Towards Causality-centric Recommender Systems

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

Doctor of Philosophy
in
Analytics

by

Xiangmeng Wang

to

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

Feb 2025

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Xiangmeng Wang, 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.

SIGNATURE: Production Note:
Signature removed prior to publication.

DATE: 3rd Feb, 2025

Abstract

With the exponential growth of information, Recommendation Systems (RecSys) have become pivotal tools in tackling data overload, for information provision in applications across academia and industry. New theories, methodologies, and applications of RecSys have been studied and developed at a rapid pace. These innovations largely rely on correlation learning, such as collaborative filtering, to model behavioral correlations and predict user preferences. However, as demonstrated by ample evidences, user behavior is a mixed reflection of user interests, driven by various factors. Thus, it is crucial to go beyond correlation learning to model the true interest, therefore improving the recommendation performance. This thesis aims to incorporate causal learning theory into recommendations. Causal learning explores causal relations in which a cause event (e.g., user interests) entails the occurrence of an effect event (e.g., user behavior). A cause-and-effect relationship is stronger than a correlation between events, and therefore, aggregating causal relations extracted from large corpora can be used in numerous recommendation applications to produce superior results than traditional approaches. In particular, this thesis first investigates existing challenging tasks encountered in recommendation systems from data, model, robustness, and trustworthy perspectives, e.g., data bias, model accuracy, model explainability, and model fairness. Then it addresses these challenges by leveraging causal approaches to design feasible solutions for each. Through comprehensive experiments, these proposed causality-centric solutions demonstrate their significant superiority over correlation-based recommendation methods, in terms of higher accuracy, improved explainability, enhanced fairness, and better generalization ability.

Dedication

I gritted my teeth and persevered through a long journey to finally complete this doctoral thesis. This year, at my 28 years of age, after more than 20 years of studying, it seems like this period has been the purest and most precious golden era of my life. In this journey, I have experienced loneliness, confusion and breakdowns, but also moments of ecstasy, touching, and countless experiences of self-redemption and mutual support with others. Yet, this acknowledgment can only capture a small fraction of those experiences. Looking back, all I feel is gratitude.

The first person I want to thank is myself. Ever since high school, I dreamed of studying abroad to experience different cultures and hear diverse voices, but the reality did not support my ambitions. I didn't perform well on the college entrance exam and ended up attending a second-tier university. But fortunately, I never gave up on the idea of pursuing a Ph.D., so throughout my four years of college, I spent every single day striving toward the goal of being recommended for the master's degree. Luckily, life did not let me down, and I successfully earned a spot to pursue my master's degree at Shanghai University. Later, I smoothly transitioned into a Ph.D. program and with dreams in my heart, I came to Sydney. The flight on February 22, 2022 holds a special meaning to me.

I am so lucky to have met my Ph.D. supervisor, Prof. Guandong Xu, who guided me through every challenge I faced during my Ph.D. journey. Being a Ph.D. student is tough, but Prof. Xu consistently provided thoughtful and detailed advice whenever I encountered difficulties. Not only did he offer practical guidance for my research and create a supportive environment, but also taught me how to be a person of integrity, honesty, and rigor. I could not have completed my Ph.D. without his guidance and support, and Prof. Xu will remain a lifelong role model for me.

I thank Dr. Qian Li for providing invaluable companionship and guidance throughout my paper submissions. Dr. Qian Li has exceptional research abilities and consistently generates new ideas quickly. We often discussed our work late into the night, and Qian always demonstrated immense patience and care. Without her thoughtful guidance, I would not have been able to start my research so quickly or produce my papers. Although I have never met Qian in person for various reasons, she will always hold a special place

in my heart. I hope one day I can offer my students the same valuable advice and support that Qian gave me.

I also want to express my deep love and gratitude to my mother, the strongest and most resilient woman I have ever known. She started her own business back in the 1990s, during a challenging economic recession in China, even without family support. Despite all the hardships and setbacks, my mom remained strong and never gave up. She has been a constant role model for me, demonstrating the true power of perseverance. My mom has provided me with unconditional love and care, going above and beyond in every way imaginable. She is gentle yet incredibly powerful, and I feel so fortunate to have her as my mother. Mom, I wish for you to always be happy and true to yourself, beyond just being my mother. I miss my grandmother deeply, who passed away in July 2023. She raised me up, and I will never forget the moment she passed away in my arms. I only wish I could have done more when she was still with me.

I want to express my gratitude to my past relationships. One taught me love, another taught me patience, and yet another taught me pain. I am now grateful for these experiences, as they have contributed to my growth. I recognize that I have loved and lost, but that is not all I see. I also want to thank Mr. Zheping Ren for his unconditional love. Even though he knows I am not perfect, he has always been patient and attentive to every detail of my daily life.

I would also like to extend my heartfelt thanks to my colleagues, including Dr. Dawei Xu, En Yu, Haoran Yang, Yicong Li, Kaize Shi, Yakun Chen, Dianer Yu, Sirui Huang, and Wei Huang. Their support and collaboration have been invaluable throughout my journey. Additionally, I am deeply grateful to my friends from the University of New South Wales, including Dr. Jiehong Li, Penghao Zhang, and Hongzhe Chen, for their encouragement and companionship. A special thanks goes to my cats, BeiBei, Nancy, and Sisley, for their natural love and companionship.

I want to celebrate and cheer for all the female researchers around the world. Femininity is one of the most powerful gifts we are blessed with, and it drives us to excel, create, and inspire. Last but not least, I would like to extend my deepest gratitude to Judea Pearl and Donald Rubin for their groundbreaking contributions to the field of causal inference. Their work has been a guiding force throughout my exploration of this valuable and promising research domain. Without their insights, my journey in this field would not have been the same.

Even now, I still feel a sense of fear and uncertainty about my future. I aspire to be a light for others and contribute my small yet meaningful value to the research community. I hope to continue growing, becoming a better version of myself, and perhaps even a role model for others in the future. The final words for myself:

“博学之，审问之，慎思之，明辨之，笃行之。”

As Tagore wrote in Fireflies, “My last tribute is to those who know I am not perfect but still love me.” This sentiment reflects my deepest gratitude to those who have stood by me, embracing my imperfections while offering unwavering and support. I will always strive to be “**Gentle, Firm, and Powerful**.” and I will never give up on this aspiration.

