects/attributes in recommendations and proposes a model-agnostic explainable recommendation with aspects-explanation.
- **PGPR**. [159, 178] The work leverages reinforcement learning to explore paths and also predict preferred items, which is a model-specific explainable recommendation.
- **ACVAE**. [161] The method involves adversarial Variational Bayes with a contrastive loss for modeling the user interaction history in recommendation.
- **GRU4Rec**. [57] It is a classical session recommendation, the first work to use recurrent neural network (RNN) for user-item interaction history session modeling in recommender systems.
- **Bert4Rec**. [124] The work proposes to adopt bidirectional encoder representations from transformer in sequential recommender systems.
- **SASRec**. [68] It relies on self-attention mechanism to improve the user-item interaction history learning for sequential recommendations.
- **SRGNN**. [157] The work uses graph neural networks with local interest and main purpose modeling for sequential recommendation enhancement.

Then for the explainability evaluation of CPER, I compare the explanation framework with attention-based explanations learned by the proposed path-based recommendation backend, qualitatively and quantitatively via evaluations introduced in Chapter 5.3.1.3.

#### 5.4.1.3 Implement Details

Some sequential recommendation baselines, including GRU4Rec, Bert4Rec, SASRec and SRGNN, are implemented by the RecBole framework [188]. For the rest baselines, I use

| Datasets | User | Item | Category/Genre | Brand/Tag |
|---|---|---|---|---|
| Musical Instruments | 1,450 | 11,457 | 429 | 1,185 |
| Automotive | 4,600 | 36,663 | 1,592 | 3,790 |
| Toys and Games | 9,300 | 58,743 | 820 | 5,404 |
| MovieLens | 50,000 | 20,118 | 19 | 1,127 |

Table 5.1: Dataset information of CPER.

the code implemented by the authors. All the comparison methods set the dimension as 100, keeping the same as CPER. For the Musical Instruments and Automotive datasets, the learning rate is 1e-3, for the Toys and Games dataset, the learning rate is 1e-4, and for the MovieLens dataset, the learning rate is 1e-5.

In the baseline comparison experiment in Chapter 5.4.5.1, I adopt the default parameter settings from RecBole framework [188], for GRU4Rec, Bert4Rec, SASRec and SRGNN. The rest baselines also keep the same parameters as the GitHub version implemented by the authors.

As for testing the effectiveness of different components of the proposed recommendation backend in Chapter 5.4.5.2, I conducted the experiment on the Amazon Musical Instruments dataset, and the learning rate is 1e-3. The training epoch is set to 300 for all variants.

When testing the explainability of the overall proposed counterfactual reasoning in Chapter 5.4.3 and 5.4.2, I conduct four experiments, including stability, effectiveness, confidence, informativeness, fidelity, and analysis of the reinforcement learning training process. The hyperparameters $\{\alpha, \beta, \lambda\}$ in counterfactual reasoning on path representations are $\{0.1, 0.5, 5\}$, respectively. For counterfactual reasoning on path topological structure, the hyperparameters $\{\zeta, \epsilon, \eta\}$ are $\{10, 10, 100\}$, respectively. I train 50 epochs for both counterfactual reasoning models and combine the generated explainable paths for comparison with attention-based weights.

#### 5.4.1.4 Research Questions

To evaluate the effectiveness and explainability of the proposed method, the experiment mainly answers the following questions.

- **RQ1**: From a *quantitative* perspective, how is the explainability of the proposed counterfactual reasoning on paths compared with the traditional attention-based explainability?

- **RQ2**: From a *qualitative* perspective, how is the explainability of the proposed counterfactual reasoning on paths compared with the traditional attention-based explainability?
- **RQ3**: How is the effectiveness of the proposed reinforcement learning-based path manipulation?
- **RQ4**: How does the proposed path-based recommendation perform compared with state-of-the-art backends, and how do the components contribute to the performance?

### 5.4.2 Quantitative Evaluation for Explainability of the Counterfactual Reasoning (RQ1)

I evaluate the explainability of the proposed counterfactual reasoning framework quantitatively. Specifically, I compare the method with traditional attention-based explanations using the previously proposed measurements in Chapter 5.3.1.3: *confidence*, *informativeness* and *fidelity*.

#### 5.4.2.1 Confidence

Inspired by information theory [12], I also introduce the uncertainty value (in Chapter 5.3.1.3) to evaluate the explainability. Specifically, a higher uncertainty value means lower confidence in the generated explainable paths. The learned explainable paths in this experiment are the intersection of two counterfactual learning methods for simplicity. To compare the confidence scores, I first randomly draw the generated path attention weights distribution, calculate the weight of the paths 3 times, and get the average entropy. Then, I summarize multiple user-item pairs and present the overall uncertainty value in Table 5.2. The entropy of attention weights probability and CPER paths probability are 2.25 and 1.87, respectively. It shows that the proposed method could learn lower entropy and more deterministic explanation results.

| | Uncertainty Value |
|---|---|
| Attention-based Explanation | 2.25 |
| My Explanation | 1.87 |

Table 5.2: Uncertainty value.

#### 5.4.2.2 Informativeness

I compare the two counterfactual reasoning methods with three baselines, including random selection, the attention module in Chapter 5.3.2 and TMER [18], on the informativeness metric, which is defined in Chapter 5.3.1.3. Specifically, if an explainable path makes sense, the user's current purchasing action should be more related to this path rather than the others. Therefore, I only feed the learned explainable paths back to the recommendation backend, and check the new prediction score. If the new score is very close to the original score, it means the generated explainable paths contribute a lot to the user preference analysis and thus can be regarded as more informative. To this end, I calculate the Mean Squared Error (MSE) between the new score and the original score on several user-item pairs and present the average value in Table 5.3.

The first baseline random guess is to randomly select paths for explanations and feed them back to the recommendation backend; the second baseline attention-based module is to select explainable paths learned by the attention mechanism and also feed them to the recommendation backend. For a fair comparison, both of them have the same path numbers as the proposed two method's average path numbers, shown by $\star$ in Table 5.3. The last baseline, TMER [18], is an attention-based explainable recommendation, which extracts the item-item paths as explanations of recommendations, a subset of the above two baselines' paths. Therefore, I keep the same ratio as my average explainable paths ratio for a fair comparison. The number of selected paths is shown by $\diamond$ in Table 5.3. Results in the table reveal that the two counterfactual-based reasoning methods contain more information for recommendations, especially the counterfactual reasoning on path structure method outperforms others. Moreover, the learned counterfactual-based explainable paths on larger datasets contribute more to the recommendation results, and other baselines also show similar trends.

#### 5.4.2.3 Fidelity

I also conduct the fidelity experiment to compare my counterfactual reasoning with the attention module in Chapter 5.3.2 and TMER [18] on the Amazon Automotive dataset, following the settings in [5]. This metric presents the decrease in the prediction score when removing different ratios of explainable paths from the input. A larger fidelity value shows stronger counterfactual weights, indicating better explainability. The results are shown in Figure 5.4. The x-axis is the sparsity of the explainable paths, which is the ratio of the explainable paths as input. To simplify the experiment, I use the intersection

| Datasets | Amazon Musical Instruments | | Amazon Automotive | | Amazon Toys and Games | | MovieLens | |
|---|---|---|---|---|---|---|---|---|
| Methods \ Evaluations | MSE | Num (Path) | MSE | Num (Path) | MSE | Num (Path) | MSE | Num (Path) |
| Random Guess | 0.998 | 30⋆ | 0.630 | 18⋆ | 0.337 | 11⋆ | 0.492 | 16⋆ |
| Attention-based Method | 0.661 | 30⋆ | 0.656 | 18⋆ | 0.263 | 11⋆ | 0.045 | 16⋆ |
| TMER | 0.668 | 11⋄ | 0.160 | 6⋄ | 0.044 | 4⋄ | 0.009 | 5⋄ |
| CR on Path Representations | 0.333 | 24.0 | **0.001** | 21.0 | 0.056 | 9.7 | 0.006 | 12.0 |
| CR on Path Structures | **0.261** | 37.3 | **0.001** | 14.3 | **0.004** | 13.0 | **7e-4** | 19.3 |

⋆: Average number of the proposed explainable paths.
⋄: Keep the same ratio as the proposed explainable paths ratio.

Table 5.3: MSE comparison between original recommendation score and only explainable paths feeding recommendation score for two perturbations on all datasets. The Num (Path) column is the number of learned explainable paths.

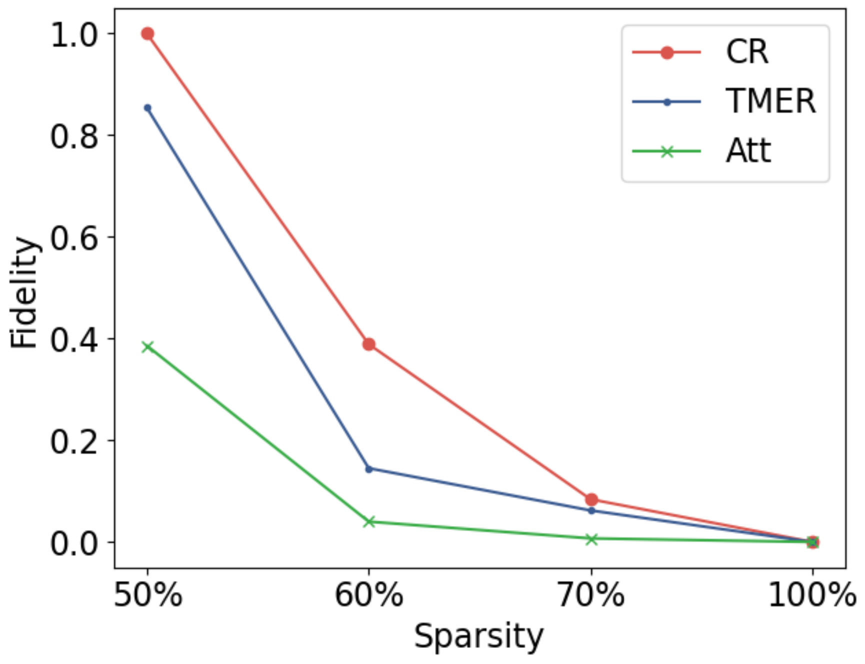


Figure 5.4: Fidelity of the counterfactual reasoning method and baselines. The x-axis is the sparsity of explainable paths input and the y-axis denotes fidelity scores.

of two counterfactual reasoning learned explainable paths set as the 100% explainable paths input. It shows that my method (CR) outperforms the baselines, which proves good fidelity of the proposed counterfactual reasoning.

### 5.4.3 Qualitative Evaluation for Explainability of the Counterfactual Reasoning (RQ2)

Recall in Chapter 5.3.1.3, I propose both qualitative and quantitative ways to evaluate the explainability of different explanation methods. Here, from the qualitative perspective, I compare the explainability of my proposed counterfactual reasoning with traditional attention-based explanations using the previously proposed measurements: *stability*, *effectiveness*, and *visualization* via traditional case study evaluation.

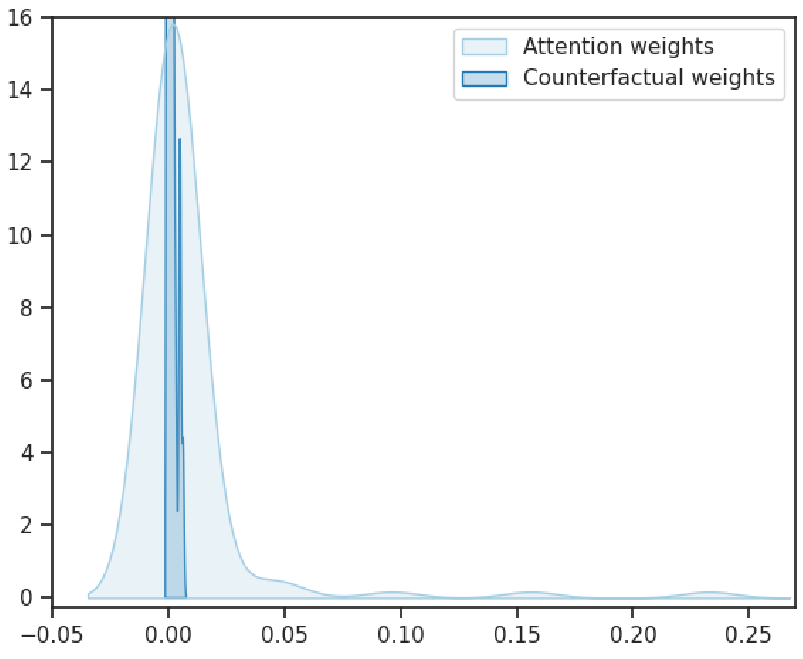


Figure 5.5: Stability study. It denotes the kernel density estimation (KDE) plot of both attention-based and counterfactual-based explainable weights.

#### 5.4.3.1 Stability

A better explanation framework should ensure their results are stable when I repeat their method multiple times because it would be ambiguous to determine the final explainable paths if the explainable model generates unstable path weights. To this end, I randomly draw 100 explainable paths and track the learned explainable weights by the counterfactual reasoning on path representations, and compare them with attention-based weights, after 10 runnings. Specifically, I draw the KDE (Kernel Density Estimation) plot of the learned attention-based path weights and counterfactual path weights to demonstrate the stability. For example, the thinner KDE plot denotes lower weight variants and better stability. In Figure 5.5, the x-axis denotes the weight variants after ten runs. The y-axis denotes the density (frequency) of attention weights and counterfactual weights. From the figure, it is straightforward that the KDE plot of counterfactual weight is much thinner than that of the attention weights. In particular, only 16% of paths' weights learned by the attention mechanism stay small variants, but all the counterfactual weights have small variants. This validates that the proposed method is more stable.

#### 5.4.3.2 Effectiveness

Intuitively, if the model learns a high weight for the "pranking" path, the explanation method is less effective and less reasonable. Therefore, to simulate and see how the counterfactual reasoning model and attention mechanism learn the importance weight of "pranking" path, I randomly draw the explainable weights learned by the counterfactual reasoning model, then I add a "pranking" path to the original path set by randomly choosing an irrelevant path (for example, a path from other distant users) to the target item and see what happens to the results. For a fair comparison, I also independently train each explanation weight ten times. In Figure 5.6, the "pranking" paths are presented as path #16. The x-axis denotes the path id, and the y-axis represents the learned explanation weight. The colored area of each bar indicates the range of learned weights. From the left figure, I find that the attention mechanism fails to separate the "pranking" paths from the original set. The reason may be that the irrelevant path has some common characteristics contained in the original paths. For many attention-based recommendation models, these "pranking" paths might not cause huge damage to the recommendation results, but using these paths for the explanation is far-fetched. This observation reveals the awkward gap between using an attention mechanism for recommendation performance and explanation. Compared with attention-based methods, the proposed method is indeed better at exploring the particular path, which is informative for the explanation.

#### 5.4.3.3 Visualization Comparison via Case Study.

Here, I randomly select two explainable path cases by the path representation perturbation and reinforcement learning-based path manipulation, respectively. In Figure 5.7, I randomly select a user in the Amazon Musical Instrument dataset and list the explainable paths found by the counterfactual reasoning on path representation. The normalized weights of these three paths are $[0.190, 0.189, 0.052]$, where the third path

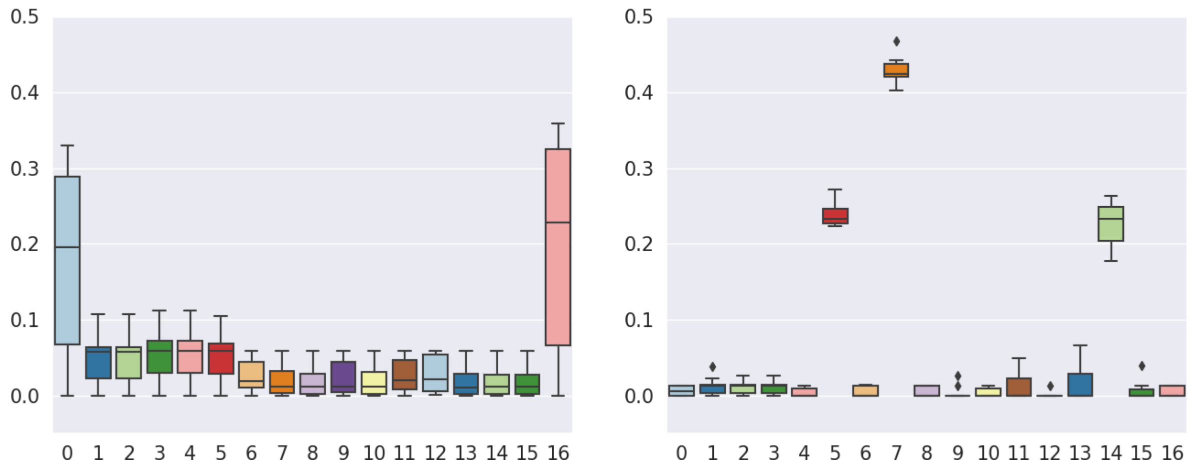


Figure 5.6: Effectiveness study. The X-axis denotes the path ID and the y-axis represents the range of explainable weights learned by 10 independent runnings. (Path #16 is the added pranking path.)

has the smallest weight. This is consistent with the commonsense because this path only contains very general information such as "*category: Musical Instrument*". Compared with other information provided in the first two paths, "*category: Musical Instrument*" is the widest category in the dataset. Therefore, compared with the other specific explanation path, the third path should be the worst explanation and deserves the lowest explainable weight. However, if I use the attention mechanism to learn the explainable weights of these three paths, the normalized weights are $[0.0279, 0.0512, 0.0512]$, where the first path has the smallest weight.