List of Publications

Journal Papers :

1. **X. Wang**, Q. Li, D. Yu, P. Cui, Z. Wang and G. Xu. Causal Disentanglement for Semantics-Aware Intent Learning in Recommendation. In IEEE Transactions on Knowledge and Data Engineering **(TKDE) (CORE A*, CCF A)**

2. **X. Wang**, Q. Li, D. Yu, Q. Li and G. Xu. Reinforced Path Reasoning for Counterfactual Explainable Recommendation. In IEEE Transactions on Knowledge and Data Engineering **(TKDE) (CORE A*, CCF A)**

3. **X. Wang**, Q. Li, D. Yu, Q. Li and G. Xu. Counterfactual Explanation for Fairness in Recommendation. In ACM Transactions on Information Systems **(TOIS) (CORE A*, CCF A)**

4. **X. Wang**, Q. Li, D. Yu, W. Huang, Q. Li and G. Xu. Neural Causal Graph Collaborative Filtering. In Information Sciences **(INS) (CORE A, CCF B)**

5. Q. Li*, **X. Wang*** **(Equal contribution)**, Z. Wang and G. Xu. Be Causal: Debiasing Social Network Confounding in Recommendation. In ACM Transactions on Knowledge Discovery from Data **(TKDD) (JCR-Q1, CCF B)**

6. **X. Wang**, Q. Li, D. Yu, Q. Li and G. Xu. Constrained Off-policy Learning over Heterogeneous Information for Fairness-aware Recommendation. In ACM Transactions on Recommender Systems **(TORS)**

Conference Papers :

1. **X. Wang**, Q. Li, D. Yu, Z. Wang, H. Chen, and G. Xu. MGPolicy: Meta Graph Enhanced Off-policy Learning for Recommendations. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval **(SIGIR)**, July 11, 15, 2022, Madrid, Spain. ACM, New York, NY, USA **(CORE A*, CCF A)**

2. **X. Wang**, Q. Li, D. Yu, and G. Xu. Off-policy Learning over Heterogeneous Information for Recommendation. In Proceedings of the ACM Web Conference 2022 **(WWW)**, April 25, 29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA **(CORE A*, CCF A)**

3. **X. Wang**, Q. Li, W. Zhang, G. Xu, S. Liu, and W. Zhu. Joint relational dependency learning for sequential recommendation. In Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining **(PAKDD)**, May 11-14, 2020, Singapore **(CORE A, CCF C)**

OTHER CO-AUTHER PAPERS :

1. D. Yu, Q. Li, **X. Wang** and G. Xu. Counterfactual Explainable Conversational Recommendation. In IEEE Transactions on Knowledge and Data Engineering **(TKDE) (CORE A*, CCF A)**

2. D. Yu, Q. Li, **X. Wang** and G. Xu. Deconfounded recommendation via causal intervention. In **Neurocomputing (JCR-Q1, CCF B)**

3. D. Yu, Q. Li, **X. Wang**, Z. Wang, Y. Cao and G. Xu. Semantics-Guided Disentangled Learning for Recommendation. In Proceedings of the 26th Pacific-Asia Conference on Knowledge Discovery and Data Mining **(PAKDD)**, May 16-19, 2022, Chengdu, China **(CORE A, CCF C)**

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

This chapter briefly introduces the causality-centric paradigm for the recommender system and outlines two core research lines for recommendation model development: data-centric and model-centric recommender systems. It also explores the key research questions associated with both research lines. Additionally, this chapter presents the contributions of the thesis and provides an overview of the thesis structure.

1.1 Causality-centric Recommendation

In the era of information explosion [44], recommender systems (RecSys) have become essential tools for information filtering, playing a crucial role in helping users navigate vast amounts of information. RecSys assists users in accessing their preferred content while benefiting content providers by increasing content exposure and generating revenue. With their powerful information search capabilities, RecSys have become widespread across various online services [36, 77], including e-commerce platforms (Amazon, Taobao), social networks (Facebook, Instagram), and video-sharing platforms (YouTube, TikTok).

The past few decades have witnessed the rapid development of recommendation models. From early shallow models [22, 64, 82, 138, 159] to recent deep learning-based models [81, 200, 204, 205], the algorithms and architectures for RecSys are rapidly evolving. In general, these works have focused on innovations in models, following the *model-centric RecSys* paradigm [43], which assumes that the main constraint that affects recommendation performance is the model architecture and the training algo-

rithm. Recent studies have highlighted the growing importance of recommendation data, particularly as large language models (LLMs) have shown that leveraging massive, high-quality datasets in natural language processing can significantly enhance recommendation accuracy [53, 59, 60]. Thus, researchers have increasingly recognized the value of refining recommendation data, which has led to the emergence of the *data-centric RecSys* paradigm [251]. Data-centric RecSys posits that the effectiveness of recommendation models is ultimately determined by the quality of the data they are trained on. Higher-quality [202] and/or greater quantities of data [127] enable these systems to better capture user preferences. In conclusion, the two primary research directions in building a successful RecSys are model-centric and data-centric approaches. While the former focuses on refining the model's architecture and training algorithms, the latter emphasizes enhancing data quality and quantity, both of which set the upper bounds for recommendation performance.

Data and model are the recipe for the supervised learning of any RecSys. To uncover users' preferences through the analysis of user behavior data using a well-designed model, certain assumptions should be made to serve as the "seasoning" (i.e., prerequisites) that guide and enhance the model training process. For example, the well-established collaborative filtering model [64] assumes that users with similar historical behaviors will likely exhibit similar future behaviors. In data-centric RecSys, social recommendation models [191] integrate user social relations from social networks to enhance conventional interaction data, operating under the assumption that users tend to prefer items their friends like. Social-aware matrix factorization methods [69, 93, 131] incorporate mechanisms such as trust propagation [93] and social matrix decomposition [131], bringing users' latent vectors closer to those of their trusted peers to reflect shared preferences.

Generally, the foundation of the assumptions for today's recommendation model is to model the *correlation*, such as behavioral correlation in collaborative filtering, or social correlation in social recommendations. However, real-world applications are driven by *causality* rather than correlation, while correlation does not imply causation [154]. For example, user behavior correlations in collaborative filtering models assume user conformity [261] (a.k.a. bandwagon effect), which can misrepresent true user preferences; that is, most users buy items based on their own preferences rather than complying with others. Instead, user behaviors reflect the decision-making process of humans, where the cause explains "why" certain items are preferred, while the effect describes "what" are the true user preferences. There are various types of causality that play a crucial role in today's recommendation systems (RecSys), such as user-aspect and interaction-

aspect causality. User-aspect causality refers to the causal relationships within a user's decision-making process. For example, a user may purchase a phone and subsequently buy a battery charger, where the purchase of the phone acts as the cause and the purchase of the charger as the effect. This causal relation reflects a sequential decision and cannot be reversed. On the other hand, interaction-aspect causality highlights how the recommendation strategy can shape and influence user behavior. For example, the absence of an interaction between a user and an item does not necessarily mean the user dislikes the item; it may simply be because the item was not presented or exposed to the user. Hence, unobserved interactions should not be interpreted as negative signals without considering the causality of exposure.