For the reinforcement learning-based counterfactual explainable method, I also draw the explainable paths for purchasing the "Boss ME70 Guitar Pedal". Note that there are some paths that overlap with my first proposed method. To better illustrate the second method, I ignore the same paths and only present some different ones in Figure 5.8, where the first two paths are selected by my method but ignored by the attention-based method and the third one is treated as an unimportant path by the method but with very high weight in the attention-based method. Figure 5.8 demonstrates that the first two paths contain a very informative relation with the target item "Boss ME70 Guitar Pedal". The third path is more general and less informative because of the redundant item node and musical instrument category.

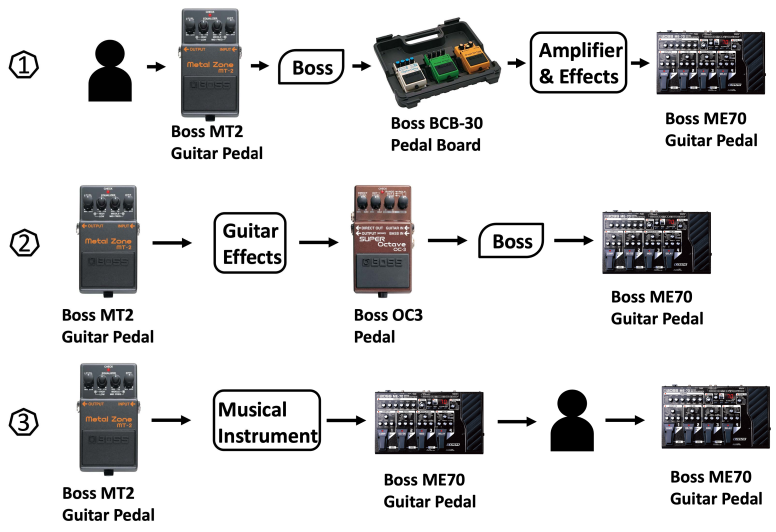


Figure 5.7: Case study for generating explainable paths through path representation perturbation. Boss is a brand, and others in round rectangles are the categories. Item names have been simplified to the keywords because of the original long names.

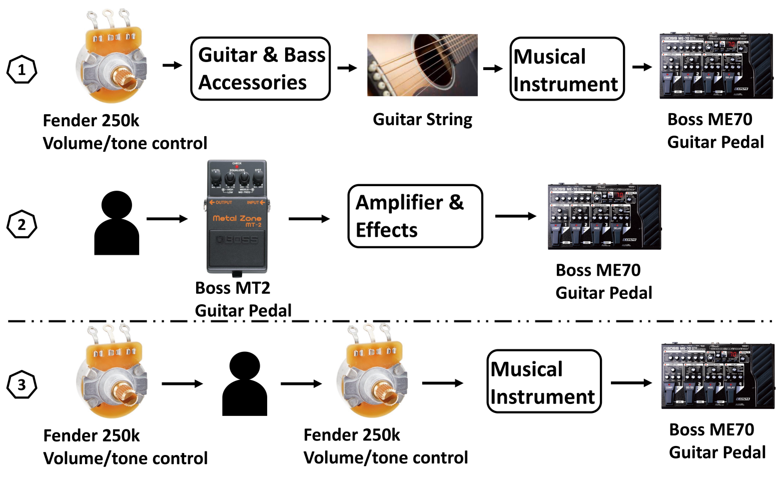


Figure 5.8: Case study for generating explainable paths through RL-based path manipulation. Contents in round rectangles are the categories. Item names have been simplified to the keywords because of the original long names. The first two paths are selected by RL-based counterfactual reasoning, and the last one has a high explainable weight via attention training.

### 5.4.4 Analysis on Reinforcement Learning-based Path Manipulation (RQ3)

To present the effectiveness of the reinforcement learning model, I track the training process for a randomly selected user-item pair with 50 iterations and repeat the training process three times with different seeds. The accumulated rewards are presented in the

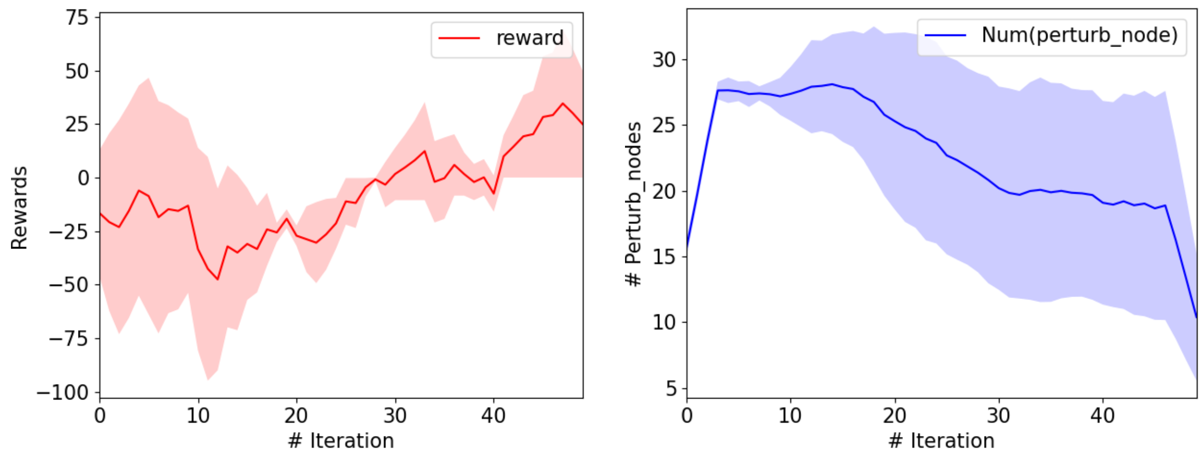


Figure 5.9: Reward & Perturbation w.r.t iteration.

left figure in Figure 5.9 where the dark color line is the mean value, and the shaded area is the range of values. The right figure in Figure 5.9 presents the number of perturbed nodes. The left figure demonstrates a gradual increase in the reward curve until it becomes steady. The right figure demonstrates that the number of perturbed nodes dropped as the agent gradually explored the appropriate policy. These observations indicate that the proposed reinforcement learning-based agent successfully explores a path manipulation policy that tried to cause a larger result drop with a relatively small manipulation cost. To better illustrate the results from counterfactual reasoning on path representation and path structure, I further study the explainable paths generated by these two methods as follows.

### 5.4.5 Effectiveness Analysis on the Recommendation Model (RQ4)

#### 5.4.5.1 Overall Performance

I compare the proposed recommendation model with seven state-of-the-art baselines on Hit Ratio (HR), and Normalized Discounted Cumulative Gain (NDCG). For each baseline, I sample 500 negative items and report the prediction results in Table 5.4. It demonstrates that CPER obtains the best NDCG performance on all datasets. This means that it can hit the ground-truth label as the first-ranking position with a higher probability than other comparison methods, which illustrates CPER can better capture the most important information. However, it does not perform well on HR@{10,20,30} on three Amazon datasets, especially on Amazon Automotive and Amazon Toys and