Therefore, modeling these diverse causalities is crucial for improving the accuracy of user preference inference and optimizing recommendation outcomes. By accounting for the underlying causes behind user behavior and interactions, RecSys can deliver more relevant and personalized content, ultimately enhancing user satisfaction. In recent years, the *causality-centric recommendation* paradigm [130] has gained significant traction, attracting substantial attention from researchers and practitioners. This emerging approach introduces new theories, methodologies, and applications that prioritize understanding causal relationships in recommendation systems. Extensive experiments and theoretical validations have shown that causality-centric methods offer several advantages over traditional recommendation techniques. These include improved accuracy, enhanced explainability, greater stability, and superior generalizability, making them highly promising for real-world applications. With these facts in mind, this thesis aims to research causality-centric recommendations, exploring causal inference approaches to increase the performance of data-centric and model-centric RecSys. This thesis aims to achieve the following research objectives:

- Provide precise causal formulations for scientific questions in data-centric and model-centric RecSys, harmonizing terminology and concepts across both domains. This will help mitigate misunderstandings in problem formulations and facilitate more rigorous statements in causality-centric recommendations.

- Discuss how causal inference approaches benefit both data-centric and model-centric RecSys, with a focus on improving performance. Detailed discussions on effective causal approaches for addressing key RecSys research questions are also included.

- Propose six causality-centric recommendation models to tackle challenging tasks

in both data-centric and model-centric RecSys, addressing issues such as data bias, model accuracy, explainability, and fairness. The proposed solutions offer valuable insights for the domains of recommendation accuracy enhancement, explainable recommender systems, fairness-aware recommender systems, and reinforcement learning-based recommender systems.

1.2 The Role of Data and Model in Recommendation

This section discusses the fundamental concepts and key research questions in the two primary research lines of recommender systems: data-centric recommender systems (Data-centric RecSys) and model-centric recommender systems (Model-centric RecSys).

Any artificial intelligence (AI) system, including recommender systems, comprises prototypical features from data and the algorithm (model) that solves the problem. Therefore, both data and model are crucial to the functioning of RecSys, making data-centric and model-centric approaches the two primary investigative strands within the field. Model-centric RecSys focuses on creating the best possible model from a given dataset, assuming that enhancing the model architecture and training process is key to improving recommendations. Conversely, data-centric RecSys emphasizes improving the quality of the data itself to achieve better performance from a given model. For years, model-centric approaches have dominated RecSys research. However, as researchers have begun recognizing the limitations posed by insufficient or poor-quality training data, there has been a shift toward investigating data-centric methods. Despite this shift, in practice, a holistic approach that simultaneously refines both the data and the model is necessary to build optimal recommender systems. The following sections will discuss existing work in these two research strands and outline the key research questions that arise in the pursuit of both data-centric and model-centric RecSys.

1.2.1 Data-centric Recommendation

Research that emphasizes the importance of data quality and enhancement is collectively referred to as data-centric recommender systems (Data-centric RecSys) [251]. The core principle of this approach is that a model's potential is determined by the quality of the data it processes. As a result, improving the quality [202] or increasing the quantity [127] of recommendation data directly enhances the system's ability to capture and predict user preferences more effectively. As shown in Figure 1.1, the evolution of

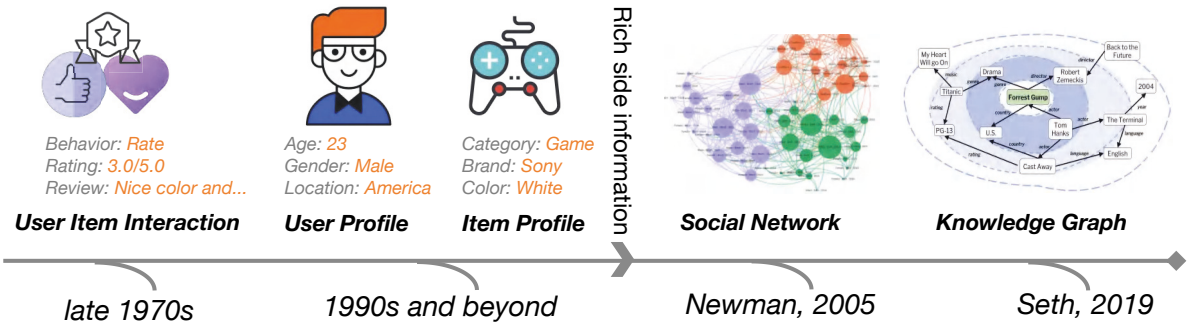


Figure 1.1: Data-centric recommendation: data enhancement evolutionary.

data-centric recommendations has significantly advanced data enhancement techniques. Starting from basic user-item interaction data in the late 1970s, the field has progressed to incorporate more complex and informative data, such as side-information-aware data, which includes user demographics, item attributes, and contextual information. Modern recommendation data is growing larger and increasingly hybrid, incorporating diverse side information such as social networks [191], knowledge graphs [204], and heterogeneous information networks [174]. As exemplified by the striking success of social recommendation [191] that incorporates users' social relations from social media to promote better user preference modeling, the benefits of enriching the recommendation data become obvious. Additionally, the recent advancements in Large Language Models (LLMs) have significantly boosted recommendation performance [53, 59, 60], largely due to the availability of vast, high-quality datasets in natural language processing. This has enabled recommendation models to better capture complex user behavior patterns. In summary, the field of Data-centric RecSys highlights the crucial importance of both data quality and quantity in the development of effective recommendation models.

However, data is highly susceptible to poisoning, as highlighted by extensive research [26, 28, 111]. This vulnerability introduces a critical challenge in data-centric recommendations, commonly referred to as the data bias issue [26].

Ubiquity of Data Biases in RecSys. Three key factors shape recommendation data: the user, the item, and their interactions. Across each of these dimensions, data bias is pervasive and undermines the effectiveness of recommendations: 1) From the user perspective, user behavior patterns are highly complex and influenced by various factors such as psychology and sociology. For instance, users often rate items similarly to others in their social circles (e.g., friends), which means that rating values may not always reflect a user's true preferences. In this context, conformity bias [261] arises, leading

recommendation models to favor recommending similar items rather than those that genuinely align with individual user preferences. 2) From the item side, the distribution of items in historical data is usually skewed. For example, the item distribution exhibits a long-tailed distribution in most recommender systems, causing the well-known exposure bias [240] issue: some items are presented more frequently in historical data than others, and thus receive more exposure in future recommendations. As a result, less-popular items, even when they match user preferences, are overlooked, creating biased recommendations that disproportionately favor popular items and significantly reduce the overall effectiveness of the system. 3) From the interaction side, user-item interactions are affected by various factors, such as user conformity and item exposure mentioned above. In addition, the RecSys itself is an interaction tool that interacts with users, that is, the exposure mechanism of the RecSys determines user feedback and user behaviors. This interaction loop not only generates biases but also exacerbates them over time, leading to the "poor get poorer" phenomenon. In conclusion, data bias is everywhere in today's RecSys caused by data issues such as data missing [219] and data imbalance [232]. As a result, the importance of data enhancement cannot manifest itself unless data biases are adequately addressed.

1.2.2 Model-centric Recommendation

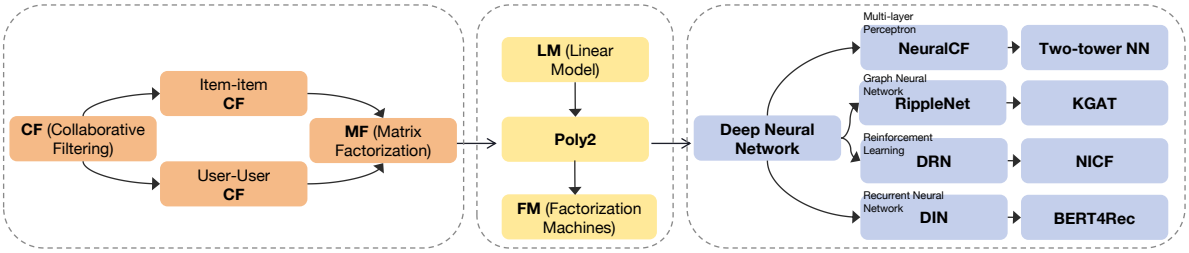


Figure 1.2: Model-centric recommendation: model innovation evolutionary.

Model-centric recommender systems (Model-centric RecSys) [43] assume that the main constraint that affects recommendation performance is the recommendation model with respect to model architecture and training algorithm. Countless efforts have been dedicated to developing a wide range of recommendation models. The evolution of these models can be categorized into three main groups: collaborative filtering-based models, linear models, and neural models. As shown in Figure 1.2, early recommendation models are primarily developed based on collaborative filtering (CF), a well-established method

in RecSys pioneered by Goldberg et al. [64] in their Tapestry system. Later, linear models [82] gained prominence due to their ability to address the cold start problem by incorporating side information such as user demographics, historical behaviors, and contextual data. The examples are degree-2 polynomial (Poly2) model [22], Follow-the-Regularized-Leader (FTRL) algorithm [138] and Factorization Machines [159]. Since 2016, deep neural network-based recommendation models have gained rapidly increasing attention by using a more flexible and representable neural network architecture. These models have not only become a major research focus in academia but have also dominated the development of industrial recommender systems, generating immense commercial value. This trend began with general multi-layer perceptrons (MLPs) and has since evolved into more advanced architectures like graph neural networks (GNNs) and recurrent neural networks (RNNs). According to recent statistics [73, 110], 90% of publications in the field now have a model-centric focus, and the sheer volume of new recommendation algorithms and methods is staggering. Most model-centric RecSys prioritize recommendation accuracy, as it directly impacts the model's utility once deployed. These efforts have resulted in substantial improvements in accuracy, with models now closely aligning with users' true preferences. However, in recent years, there has been a growing recognition that accuracy alone is not enough. Researchers have increasingly emphasized addressing beyond-accuracy objectives to ensure a trustworthy and well-rounded recommendation system. For instance, modern recommenders, with their use of complex neural networks, have become increasingly sophisticated and opaque. As a result, transparency has emerged as a major concern for these black-box models. To assist system designers in understanding and tracking the decision-making processes of recommendation models, it is crucial to provide explanations of predictions, which are essential for model debugging [49]. Consequently, explainable recommendation systems - designed to offer personalized recommendations along with explanations - have gained significant attention and become a hot topic in the field.

In conclusion, two indispensable and complementary topics dominate the model-centric RecSys research: recommendation accuracy and beyond-accuracy objectives.

- **Recommendation Accuracy.** Two widely adopted methods contribute to improving recommendation accuracy: 1) *Model architecture refinement*: This involves innovating model architectures, such as combining different neural networks or enhancing training algorithms through techniques like contrastive learning and semi-supervised learning. Recent advancements in neural models have demonstrated that traditional collaborative filtering is limited in its ability to capture

complex user behavior beyond mere user similarities. To address this, various types of neural networks have been incorporated into collaborative filtering models. For instance, Neural Collaborative Filtering (NCF)[81] employs a multi-layer perceptron (MLP) to learn a user behavior similarity function based on simple user/item one-hot encodings. Similarly, Neural Graph Collaborative Filtering (NGCF)[205] utilizes graph neural networks (GCNs) to learn collaborative signals from a user-item interaction graph, leveraging the GCNs' capability to capture the semantics of both users and items. 2) *Model optimization:* This approach focuses on designing optimal learning objectives (e.g., loss functions) to effectively guide the learning trajectory of recommendation models. For instance, while off-policy learning is commonly used to optimize reinforcement learning agents, it often suffers from biased estimations due to the nature of its learning objectives [30, 133, 199]. Since off-policy learning trains a target policy based on data collected by a previously deployed logging policy, the bias arises from the distribution discrepancy between the logging policy and the new target recommendation policy. Therefore, it is crucial to design unbiased policy optimization objectives to ensure the accuracy of recommendations generated by off-policy learning-based methods.

- **Beyond-accuracy Objectives.** Explainability and fairness are two key beyond-accuracy objectives essential for ensuring a trustworthy recommender system: 1) *Explainability:* Explainability is crucial for any black-box recommendation model. Researchers have invested significant effort into uncovering the underlying mechanisms driving these models. Firstly, providing users with explanations for recommendations enhances their understanding of how the system works and increases their trust in it. Secondly, explainability contributes to system transparency by clarifying why certain recommendations are made, allowing for better tracking of the model's decision-making process. In essence, explainability not only improves the persuasiveness of recommendations but also aids system designers in debugging and refining the models. 2) *Fairness:* Recommender systems (RecSys) are susceptible to ethical and fairness issues, which can significantly impact user satisfaction and damage stakeholder reputations. For instance, women make up a large proportion of the nursing workforce. When a job recommendation system learns from this data to estimate future employment opportunities, it may disproportionately suggest nursing positions to women while offering fewer such opportunities to men, thereby perpetuating gender discrimination. To enhance the satisfaction of all participants, addressing fairness issues in recommender systems

is crucial for fostering a positive and sustainable ecosystem.