| Datasets | Metrics@ | $K$ | CountER | PGPR | ACVAE | Gru4Rec | BERT4Rec | SASRec | SRGNN | CPER | Improvement |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Amazon Musical Instruments | HR@ | 5 | 0.0345 | 0.0177 | 0.0137 | 0.0561 | 0.0462 | 0.0566 | 0.0662 | **0.0833** | 25.83% |
| | | 10 | 0.0486 | 0.0243 | 0.0243 | 0.0737 | 0.0662 | 0.0724 | 0.0917 | **0.0931** | 1.53% |
| | | 20 | 0.0760 | 0.0382 | 0.0453 | 0.1037 | 0.0959 | 0.1097 | **0.1193** | 0.0931 | -21.96% |
| | | 30 | 0.1254 | 0.0512 | 0.0674 | 0.1254 | 0.1159 | **0.1386** | 0.1379 | 0.1098 | -20.78% |
| | NDCG@ | 5 | 0.0253 | 0.0153 | 0.0101 | 0.0369 | 0.0293 | 0.0305 | 0.0458 | **0.0796** | 73.80% |
| | | 10 | 0.0297 | 0.0175 | 0.0127 | 0.0426 | 0.0355 | 0.0372 | 0.0538 | **0.0803** | 49.26% |
| | | 20 | 0.0367 | 0.0211 | 0.0181 | 0.0501 | 0.0429 | 0.0422 | 0.0608 | **0.0830** | 36.51% |
| | | 30 | 0.0434 | 0.0239 | 0.0228 | 0.0548 | 0.0472 | 0.0466 | 0.0648 | **0.0864** | 33.33% |
| Amazon Automotive | HR@ | 5 | 0.0227 | 0.0131 | 0.0099 | 0.0586 | 0.0504 | 0.0435 | 0.0670 | **0.0797** | 18.96% |
| | | 10 | 0.0355 | 0.0204 | 0.0201 | 0.0786 | 0.0683 | 0.0622 | **0.0961** | 0.0800 | -16.75% |
| | | 20 | 0.0741 | 0.0348 | 0.0400 | 0.1166 | 0.0983 | 0.0915 | **0.1259** | 0.0837 | -33.52% |
| | | 30 | 0.1047 | 0.0483 | 0.0617 | 0.1407 | 0.1217 | 0.1126 | **0.1450** | 0.0943 | -34.97% |
| | NDCG@ | 5 | 0.0147 | 0.0111 | 0.0070 | 0.0413 | 0.0336 | 0.0306 | 0.0469 | **0.0797** | 69.94% |
| | | 10 | 0.0188 | 0.0135 | 0.0104 | 0.0477 | 0.0393 | 0.0366 | 0.0563 | **0.0798** | 41.74% |
| | | 20 | 0.0286 | 0.0172 | 0.0155 | 0.0573 | 0.0469 | 0.0440 | 0.0638 | **0.0807** | 26.49% |
| | | 30 | 0.0351 | 0.0201 | 0.0200 | 0.0624 | 0.0519 | 0.0485 | 0.0679 | **0.0830** | 22.24% |
| Amazon Toys and Games | HR@ | 5 | 0.0269 | 0.0144 | 0.0106 | 0.0906 | 0.0678 | 0.0817 | 0.0726 | **0.0939** | 3.64% |
| | | 10 | 0.0471 | 0.0201 | 0.0209 | **0.1429** | 0.1001 | 0.1172 | 0.1082 | 0.0943 | -34.01% |
| | | 20 | 0.0715 | 0.0315 | 0.0411 | **0.2006** | 0.1455 | 0.1528 | 0.1539 | 0.1005 | -49.90% |
| | | 30 | 0.1011 | 0.0413 | 0.0623 | **0.2330** | 0.1778 | 0.1772 | 0.1823 | 0.1184 | -49.18% |
| | NDCG@ | 5 | 0.0155 | 0.0128 | 0.0075 | 0.0585 | 0.0437 | 0.0537 | 0.0475 | **0.0939** | 60.51% |
| | | 10 | 0.0219 | 0.0147 | 0.0110 | 0.0754 | 0.0541 | 0.0652 | 0.0590 | **0.0940** | 24.67% |
| | | 20 | 0.0287 | 0.0176 | 0.0162 | 0.0900 | 0.0656 | 0.0741 | 0.0705 | **0.0956** | 6.22% |
| | | 30 | 0.0350 | 0.0197 | 0.0207 | 0.0969 | 0.0725 | 0.0793 | 0.0765 | **0.0994** | 2.58% |
| MovieLens | HR@ | 5 | 0.0425 | 0.0120 | 0.0102 | 0.0901 | 0.1134 | 0.1024 | 0.1092 | **0.1397** | 23.19% |
| | | 10 | 0.0782 | 0.0123 | 0.0200 | 0.1633 | 0.2001 | 0.1738 | 0.1986 | **0.2486** | 24.24% |
| | | 20 | 0.1392 | 0.0128 | 0.0396 | 0.2806 | 0.3272 | 0.2762 | 0.3334 | **0.4058** | 21.72% |
| | | 30 | 0.1895 | 0.0131 | 0.0596 | 0.3731 | 0.4140 | 0.3519 | 0.4227 | **0.5148** | 21.79% |
| | NDCG@ | 5 | 0.0253 | 0.0119 | 0.0072 | 0.0550 | 0.0683 | 0.0626 | 0.0666 | **0.1026** | 50.22% |
| | | 10 | 0.0368 | 0.0120 | 0.0106 | 0.0785 | 0.0961 | 0.0856 | 0.0953 | **0.1397** | 45.37% |
| | | 20 | 0.0520 | 0.0122 | 0.0156 | 0.1079 | 0.1281 | 0.1113 | 0.1292 | **0.1803** | 39.55% |
| | | 30 | 0.0627 | 0.0122 | 0.0199 | 0.1276 | 0.1466 | 0.1274 | 0.1483 | **0.2038** | 37.42% |

Table 5.4: Comparison with baselines on {HR, NDCG}@{5, 10, 20, 30}. The best performance is bold, and the second-best performance is underlined. The last column presents CPER performance improvement compared with the best baseline.

Games. The rationale is that these two datasets are sparser than Amazon Musical Instruments and MovieLens, resulting in less information on user-item interactions and paths. Meanwhile, CPER is highly related to paths on the recommendation graph, which makes it more sensitive to the sparsity of the dataset. However, this shortcoming has little influence on the following module, counterfactual reasoning for explainability, because the counterfactual reasoning methods focus more on the deviation of recommendation scores. Even if the foremost indicator of this work is not the recommendation performance, but the explainability, I admit this limitation of this path-based recommendation backend and will improve it in the future.

#### 5.4.5.2 Effectiveness of Different Components

I further evaluate the effectiveness of the main components in CPER by comparing different variants on the Amazon Musical Instrument dataset. I report the results of

| Variants | HR@20 | NDCG@20 |
|---|---|---|
| CPER | **0.0931** | **0.0830** |
| CPER¬TempDW | 0.0866 (↓ 7%) | 0.0797 (↓ 4%) |
| CPER¬Trans | 0.0763 (↓ 18%) | 0.0440 (↓ 47%) |
| CPER¬Att | 0.0587 (↓ 37%) | 0.0490 (↓ 41%) |

Table 5.5: Different variants result on {HR, NDCG}@20 for comparison. ¬ represents the variant without the following module. The best results are in bold.

{HR, NDCG}@20 in Table 5.5. For the last column, I show the smallest percentage drop compared with CPER performance for each variant on two evaluation metrics. Specifically, CPER is the complete model, which obtains the best results. CPER¬TempDW removes the pre-training phrase with temporal DeepWalk, which decreases 3.98% on NDCG@20. CPER¬Trans takes the average of history item representations as the path representation instead of using a more powerful sequential model Transformer, the performance of which decreases 18.05% on HR@20. CPER¬Att means I remove all the attention layers and the related module, the item enhancement via path embeddings. The results then drop down by 36.95% on HR@20. Note that most recommendation models leverage these attention units for pursuing recommendation accuracy, and therefore they indeed contribute much to the prediction results. However, they are not intentionally designed for model explainability, making the attention-based explainability very limited as discussed in the previous explainability evaluation.

## 5.5 Summary

This Chapter focuses on the explainability of graph-based explainable recommender systems, and proposes a model-agnostic counterfactual path-based explainable recommendation algorithm. Specifically, I propose two methods to generate counterfactual paths: one is to apply slight perturbation on the path representations, and the other is to perform perturbation as little as possible on path topological structure via our designed reinforcement learning, both of whose objective is to drop the recommendation score as much as possible. For a better comparison of the quality of explainability with traditional attention-based explanation in the recommendation, I also introduce a package of solutions for qualitative and quantitative evaluation. Through experiments, both two counterfactual explainable methods obtain more reliable and trustworthy explainable paths compared with the traditional explainable attention weights.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

The research objective of the thesis is to model users' historical behaviors on the recommendation graph, and learn the users' preferences for recommendations, and provide reliable explanations. The thesis has explored the research from three aspects, including graph data, the recommendation, and explanation.

The graph data aspect focuses on the imbalanced characteristic of recommendation data. This issue leads to an imbalanced graph structure in which most users/items on the recommendation graph have few interactions (degrees). This phenomenon conforms with long-tail distribution, and results in unfairness on the tail vertices on the recommendation graph. To alleviate the unfairness problem due to the imbalanced graph structure in the recommendation scenario, I study the degree evolution of vertices, identify three types of vertice groups, and propose FairDGE, a novel dynamic graph embedding algorithm, that learns vertices' degree evolution and encodes fair embeddings for recommendation. The proposed methods alleviate the fairness problem without too much performance sacrifice.

The recommendation aspect focuses on the issue of inadequate understanding and modeling of user historical behaviors. To address the issue, I propose a reinforcement learning framework to explore the explainable paths, instead of traditional random explainable paths selection in the massive and heterogeneous recommendation graph. Moreover, I also sequentially model each user's historical behavior evolution using the explored meaningful explainable paths for high-quality recommendations. The proposed method is validated as a state-of-the-art graph-based explainable recommendation.

The explanation aspect focuses on the issue of unreliable explanations of existing work. This issue is mainly due to the mostly used attention mechanism for explainability modeling, which has been controversial these years. The attention mechanism was proposed for performance improvement but not explainability and it shows instability explainability modeling problem. To this end, I propose to use counterfactual learning for path-based explainability modeling. Specifically, I propose two post-hoc explainability modeling algorithms, including the perturbations on the path representations, and the perturbations on the topological path structures. After the perturbed paths are fed back to the backend recommendation model, the more the recommendation score decreases compared to the original recommendation score, the greater the interpretability weight of the original explainable paths. I also propose a package of qualitative and quantitative evaluations for the path-based explainability. Finally, the proposed counterfactual-based reasoning is validated as the state-of-the-art explainability modeling.