1.3 Research Questions

In summary, the RecSys domain grapples with three critical research challenges from both data-centric and model-centric perspectives: data biases, recommendation accuracy, and beyond-accuracy objectives. To address these challenges, this thesis seeks to explore causality-centric methods aimed at providing solutions for each of these research questions. In particular, this thesis investigates the following research qustions:

- **R1:** How to give precise causal formulations for the scientific questions in the topics of data biases, recommendation accuracy and beyond-accuracy objectives, thereby explicating the perplexing causal notions?
- **R2:** How to resolve data bias using effective causal inference approaches?
- **R3:** How to achieve better recommendation accuracy by adopting causal inference approaches to refine 1) model architecture and 2) model optimization?
- **R4:** How to achieve beyond-accuracy objectives regarding 1) model fairness and 2) model explainability through causal inference?

1.4 Thesis Organization

The organization of this thesis is outlined in Figure 1.3. Specifically, Chapter 2 provides a comprehensive review of recommender systems (RecSys) and causal inference for recommendations. Chapter 3 introduces causal notions related to the key RecSys research questions explored in this thesis, including addressing data bias, improving model accuracy, enhancing model explainability, and ensuring model fairness. This chapter also presents effective causal inference approaches to tackle these challenges. Chapters 4, 5, and 6 offer practical solutions for research questions R2, R3, and R4, leveraging causal inference methods to promote more accurate and trustworthy recommendations. In detail, Chapter 4 presents two solutions targeting exposure bias and popularity bias in data-centric RecSys. Chapter 5 explores how causal inference can enhance model architecture and optimization to improve model accuracy. Chapter 6 focuses on counterfactual explanations and counterfactual fairness to achieve model explainability

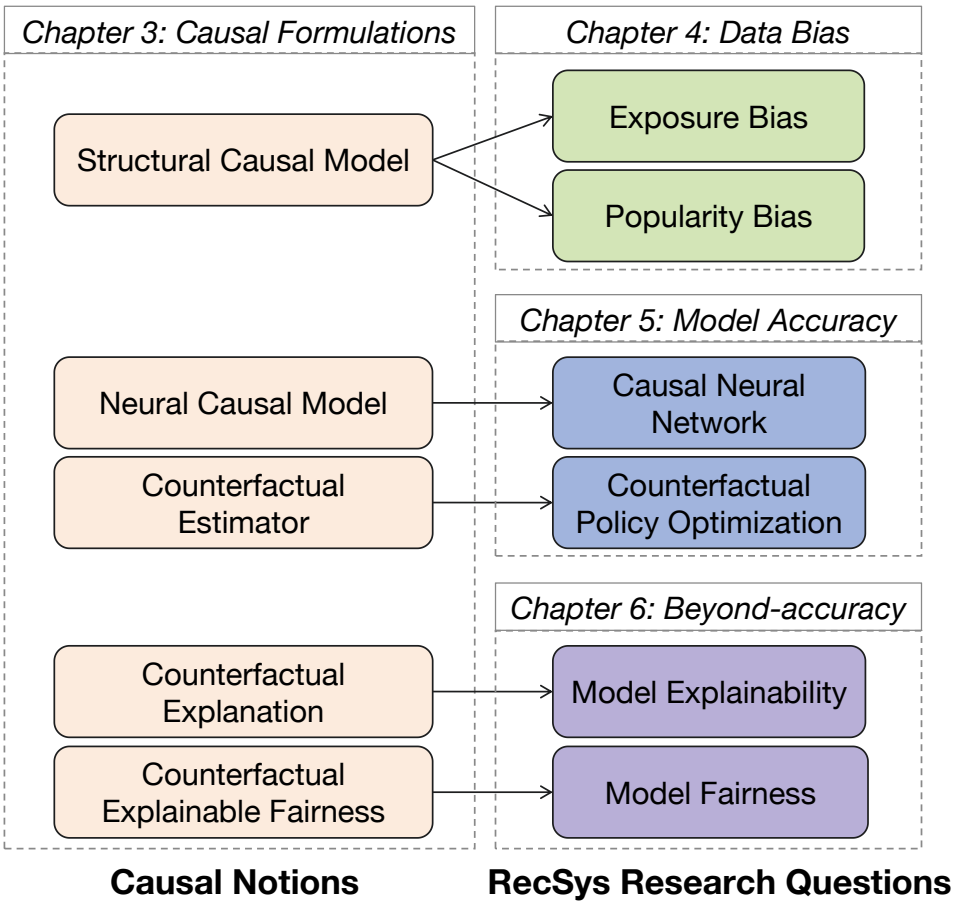


Figure 1.3: Thesis organization.

and fairness. Finally, Chapter 7 concludes the thesis, while Chapter 8 identifies promising directions for future research.

CHAPTER 2

LITERATURE REVIEW

This section provides a comprehensive review of the literature on two closely related domains: the recommender system (RecSys) and causal inference for recommendations.

2.1 Recommender System

As discussed in Section 1.2, two research lines are critical for RecSys research: 1) data-centric recommender systems and 2) model-centric recommender systems. Practical solutions for these topics are discussed in the following sections.

2.1.1 Data-centric Recommender System

Data-centric recommender systems aim to achieve better recommendations through the *enhancement of recommendation data*. Their fundamental principle comes from the core idea: "*Data is the food for AI*" [111], in which data ultimately dictate the upper limits of model capabilities. For data-centric recommender systems, the idea is straightforward: the higher the quantity [127] and/or quality [202] of recommendation data, the more effectively RecSys can capture user preferences. Thus, the relevant literature can be roughly categorized as multimodal data enrichment methods and data issue-aware methods, aligning with improving the quantity and quality of data, respectively.