In summary, the thesis offers a deeper understanding of graph-based explainable recommender systems. The proposed methods offer valuable insights for future research in this field.

## 6.2 Future Work

Large language models (LLMs) have emerged these months thanks to their strong ability to model textual meaning. Some early work [6, 32, 48, 60, 62, 75, 133, 175, 181] that have initially explored the application of LLMs in the field of recommendation. However, till now the LLMs cannot effectively improve the performance of recommendations, even worse than the graph-based recommendations. The reason is: 1) LLMs require a large amount of data to train effectively, especially if the goal is to identify complex patterns and relationships among items and users. If the available data is limited, the model's predictions tends to be inaccurate and unreliable. 2) LLMs heavily rely on prompt quality to generate answers. How to design proper prompts to make the LLMs understand and learn the data is a crucial challenge to solve. Therefore, applying the advantages of LLMs to model large amounts of recommendation data and achieve unprecedented recommendation performance is a research area that has the potential to be explored in the future.

Moreover, according to the ablation study of [153], the data augmentation of users' and items' profiles contributes the most to the performance of the recommendation, which illustrates that LLMs show the advantage of textual modeling. This promotes that LLMs will bring large benefits when modeling textual data for recommendations, like conversational recommendations [54], explainable recommendations [81], and so on.

From a larger viewpoint, LLMs also have the potential ability to address severe problems in graphs and bring more benefits for large graphs. Early work [11, 23, 125] has discussions on this research field and achieved some results. If the LLMs assist the graphs to achieve a better ability to understand and analyze graph structure information, this will benefit tremendous underlying downstream applications.

# Bibliography

[1] H. Abdollahpouri and S. Essinger, *Towards effective exploration/exploitation in sequential music recommendation*, arXiv preprint arXiv:1812.03226, (2018).

[2] Q. Ai, V. Azizi, X. Chen, and Y. Zhang, *Learning heterogeneous knowledge base embeddings for explainable recommendation*, Algorithms, 11 (2018), p. 137.

[3] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv, (2014).

[4] D. Bahdanau, K. H. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, in 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[5] M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. C.-H. Lam, and Y. Zhang, *Robust counterfactual explanations on graph neural networks*, Advances in Neural Information Processing Systems, 34 (2021), pp. 5644–5655.

[6] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He, *Tallrec: An effective and efficient tuning framework to align large language model with recommendation*, arXiv preprint arXiv:2305.00447, (2023).

[7] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, science, 286 (1999), pp. 509–512.

[8] S. Barocas and A. D. Selbst, *Big data's disparate impact*, California law review, (2016), pp. 671–732.

[9] C. D. Barros, M. R. Mendonça, A. B. Vieira, and A. Ziviani, *A survey on embedding dynamic graphs*, ACM Computing Surveys (CSUR), 55 (2021), pp. 1–37.

[10] C. Basu, H. Hirsh, W. Cohen, et al., *Recommendation as classification: Using social and content-based information in recommendation*, in Aaai/iaai, 1998, pp. 714–720.

[11] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, et al., *Graph of thoughts: Solving elaborate problems with large language models*, arXiv preprint arXiv:2308.09687, (2023).

[12] L. Brillouin, *Science and information theory*, Courier Corporation, 2013.

[13] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, *On identifiability in transformers*, in International Conference on Learning Representations, 2019.

[14] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, and Y. Li, *Sequential recommendation with graph neural networks*, in Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, 2021, pp. 378–387.

[15] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, *An attentive survey of attention models*, arXiv preprint arXiv:1904.02874, (2019).

[16] C. Chen, M. Zhang, Y. Liu, and S. Ma, *Neural attentional rating regression with review-level explanations*, in Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1583–1592.

[17] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, *Large-scale interactive recommendation with tree-structured policy gradient*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3312–3320.

[18] H. Chen, Y. Li, X. Sun, G. Xu, and H. Yin, *Temporal meta-path guided explainable recommendation*, in Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 1056–1064.

[19] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, *Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention*, in Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, 2017, pp. 335–344.

[20] J. Chen, F. Zhuang, X. Hong, X. Ao, X. Xie, and Q. He, *Attention-driven factor model for explainable personalized recommendation*, in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 909–912.

[21] X. Chen, Y. Zhou, D. Wu, W. Zhang, Y. Zhou, B. Li, and W. Wang, *Imagine by reasoning: A reasoning-based implicit semantic data augmentation for long-tailed classification*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 356–364.

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

[23] Z. Chen, H. Mao, H. Li, W. Jin, H. Wen, X. Wei, S. Wang, D. Yin, W. Fan, H. Liu, et al., *Exploring the potential of large language models (llms) in learning on graphs*, arXiv preprint arXiv:2307.03393, (2023).

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

[25] K. Choi, D. Yoo, G. Kim, and Y. Suh, *A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis*, electronic commerce research and applications, 11 (2012), pp. 309–317.

[26] D. Cong, Y. Zhao, B. Qin, Y. Han, M. Zhang, A. Liu, and N. Chen, *Hierarchical attention based neural network for explainable recommendation*, in Proceedings of the 2019 on International Conference on Multimedia Retrieval, 2019, pp. 373–381.

[27] E. Dai and S. Wang, *Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information*, in Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 680–688.

[28] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, *Go for a walk and arrive at the*

*answer: Reasoning over paths in knowledge bases using reinforcement learning*, in ICLR, 2018.

[29] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, *Fairness through awareness*, in Proceedings of the 3rd innovations in theoretical computer science conference, 2012, pp. 214–226.

[30] A. M. Elkahky, Y. Song, and X. He, *A multi-view deep learning approach for cross domain user modeling in recommendation systems*, in Proceedings of the 24th international conference on world wide web, 2015, pp. 278–288.

[31] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, *Graph neural networks for social recommendation*, in The World Wide Web Conference, 2019, pp. 417–426.

[32] W. Fan, Z. Zhao, J. Li, Y. Liu, X. Mei, Y. Wang, J. Tang, and Q. Li, *Recommender systems in the era of large language models (llms)*, arXiv preprint arXiv:2307.02046, (2023).

[33] Z. Fan, Z. Liu, Y. Wang, A. Wang, Z. Nazari, L. Zheng, H. Peng, and P. S. Yu, *Sequential recommendation via stochastic self-attention*, in Proceedings of the ACM Web Conference 2022, 2022, pp. 2036–2047.

[34] S. Feng, L. V. Tran, G. Cong, L. Chen, J. Li, and F. Li, *Hme: A hyperbolic metric embedding approach for next-poi recommendation*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1429–1438.

[35] M. Fu, H. Qu, Z. Yi, L. Lu, and Y. Liu, *A novel deep learning-based collaborative filtering model for recommendation system*, IEEE transactions on cybernetics, 49 (2018), pp. 1084–1096.

[36] Z. Fu, Y. Xian, R. Gao, J. Zhao, Q. Huang, Y. Ge, S. Xu, S. Geng, C. Shah, Y. Zhang, et al., *Fairness-aware explainable recommendation over knowledge graphs*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 69–78.

[37] J. Gao, X. Wang, Y. Wang, and X. Xie, *Explainable recommendation through attentive multi-view learning*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3622–3629.

[38] J. GAO, T. ZHANG, AND C. XU, *A unified personalized video recommendation via dynamic recurrent neural networks*, in Proceedings of the 25th ACM international conference on Multimedia, 2017, pp. 127–135.

[39] 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*, arXiv preprint arXiv:2204.11159, (2022).

[40] S. GENG, Z. FU, J. TAN, Y. GE, G. DE MELO, AND Y. ZHANG, *Path language modeling over knowledge graphsfor explainable recommendation*, in Proceedings of the ACM Web Conference 2022, 2022, pp. 946–955.

[41] A. GHAZIMATIN, O. BALALAU, R. SAHA ROY, AND G. WEIKUM, *Prince: Provider-side interpretability with counterfactual explanations in recommender systems*, in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 196–204.

[42] A. GHAZIMATIN, S. PRAMANIK, R. SAHA ROY, AND G. WEIKUM, *Elixir: learning from user feedback on explanations to improve recommender models*, in Proceedings of the Web Conference 2021, 2021, pp. 3850–3860.

[43] C. GRIMSLEY, E. MAYFIELD, AND J. BURSTEN, *Why attention is not explanation: Surgical intervention and causal reasoning about neural models*, (2020).

[44] Q. GUO, Z. SUN, J. ZHANG, AND Y.-L. THENG, *An attentional recurrent neural network for personalized next location recommendation*, in Proceedings of the AAAI Conference on artificial intelligence, vol. 34, 2020, pp. 83–90.