Multimodal data enrichment methods aim to refine the data by including multiple modalities or types of data information, such as text [101], images [201], audio [38],

or any combination thereof. Traditional RecSys relies mainly on simple user-item interaction data, such as user clicks [165], purchases [12] and ratings [182], to generate recommendations. For instance, early collaborative filtering (CF) [18] methods utilize the user ratings matrix on items to identify similar users and recommend items that similar users have liked. Another example is content-based filtering [194], which recommends items based on a user's past interactions, such as articles they have clicked on or products they have purchased. Recent models have largely incorporated multimodal data, enabling recommender systems to capture more comprehensive patterns of user behavior. Two of the most notable achievements are social network-based models [46, 93, 116, 132] and heterogeneous information network-based models [177, 246, 247]. Social networks [191] record users' social relations across social platforms such as Facebook and Instagram. Social network-based models aim to use these rich social relations to capture users' collaborative behavior with their friends in order to refine the recommendation model. SocialMF [93] is a representative work that extends the matrix factorization (MF) model to include the social information of users. SoReg [132] follows [93] to model social information as regularization terms to constrain the MF model. SREE [116] models user and item embeddings into an Euclidean space as well as users' social relations. GraphRec [46] is a state-of-the-art social recommender that organizes user behaviors as a user-item interaction graph and models social information with a Graph Neural Network. As a newly emerging graph structure, the heterogeneous information network (HIN) [175] has been shown to be effective in modeling complex objects and providing rich semantic information to recommendation systems [185]. Many HIN-based recommendation methods achieve state-of-the-art performance [175]. For example, HeteMF [246] utilizes meta path-based similarities as regularization terms in the MF model. HeteRec [247] learns meta path-based latent features based on different types of entity relationships and proposes an enhanced personalized recommendation framework. SemRec [177] proposes a weighted HIN and designs a meta path-based collaborative filtering (CF) model to integrate heterogeneous information. To conclude, multimodal data enrichment methods make the recommendation data grow larger and increasingly hybrid, incorporating additional contextual information (e.g., social relations and heterogeneous information) to provide a more comprehensive picture of the interactions between users and items.

Data issue-aware methods aim to improve the data quality through data debias means. Data issues are largely presented, such as data missing [23] and data noise [230], causing various data biases that deteriorate the recommendation performance. *Exposure bias* is a common data bias arising from the partially observed nature of user-item

interactions. Traditional methods, such as matrix factorization (MF) models [18, 165], consider observed user interactions (e.g., clicks) as positive while treating all unobserved interactions as negative. However, in real-world scenarios, users are only exposed to a subset of items due to various reasons, such as platform selection. Consequently, unobserved interactions do not necessarily indicate negative preferences, leading to exposure bias issues in those traditional methods. Data imputation methods [27, 85, 136, 150] handle the exposure bias problem using missing data imputation strategies, as missing data can be interpreted as the source of unobserved interactions. They aim to jointly model the generative process of user feedback (e.g., rating values) and the missing mechanism. For example, Marlin and Zemel [136] model the missing probability of a user-item pair dependent on the user rating values through a Mixture of Multinomials (MM) model. A probabilistic matrix factorization is proposed to characterize the missing probability of a user-item pair [182] to improve the flexibility of the MM model. Hernandez et al. [85] use a probabilistic matrix factorization (PMF) model with hierarchical priors for ordinal rating data, which increases the robustness of the hyperparameter selection. Ohsawa et al. [150] further extend the PMF model to a Gated PMF by considering the dependency between why a user consumes an item and how that affects the rating value. Chen et al. [27] model user consumption with social influence to better estimate the user's preference for items. However, data-imputation-based methods are not robust due to their unstable imputation accuracy, which would be propagated into training a prediction model and easily mislead the prediction [119, 219]. *Popularity bias* is another common bias issue caused by the uneven distribution of items, e.g., popular items are recommended even more frequently than their popularity would warrant. Regularization-based methods are commonly used to handle popularity bias by pushing the model toward balanced recommendation lists. Abdollahpouri et al. [1] introduced a $LapDQ$ regularizer to constrain the equal probability of recommending popular items and long-tail items. Kamishima et al. [100] uses the mean-match regularizer [99] in an information-neutral recommender system. They initially utilized mutual information to assess the impact of features on recommendation outcomes. Through a series of mathematical approximations and derivations, they derived a specific regularization term: $-\left(\boldsymbol{M}_{D^{(0)}}(\{\hat{r}\}) - \boldsymbol{M}_{D^{(1)}}(\{\hat{r}\})\right)^2$, where $\boldsymbol{M}_D(\{\hat{r}\}) = \frac{1}{|D|}\sum_{(x_i,y_i,v_i)\in D}\hat{r}(x_i,y_i,v_i)$. However, regularization-based models imply strong assumptions, such as the equal exposure probability of popular and nonpopular items, and thus are usually unrealistic in real-world recommendations.

2.1.2 Model-centric Recommender System

Table 2.1: Taxonomy of model-centric recommender system.

	CF-based models	**Linear models**	**Neural models**
Approach	Autocomplete	Feature engineering	Deep neural networks
Data	Historical data	Complex sources	Complex sources
Evaluation setup	Offline datasets	Offline datasets Auxiliary information	Offline datasets Auxiliary information
Topic	Collaborative Filtering	Feature Engineering Features Conjunction etc.	Sequential RecSys Graph-based RecSys Explainable RecSys etc.

Model-centric recommender systems aim to primarily refine the model architecture and training algorithm [43]. In the history of the recommender system, model-centric recommender systems have innovated from traditional collaborative filtering-based models to linear models and to the recent neural models. Table 2.1 provides the taxonomy of these three types of models by summarizing their own characteristics.

Collaborative Filtering-based Models are often developed in a class of collaborative filtering (CF) methods, such as content-based filtering [107, 136] and matrix factorization [90]. They train the recommendation model with the historical interaction logs of users (e.g., users' historical ratings on movies) and all relevant profiles, such as users' age, social relations, and movie type. The recommendation model is evaluated under offline evaluation metrics (e.g., Precision, Recall) and is used to predict users' future preferences (e.g., ratings) on items that they have not yet examined. In a nutshell, collaborative filtering-based models aim at performing user-item matrix completion on the organic user-item interactions. Thus, the process can be termed as an "auto-complete" manner [95]. Ranging from rating prediction to next-item prediction tasks, collaborative filtering-based models have found wild-spread success in academic and industrial scenarios [83]. The core concept is to use collaborative user behaviors to predict future user preferences, i.e., similar users may share similar tastes towards items [183]. Based on such intuition, early approaches calculate the user behavior similarity (user-based CF [105, 164]) or the item similarity (item-based CF [123, 169]) based on functions such as Pearson correlation coefficient. Later, latent factor models (LFM) [2], e.g., matrix factorization [108] and tensor factorization [109], gain prominence by collaboratively uncovering the latent space that encodes the interaction matrix of the user. LFM aims to model users and items as latent factors to calculate matching scores of user-item pairs based on pure user explicit feedback data, e.g., ratings. Learning-to-rank (LTR) [21]

steps from learning the absolute matching scores to learning the relative ordering of items. A representative method is Bayesian personalized ranking (BPR) [162], a pairwise LTR method that is tolerated to learn from the user feedback involved, e.g. clicks. BPR uses the inner products between latent vectors as user-item similarities to predict user preference scores.

Linear Models gain prominence due to their ability to address the cold start problem by incorporating side information such as user demographics, historical behaviors, and contextual data [82]. These models are designed to capture user preferences within more complex behavioral patterns compared to early CF methods. A linear model models recommendation predictions with the function $\phi(\boldsymbol{w},\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$, where each feature $\boldsymbol{x}$ (e.g., the brand of the item) is assigned a weight $\boldsymbol{w}$. For example, degree-2 polynomial (Poly2) model [22] learns weights for every possible pair of characteristics. Facebook [82] designed a hybrid model for automatic feature engineering, where the logistic regression is combined with decision trees. They constructed boosted decision trees to transform input features, and the output of each tree serves as input feature to a logistic linear classifier. McMahan et al. [138] proposed the Follow-the-Regularized-Leader (FTRL) algorithm. Besides, an FTRL-Proximal online learning strategy is designed and used to enhance CTR prediction accuracy. The logistic regression model is practical for industrial recommender systems due to its ability to model side information and ease of extension. However, LR cannot inherently model feature conjunctions, relying on artificial feature engineering or other models like Poly2 or boosted decision trees for this purpose.

Neural Models typically use deep neural networks [10, 80, 205] as the backbone for recommender systems. They have drawn a lot of attention recently thanks to their strong capacity to solve complex tasks and adapt to more complex data. Specifically, neural models can handle non-linear user-item relations by using the higher-order network structure, while special classes of neural networks such as Graph Neural Networks (GNN) [80] and Recurrent Neural Networks (RNN) [216] are well adapted with complex data sources such as graphs and sequences. Relevant approaches to neural models can be classified into similarity learning methods and representation learning methods. Similarity learning methods aim to learn a similarity function with a predictive neural network based on simple user/item representations, e.g., one-hot [81]. Neural collaborative filtering (NCF) [81] is the first work using a multi-layer perceptron (MLP) for similarity prediction. Following NCF, a majority of models (e.g., [3, 88, 157, 248]) have abandoned the fixed similarity function in CF-based models (e.g., dot product) to use MLPs as general function approximators. The rationale is that MLPs are general function approximators,

so they should be strictly better than a fixed similarity function such as the dot product. Another branch of representation learning methods learns rich user/item representations and adopts a simple matching function (e.g., inner product) to calculate matching scores. They aim to use different neural networks to encode latent user behavior patterns, e.g., temporal user behavior and multi-type behaviors, into user/item representations. For example, GRU4Rec [86] models temporal information from user interaction histories based on a Recurrent Neural Network with the gated recurrent unit. NMTR [52] performs multi-task learning for the joint training of three neural network units to cascade relations among different user behavior types (e.g., click, purchase). Recently, Graph Convolutional Networks (GCN) have been widely adopted in neural models, primarily due to their well-established performance in learning local and global information from large-scale graphs. As evidenced by several studies [10, 80, 184, 205], GCN advances from other neural networks in capturing graph structural information. This is because GCN leverages convolutional operations to aggregate both local information from direct neighboring nodes and global information from higher-hop neighbors. Besides, the parameter sharing of GCN allows it to generalize well to unseen nodes during training and efficiently adapt to large-scale graphs. GCN-based neural models aim to acquire vectorized user and item embeddings using a GCN and then use these embeddings to optimize a CF model. For instance, neural graph collaborative filtering [205] exploits a GCN to propagate neighboring node messages in the interaction graph to obtain user and item embeddings. The learned embeddings capture user collaborative behavior and are used to predict preference scores for CF optimization. Hyperbolic graph collaborative filtering [184] offers a compelling solution by integrating GCN with hyperbolic learning techniques to acquire user and item embeddings within the hyperbolic space. By leveraging the exponential neighborhood expansion inherent in the hyperbolic space, HGCF effectively captures higher-order relationships among users and items, enhancing the learning capabilities for downstream CF models. GC-MC [10] uses a GCN-based auto-encoder to learn latent features of users and items from an interaction graph and reconstructs the rating links for matrix completion. Later, LightGCN [80] simplifies the GCN in the recommendation task by only including neighborhood aggregation for calculating user and item representations, which further boosts the efficiency of subsequent GCF approaches, e.g., [50, 148, 225].

In recognition of advances in deep learning techniques, the majority of new recommendation algorithms described in academic papers are aimed at improving the accuracy of recommendations [205]. In a parallel research direction, *explainable recommendation*

frames the task as generating not only high-quality recommendations but also intuitive explanations, in which the problem of why certain items are recommended can be answered through enhancing the recommender with the *interpretability*. Recent work [79] has repeatedly demonstrated the importance of interpretability in terms of increasing recommendation transparency, persuasion, efficacy, reliability, and satisfaction. Interpretable explanations are usually encoded by the context of the users/items, such as the specific gender, age of a user, and the type and year of a movie.

Other research also shows that *fairness* is also one of the main concerns of recommender systems, which profoundly harms user satisfaction [51, 117] and stakeholder benefits [11, 57, 118]. For example, an RS may differentiate items to allocate more exposure to popular items while overlooking niche items [1, 210], causing major concerns such as the Matthew effect [156]. *Fairness-aware recommendation* [51] has emerged as a promising solution to prevent unintended discrimination and unfairness in RS.

2.1.2.1 Explainable Recommendation

The explainable recommendation has been shown to improve user satisfaction [192] and system transparency [6]. Research in explainable recommendation can be divided into two main categories [58, 256]. The first category of model-intrinsic approaches [84, 122, 152, 169, 179, 198] designs interpretable recommendation models to facilitate explanations. These model-intrinsic methods train explainable models to identify influential factors for users' preferences, and then construct explanations accordingly. For example, Shulman et al. [179] propose a per-user decision tree model to predict users' ratings on items. Each decision tree is built with a single user as root and items as leaves, while a linear regression is applied to learn leaf node values to build decision rules. The explanation is formed as a sequence of decisions along the decision tree. Lin et al. [122] propose a personalized association rule mining method targeted at a specific user. Explanations are generated by identifying association rules between users and items. Other works [84, 114, 152, 169, 198, 208] explore content-based approach [198] and neighborhood-based collaborative filtering [84, 169] to generate explanations based on users' neighbors [84, 114, 208], item similarity [169], user/item features [198], or combinations thereof [152].