[45] W. HAMILTON, Z. YING, AND J. LESKOVEC, *Inductive representation learning on large graphs*, in NIPS, 2017.

[46] X. HAN, C. SHI, S. WANG, S. Y. PHILIP, AND L. SONG, *Aspect-level deep collaborative filtering via heterogeneous information networks.*, in IJCAI, 2018, pp. 3393–3399.

[47] N. HARIRI, B. MOBASHER, AND R. BURKE, *Context-aware music recommendation based on latenttopic sequential patterns*, in Proceedings of the sixth ACM conference on Recommender systems, 2012, pp. 131–138.

[48] J. HARTE, W. ZORGDRAGER, P. LOURIDAS, A. KATSIFODIMOS, D. JANNACH, AND M. FRAGKOULIS, *Leveraging large language models for sequential recommendation*, in Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1096–1102.

[49] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[50] X. HE, T. CHEN, M.-Y. KAN, AND X. CHEN, *Trirank: Review-aware explainable recommendation by modeling aspects*, in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, 2015, pp. 1661–1670.

[51] X. HE AND T.-S. CHUA, *Neural factorization machines for sparse predictive analytics*, in Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, 2017, pp. 355–364.

[52] 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, 2020, pp. 639–648.

[53] 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, 2017, pp. 173–182.

[54] Z. HE, Z. XIE, R. JHA, H. STECK, D. LIANG, Y. FENG, B. P. MAJUMDER, N. KALLUS, AND J. MCAULEY, *Large language models as zero-shot conversational recommenders*, in Proceedings of the 32nd ACM international conference on information and knowledge management, 2023, pp. 720–730.

[55] R. HECKEL, M. VLACHOS, T. PARNELL, AND C. DÜNNER, *Scalable and interpretable product recommendations via overlapping co-clustering*, in 2017 IEEE 33rd International Conference on Data Engineering (ICDE), IEEE, 2017, pp. 1033–1044.

[56] B. HIDASI AND A. KARATZOGLOU, *Recurrent neural networks with top-k gains for session-based recommendations*, in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 843–852.

[57] B. HIDASI, A. KARATZOGLOU, L. BALTRUNAS, AND D. TIKK, *Session-based recommendations with recurrent neural networks*, arXiv preprint arXiv:1511.06939, (2015).

[58] B. HU, C. SHI, W. X. ZHAO, AND P. S. YU, *Leveraging meta-path based context for top-n recommendation with a neural co-attention model*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1531–1540.

[59] S. HUA, W. CHEN, Z. LI, P. ZHAO, AND L. ZHAO, *Path-based academic paper recommendation*, in Web Information Systems Engineering–WISE 2020: 21st International Conference, Amsterdam, The Netherlands, October 20–24, 2020, Proceedings, Part II 21, Springer, 2020, pp. 343–356.

[60] W. HUA, L. LI, S. XU, L. CHEN, AND Y. ZHANG, *Tutorial on large language models for recommendation*, in Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1281–1283.

[61] X. HUANG, Q. FANG, S. QIAN, J. SANG, Y. LI, AND C. XU, *Explainable interaction-driven user modeling over knowledge graph for sequential recommendation*, in Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 548–556.

[62] X. HUANG, J. LIAN, Y. LEI, J. YAO, D. LIAN, AND X. XIE, *Recommender ai agent: Integrating large language models for interactive recommendations*, arXiv preprint arXiv:2308.16505, (2023).

[63] B. HUI, L. ZHANG, X. ZHOU, X. WEN, AND Y. NIAN, *Personalized recommendation system based on knowledge embedding and historical behavior*, Applied Intelligence, (2022), pp. 1–13.

[64] S. JAIN AND B. C. WALLACE, *Attention is not explanation*, arXiv preprint arXiv:1902.10186, (2019).

[65] M. JAMALI AND M. ESTER, *A matrix factorization technique with trust propagation for recommendation in social networks*, in Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 135–142.

[66] B. JIN, C. GAO, X. HE, D. JIN, AND Y. LI, *Multi-behavior recommendation with graph convolutional networks*, in Proceedings of the 43rd International ACM

SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 659–668.

[67] V. KAFFES, D. SACHARIDIS, AND G. GIANNOPOULOS, *Model-agnostic counterfactual explanations of recommendations*, in Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization, 2021, pp. 280–285.

[68] W.-C. KANG AND J. MCAULEY, *Self-attentive sequential recommendation*, in ICDM, IEEE, 2018, pp. 197–206.

[69] T. N. KIPF AND M. WELLING, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907, (2016).

[70] J. KLEINBERG, J. LUDWIG, S. MULLAINATHAN, AND A. RAMBACHAN, *Algorithmic fairness*, in Aea papers and proceedings, vol. 108, American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203, 2018, pp. 22–27.

[71] I. KONSTAS, V. STATHOPOULOS, AND J. M. JOSE, *On social networks and collaborative recommendation*, in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, 2009, pp. 195–202.

[72] Ö. D. KÖSE AND Y. SHEN, *Fairness-aware node representation learning*, arXiv preprint arXiv:2106.05391, (2021).

[73] K. LANG, *Newsweeder: Learning to filter netnews*, in Machine Learning Proceedings 1995, Elsevier, 1995, pp. 331–339.

[74] C.-T. LI, C. HSU, AND Y. ZHANG, *Fairsr: Fairness-aware sequential recommendation through multi-task learning with preference graph embeddings*, ACM Transactions on Intelligent Systems and Technology (TIST), 13 (2022), pp. 1–21.

[75] L. LI, Y. ZHANG, D. LIU, AND L. CHEN, *Large language models for generative recommendation: A survey and visionary discussions*, arXiv preprint arXiv:2309.01157, (2023).

[76] Y. LI, H. CHEN, Y. LI, L. LI, S. Y. PHILIP, AND G. XU, *Reinforcement learning based path exploration for sequential explainable recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2023).

[77] Y. Li, H. Chen, X. Sun, Z. Sun, L. Li, L. Cui, P. S. Yu, and G. Xu, *Hyperbolic hypergraphs for sequential recommendation*, arXiv preprint arXiv:2108.08134, (2021).

[78] Y. Li, M. Liu, J. Yin, C. Cui, X.-S. Xu, and L. Nie, *Routing micro-videos via a temporal graph-guided recommendation system*, in Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1464–1472.

[79] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, and E. Chen, *Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors*, in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 1734–1743.

[80] T.-P. Liang, H.-J. Lai, and Y.-C. Ku, *Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings*, Journal of Management Information Systems, 23 (2006), pp. 45–70.

[81] C.-S. Lin, C.-N. Tsai, S.-T. Su, J.-S. Jwo, C.-H. Lee, and X. Wang, *Predictive prompts with joint training of large language models for explainable recommendation*, Mathematics, 11 (2023), p. 4230.

[82] C.-Y. Lin, *Rouge: A package for automatic evaluation of summaries*, in Text summarization branches out, 2004, pp. 74–81.

[83] D.-R. Liu, C.-H. Lai, and W.-J. Lee, *A hybrid of sequential rules and collaborative filtering for product recommendation*, Information Sciences, 179 (2009), pp. 3505–3519.

[84] P. Liu, L. Zhang, and J. A. Gulla, *Dynamic attention-based explainable recommendation with textual and visual fusion*, Information Processing & Management, 57 (2020), p. 102099.

[85] S. Liu and Z. Chen, *Sequential behavior modeling for next micro-video recommendation with collaborative transformer*, in 2019 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2019, pp. 460–465.

[86] Z. Liu, T.-K. Nguyen, and Y. Fang, *Tail-gnn: Tail-node graph neural networks*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1109–1119.

[87] Z. LIU, T.-K. NGUYEN, AND Y. FANG, *On generalized degree fairness in graph neural networks*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 4525–4533.

[88] S. LUO, C. MA, Y. XIAO, AND L. SONG, *Improving long-tail item recommendation with graph augmentation*, in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 1707–1716.

[89] H. MA, H. YANG, M. R. LYU, AND I. KING, *Sorec: social recommendation using probabilistic matrix factorization*, in Proceedings of the 17th ACM conference on Information and knowledge management, 2008, pp. 931–940.

[90] T. MA, L. HUANG, Q. LU, AND S. HU, *Kr-gcn: Knowledge-aware reasoning with graph convolution network for explainable recommendation*, ACM Transactions on Information Systems (TOIS), (2022).

[91] W. MA, M. ZHANG, Y. CAO, W. JIN, C. WANG, Y. LIU, S. MA, AND X. REN, *Jointly learning explainable rules for recommendation with knowledge graph*, in The world wide web conference, 2019, pp. 1210–1221.

[92] T. MIKOLOV, K. CHEN, G. CORRADO, AND J. DEAN, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781, (2013).

[93] C. MOLNAR, *Interpretable machine learning*, Lulu. com, 2020.

[94] J. MULLENBACH, S. WIEGREFFE, J. DUKE, J. SUN, AND J. EISENSTEIN, *Explainable prediction of medical codes from clinical text*, arXiv preprint arXiv:1802.05695, (2018).

[95] M. NAUMOV, D. MUDIGERE, H.-J. M. SHI, J. HUANG, N. SUNDARAMAN, J. PARK, X. WANG, U. GUPTA, C.-J. WU, A. G. AZZOLINI, ET AL., *Deep learning recommendation model for personalization and recommendation systems*, arXiv preprint arXiv:1906.00091, (2019).

[96] J. NI, J. LI, AND J. MCAULEY, *Justifying recommendations using distantly-labeled reviews and fine-grained aspects*, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 188–197.

[97] M. NICKEL AND D. KIELA, *Poincaré embeddings for learning hierarchical representations*, Advances in neural information processing systems, 30 (2017).

[98] M. O'MAHONY, N. HURLEY, N. KUSHMERICK, AND G. SILVESTRE, *Collaborative recommendation: A robustness analysis*, ACM Transactions on Internet Technology (TOIT), 4 (2004), pp. 344–377.

[99] G. PANAGOPOULOS, G. NIKOLENTZOS, AND M. VAZIRGIANNIS, *Transfer graph neural networks for pandemic forecasting*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 4838–4845.

[100] K. PAPINENI, S. ROUKOS, T. WARD, AND W.-J. ZHU, *Bleu: a method for automatic evaluation of machine translation*, in Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[101] A. PAREJA, G. DOMENICONI, J. CHEN, T. MA, T. SUZUMURA, H. KANEZASHI, T. KALER, T. SCHARDL, AND C. LEISERSON, *Evolvegcn: Evolving graph convolutional networks for dynamic graphs*, in Proceedings of the AAAI conference on artificial intelligence, vol. 34, 2020, pp. 5363–5370.

[102] H. PARK, H. JEON, J. KIM, B. AHN, AND U. KANG, *Uniwalk: Explainable and accurate recommendation for rating and network data*, arXiv preprint arXiv:1710.07134, (2017).

[103] M. J. PAZZANI AND D. BILLSUS, *Content-based recommendation systems*, in The adaptive web: methods and strategies of web personalization, Springer, 2007, pp. 325–341.

[104] M. J. PAZZANI, J. MURAMATSU, D. BILLSUS, ET AL., *Syskill & webert: Identifying interesting web sites*, in AAAI/IAAI, Vol. 1, 1996, pp. 54–61.

[105] G. PEAKE AND J. WANG, *Explanation mining: Post hoc interpretability of latent factor models for recommendation systems*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2060–2069.

[106] J. PEARL, *Causality*, Cambridge university press, 2009.

[107] B. L. PEREIRA, A. UEDA, G. PENHA, R. L. SANTOS, AND N. ZIVIANI, *Online learning to rank for sequential music recommendation*, in Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 237–245.

[108] B. Perozzi, R. Al-Rfou, and S. Skiena, *Deepwalk: Online learning of social representations*, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[109] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, *Explainability methods for graph convolutional neural networks*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 10772–10781.

[110] R. Qiu, Z. Huang, H. Yin, and Z. Wang, *Contrastive learning for representation degeneration problem in sequential recommendation*, in Proceedings of the fifteenth ACM international conference on web search and data mining, 2022, pp. 813–823.

[111] M. Raghavan, *The Societal Impacts of Algorithmic Decision-Making*, PhD thesis, Cornell University, 2021.

[112] N. Ranjbar, S. Momtazi, and M. Homayounpour, *Explaining recommendation system using counterfactual textual explanations*, arXiv preprint arXiv:2303.11160, (2023).

[113] G. B. Robinson, *Automated collaborative filtering system*, Aug. 4 1998. US Patent 5,790,426.

[114] M. Saebi, S. Krieg, C. Zhang, M. Jiang, and N. Chawla, *Heterogeneous relational reasoning in knowledge graphs with reinforcement learning*, arXiv preprint arXiv:2003.06050, (2020).

[115] M. Salehi, *An effective recommendation based on user behaviour: a hybrid of sequential pattern of user and attributes of product*, International Journal of Business Information Systems, 14 (2013), pp. 480–496.

[116] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, *Dysat: Deep neural representation learning on dynamic graphs via self-attention networks*, in Proceedings of the 13th international conference on web search and data mining, 2020, pp. 519–527.

[117] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Item-based collaborative filtering recommendation algorithms*, in Proceedings of the 10th international conference on World Wide Web, 2001, pp. 285–295.

[118] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, *Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system*, in Proceedings of the 1998 ACM conference on Computer supported cooperative work, 1998, pp. 345–354.

[119] S. Serrano and N. A. Smith, *Is attention interpretable?*, arXiv preprint arXiv:1906.03731, (2019).

[120] X. Sha, Z. Sun, and J. Zhang, *Hierarchical attentive knowledge graph embedding for personalized recommendation*, Electronic Commerce Research and Applications, 48 (2021), p. 101071.

[121] G. Shani, D. Heckerman, R. I. Brafman, and C. Boutilier, *An mdp-based recommender system.*, Journal of Machine Learning Research, 6 (2005).

[122] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu, *Semantic path based personalized recommendation on weighted heterogeneous information networks*, in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, 2015, pp. 453–462.

[123] W. Song, Y. Dong, N. Liu, and J. Li, *Guide: Group equality informed individual fairness in graph neural networks*, in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1625–1634.

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

[125] K. Sun, Y. E. Xu, H. Zha, Y. Liu, and X. L. Dong, *Head-to-tail: How knowledgeable are large language models (llm)? aka will llms replace knowledge graphs?*, arXiv preprint arXiv:2308.10168, (2023).

[126] X. Sun, H. Cheng, J. Li, B. Liu, and J. Guan, *All in One: Multi-Task Prompting for Graph Neural Networks*, in KDD, 2023, pp. 2120–2131.

[127] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, *Pathsim: Meta path-based top-k similarity search in heterogeneous information networks*, Proceedings of the VLDB Endowment, 4 (2011), pp. 992–1003.

[128] O. Tal, Y. Liu, J. Huang, X. Yu, and B. Aljbawi, *Neural attention frameworks for explainable recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2019).

[129] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang, *Counterfactual explainable recommendation*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1784–1793.

[130] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, *Line: Large-scale information network embedding*, in Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[131] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang, *Investigating and mitigating degree-related biases in graph convoltuional networks*, in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1435–1444.

[132] Z. Tang, Y. Chen, Y. Liu, and K. Zhang, *Tier balancing: Towards dynamic fairness over underlying causal factors*, in International Conference on Learning Representations (ICLR), 2023.

[133] G. Trichopoulos, M. Konstantakis, G. Alexandridis, and G. Caridakis, *Large language models as recommendation systems in museums*, Electronics, 12 (2023), p. 3829.

[134] L. H. Ungar and D. P. Foster, *Clustering methods for collaborative filtering*, in AAAI workshop on recommendation systems, vol. 1, Menlo Park, CA, 1998, pp. 114–129.

[135] A. Van den Oord, S. Dieleman, and B. Schrauwen, *Deep content-based music recommendation*, Advances in neural information processing systems, 26 (2013).

[136] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in Advances in neural information processing systems, 2017, pp. 5998–6008.

[137] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, *Graph attention networks*, arXiv, (2017).

[138] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, in International Conference on Learning Representations, 2018.

[139] S. Verma, J. Dickerson, and K. Hines, *Counterfactual explanations for machine learning: A review*, arXiv preprint arXiv:2010.10596, (2020).

[140] M. Wan and J. J. McAuley, *Item recommendation on monotonic behavior chains*, in Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018, ACM, 2018, pp. 86–94.

[141] M. Wan, R. Misra, N. Nakashole, and J. J. McAuley, *Fine-grained spoiler detection from large-scale review corpora*, in Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, Association for Computational Linguistics, 2019, pp. 2605–2610.

[142] C. Wang, M. Zhang, W. Ma, Y. Liu, and S. Ma, *Make it a chorus: knowledge-and time-aware item modeling for sequential recommendation*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 109–118.

[143] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, *Ripplenet: Propagating user preferences on the knowledge graph for recommender systems*, in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 417–426.

[144] H. Wang, K. Zhou, X. Zhao, J. Wang, and J.-R. Wen, *Curriculum pre-training heterogeneous subgraph transformer for top-n recommendation*, ACM Transactions on Information Systems, 41 (2023), pp. 1–28.

[145] N. Wang, H. Wang, Y. Jia, and Y. Yin, *Explainable recommendation via multi-task learning in opinionated text data*, in The 41st international ACM SIGIR conference on research & development in information retrieval, 2018, pp. 165–174.

[146] R. Wang, X. Wang, C. Shi, and L. Song, *Uncovering the structural fairness in graph contrastive learning*, Advances in Neural Information Processing Systems, 35 (2022), pp. 32465–32473.

[147] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, *Kgat: Knowledge graph attention network for recommendation*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.

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

[149] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, *Explainable reasoning over knowledge graphs for recommendation*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 5329–5336.

[150] X. Wang, Y. Wang, and Y. Ling, *Attention-guide walk model in heterogeneous information network for multi-style recommendation explanation*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 6275–6282.

[151] Y. Wang, Y. Zhao, Y. Dong, H. Chen, J. Li, and T. Derr, *Improving fairness in graph neural networks via mitigating sensitive attribute leakage*, in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1938–1948.

[152] Z. Wang, J. Zhang, H. Xu, X. Chen, Y. Zhang, W. X. Zhao, and J.-R. Wen, *Counterfactual data-augmented sequential recommendation*, in Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 347–356.

[153] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, *Llmrec: Large language models with graph augmentation for recommendation*, arXiv preprint arXiv:2311.00423, (2023).

[154] S. Wiegreffe and Y. Pinter, *Attention is not not explanation*, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 11–20.

[155] R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Machine learning, 8 (1992), pp. 229–256.

[156] H. Wu, C. Geng, and H. Fang, *Causality and correlation graph modeling for effective and explainable session-based recommendation*, ACM Transactions on the Web, 18 (2023), pp. 1–25.

[157] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, *Session-based recommendation with graph neural networks*, in AAAI, vol. 33, 2019, pp. 346–353.

[158] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, *Random walks: A review of algorithms and applications*, IEEE Transactions on Emerging Topics in Computational Intelligence, 4 (2019), pp. 95–107.

[159] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, and Y. Zhang, *Reinforcement knowledge graph reasoning for explainable recommendation*, in Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 285–294.

[160] L. Xie, Z. Hu, X. Cai, W. Zhang, and J. Chen, *Explainable recommendation based on knowledge graph and multi-objective optimization*, Complex & Intelligent Systems, 7 (2021), pp. 1241–1252.

[161] Z. Xie, C. Liu, Y. Zhang, H. Lu, D. Wang, and Y. Ding, *Adversarial and contrastive variational autoencoder for sequential recommendation*, in Proceedings of the Web Conference 2021, 2021, pp. 449–459.

[162] K. Xiong, W. Ye, X. Chen, Y. Zhang, W. X. Zhao, B. Hu, Z. Zhang, and J. Zhou, *Counterfactual review-based recommendation*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 2231–2240.

[163] W. Xiong, T. Hoang, and W. Y. Wang, *Deeppath: A reinforcement learning method for knowledge graph reasoning*, in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017), Copenhagen, Denmark, September 2017, ACL.

[164] C. Xu, P. Zhao, Y. Liu, J. Xu, V. S. S. S. Sheng, Z. Cui, X. Zhou, and H. Xiong, *Recurrent convolutional neural network for sequential recommendation*, in The World Wide Web Conference, 2019, pp. 3398–3404.

[165] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, *Show, attend and tell: Neural image caption generation with*

*visual attention*, in International conference on machine learning, PMLR, 2015, pp. 2048–2057.

[166] K. YANG AND J. STOYANOVICH, *Measuring fairness in ranked outputs*, in Proceedings of the 29th international conference on scientific and statistical database management, 2017, pp. 1–6.

[167] Y. YANG, J. CAO, M. STOJMENOVIC, S. WANG, Y. CHENG, C. LUM, AND Z. LI, *Time-capturing dynamic graph embedding for temporal linkage evolution*, IEEE Transactions on Knowledge and Data Engineering, 35 (2021), pp. 958–971.

[168] Y. YANG, H. YIN, J. CAO, T. CHEN, Q. V. H. NGUYEN, X. ZHOU, AND L. CHEN, *Time-aware dynamic graph embedding for asynchronous structural evolution*, IEEE Transactions on Knowledge and Data Engineering, (2023).

[169] H. YIN, B. CUI, J. LI, J. YAO, AND C. CHEN, *Challenging the long tail recommendation*, arXiv preprint arXiv:1205.6700, (2012).

[170] J. YIN, C. LIU, W. WANG, J. SUN, AND S. C. HOI, *Learning transferrable parameters for long-tailed sequential user behavior modeling*, in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 359–367.

[171] J. YOU, T. DU, AND J. LESKOVEC, *Roland: graph learning framework for dynamic graphs*, in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2358–2366.

[172] J. YU, H. YIN, X. XIA, T. CHEN, L. CUI, AND Q. V. H. NGUYEN, *Are graph augmentations necessary? simple graph contrastive learning for recommendation*, in Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval, 2022, pp. 1294–1303.

[173] X. YU, X. REN, Y. SUN, Q. GU, B. STURT, U. KHANDELWAL, B. NORICK, AND J. HAN, *Personalized entity recommendation: A heterogeneous information network approach*, in Proceedings of the 7th ACM international conference on Web search and data mining, 2014, pp. 283–292.

[174] Q. YUAN, L. CHEN, AND S. ZHAO, *Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation*, in Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 245–252.

[175] J. ZHANG, R. XIE, Y. HOU, W. X. ZHAO, L. LIN, AND J.-R. WEN, *Recommendation as instruction following: A large language model empowered recommendation approach*, arXiv preprint arXiv:2305.07001, (2023).

[176] M. ZHANG, S. WU, X. YU, Q. LIU, AND L. WANG, *Dynamic graph neural networks for sequential recommendation*, IEEE Transactions on Knowledge and Data Engineering, 35 (2022), pp. 4741–4753.

[177] S. ZHANG, L. YAO, A. SUN, AND Y. TAY, *Deep learning based recommender system: A survey and new perspectives*, ACM computing surveys (CSUR), 52 (2019), pp. 1–38.

[178] Y. ZHANG, Q. AI, X. CHEN, AND W. B. CROFT, *Joint representation learning for top-n recommendation with heterogeneous information sources*, in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1449–1458.

[179] Y. ZHANG AND X. CHEN, *Explainable recommendation: A survey and new perspectives*, arXiv preprint arXiv:1804.11192, (2018).

[180] Y. ZHANG, X. CHEN, ET AL., *Explainable recommendation: A survey and new perspectives*, Foundations and Trends® in Information Retrieval, 14 (2020), pp. 1–101.

[181] Y. ZHANG, F. FENG, J. ZHANG, K. BAO, Q. WANG, AND X. HE, *Collm: Integrating collaborative embeddings into large language models for recommendation*, arXiv preprint arXiv:2310.19488, (2023).

[182] Y. ZHANG, G. LAI, M. ZHANG, Y. ZHANG, Y. LIU, AND S. MA, *Explicit factor models for explainable recommendation based on phrase-level sentiment analysis*, in Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 2014, pp. 83–92.

[183] H. ZHAO, S. GUO, AND Y. LIN, *Hierarchical classification of data with long-tailed distributions via global and local granulation*, Information Sciences, 581 (2021), pp. 536–552.

[184] K. ZHAO, X. WANG, Y. ZHANG, L. ZHAO, Z. LIU, C. XING, AND X. XIE, *Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 239–248.

[185] P. ZHAO, A. LUO, Y. LIU, F. ZHUANG, J. XU, Z. LI, V. S. SHENG, AND X. ZHOU, *Where to go next: A spatio-temporal gated network for next poi recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2020).

[186] Q. ZHAO, Z. WU, Z. ZHANG, AND J. ZHOU, *Long-tail augmented graph contrastive learning for recommendation*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023, pp. 387–403.

[187] S. ZHAO, T. ZHAO, I. KING, AND M. R. LYU, *Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation*, in Proceedings of the 26th international conference on world wide web companion, 2017, pp. 153–162.

[188] W. X. ZHAO, S. MU, Y. HOU, Z. LIN, Y. CHEN, X. PAN, K. LI, Y. LU, H. WANG, C. TIAN, ET AL., *Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021.

[189] Y. ZHAO, X. WANG, J. CHEN, Y. WANG, W. TANG, X. HE, AND H. XIE, *Time-aware path reasoning on knowledge graph for recommendation*, ACM Transactions on Information Systems, 41 (2022), pp. 1–26.

[190] K. ZHOU, H. WANG, W. X. ZHAO, Y. ZHU, S. WANG, F. ZHANG, Z. WANG, AND J.-R. WEN, *S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization*, in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1893–1902.

[191] Y. ZHOU, H. WANG, J. HE, AND H. WANG, *From intrinsic to counterfactual: On the explainability of contextualized recommender systems*, arXiv preprint arXiv:2110.14844, (2021).

[192] Q. ZHU, X. ZHOU, J. WU, J. TAN, AND L. GUO, *A knowledge-aware attentional reasoning network for recommendation.*, in AAAI, 2020, pp. 6999–7006.

[193] Y. ZHU, Y. GONG, Q. LIU, Y. MA, W. OU, J. ZHU, B. WANG, Z. GUAN, AND D. CAI, *Query-based interactive recommendation by meta-path and adapted attention-gru*, in Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2585–2593.