As modern recommendation models become complicated and black-box, e.g., embedding-based models [74, 91, 206] and deep neural networks [24, 81], developing model-intrinsic approaches [256] is not practical anymore. Thus, another category of model-agnostic methods [62, 149, 180] aims to explain black-box models in a post hoc manner. For

instance, Ghazimatin et al. [62] perform graph learning on a heterogeneous information network that unifies users' social relations and historical interactions. The learned graph embeddings are used to train an explainable learning-to-rank model to output explanation paths. Singh et al. [180] train a Learning-to-rank latent factor model to produce ranking labels. These ranking labels are then used to train an explainable tree-based model to generate interpretable item features for ranking lists.

However, existing model-agnostic approaches suffer from major limitations: 1) They mainly concentrate on factual scenarios using historical user-item interactions to identify item features. However, they typically neglect the counterfactual scenario in which changing explanations can affect future recommendations. This restricts their ability to predict changes in future recommendations if explanations are modified. Incorporating these counterfactual scenarios is vital for a deeper understanding of the system and to understand how varied explanations influence recommendations. 2) They construct explanations using a fixed number of influential factors without taking into account the complexity of the explanation [258]. Instead of seeking the minimal set of influential factors for recommendations, these methods might incorporate redundant or irrelevant elements. This can result in unnecessary computational complexity. For example, given the condition that the *brand* have been "Apple" well explains the user's preference, using "Apple", "Electronic", "California" as the explanation becomes redundant and decreases the explanation efficiency.

Recently, counterfactual explanation [189] has emerged as a favorable opportunity to solve the above questions. The counterfactual explanation interprets the recommendation mechanism by exploring how minimal changes to items or users affect the recommendation decisions. Typically, *counterfactual explanation* is defined as a minimal set of influential factors that, if applied, flip the recommendation decision. To build counterfactual explanations, we have to fundamentally address: "*what the recommendation result would be if a minimal set of factors (e.g., user behaviors | item features) had been different*?" [197]. With counterfactual explanations, users can understand how minimal changes affect recommendations and conduct counterfactual thinking under the "what-if" scenario [153].

2.1.2.2 Fairness-aware Recommendation

Recommender systems have long dealt with major concerns of recommendation unfairness, which profoundly harm user satisfaction [51, 117] and stakeholder benefits [11, 57, 118]. Recent works on fairness-aware recommendation mainly discuss two

primary topics, i.e., user-side fairness [29, 51, 92, 117, 119, 242] and item-side fairness [1, 40, 54, 128]. User-side fairness concerns whether the recommendation is fair to different users/user groups, e.g., retaining equivalent accuracy or recommendation explainability. Relevant approaches attribute the causes of user-side unfairness to discrimination factors, such as sensitive features (e.g., gender [29, 242], age [119]) and user inactiveness [51, 117], etc. They mainly propose fairness metrics to constraint recommendation models (e.g., collaborative filtering [242]) to produce fair recommendations. For example, Yao et al. [242] study the unfairness of collaborative filtering (CF)-based recommenders on gender-imbalanced data. They propose four metrics to assess different types of fairness, and then add these metrics as constraints to the CF model learning objective to produce fair recommendations. Chen et al. [29] investigate gender-based inequalities in the context of resume ranking engines. They use statistical tests to show the existence of direct and indirect gender discrimination in resume ranking results provided to job recruiters. Li et al. [117] investigate the unfair recommendation between active and inactive user groups, and provide a re-ranking approach to mitigate the activity unfairness by adding constraints over evaluation metrics of ranking. As modern content providers are more concerned about user privacy, it is generally not easy to access sensitive user features for recommendation [163]. Meanwhile, users often prefer not to disclose personal information that raises discrimination [9], limiting the application of user-side fairness recommendations. Another topic of item-side fairness-aware recommendation [1, 40, 54, 128] is interested in examining whether the recommendation treats items fairly, e.g., similar ranking prediction errors for different items, fair allocations of exposure to each item. For instance, Abdollahpouri et al. [1] address item exposure unfairness in learning-to-rank (LTR) recommenders. They include a fairness regularization term in the LTR objective function, which controls the recommendations favored toward popular items. Diaz [40] address the exposure unfairness due to the biased item popularity in user behavior data. They propose the concept of expected exposure as the average attention ranked items receive from users and apply it to measure item exposure under stochastic versions of existing retrieval and recommendation algorithms. Ge et al. [54] consider the dynamic fairness of item exposure due to changing group labels of items. They calculate the item exposure unfairness with a fairness-related cost function. The cost function is merged into a Markov Decision Process to capture the dynamic item exposure for recommendations. Liu et al. [128] focus on item exposure unfairness in interactive recommender systems (IRS). They propose a reinforcement learning method to maintain a long-term balance between accuracy and exposure fairness in IRS. Other

successful investigations of fairness can also be closely related to enhancing recommendation diversity [102, 135], e.g., increase aggregate diversity [135], and increasing fairness transparency of graph neural networks [42].

Despite the great efforts, fairness-aware recommendations mitigate user and item unfairness in a black-box manner but do not explain why the unfairness appears. Understanding the "why" is desirable for both model transparency and facilitates data curation to remove unfair factors [118]. Thus, *fairness explanation*, as an important aspect of fairness research, is emerging as a hot topic and would enhance the design of fairness-aware recommendation approaches by tracking unfair factors for model curation.

2.2 Causal Inference

Causal inference aims to learn the causality, that is, the generic relationship between entities, to understand the effect and the causes that give rise to it. The capacity to understand causality is viewed as the foundation of human-level intelligence and is thus a significant component of artificial intelligence [155]. Causal inference has been investigated in a myriad of high-impact domains, such as medical science [158], epidemiology [166] and meteorology [33]. Causal inference is supported by several key frameworks, with the two most widely referenced being 1) the Potential Outcomes Framework (POF) by Rubin et al. [167] and 2) the Structural Causal Model (SCM) by Pearl et al. [154]. POF focuses on estimating the causal effect of a treatment (e.g., an item feature) on an outcome, such as recommendation results. Inverse propensity weighting (IPW) [124] is commonly used within POF to address selection bias and confounding in observational studies, creating a pseudo-population where treatment assignment is independent of observed covariates. However, POF has limitations, such as a lack of intuitive graphical models to represent causal relationships, and its effectiveness heavily depends on the accuracy of propensity score estimation. This often leads to issues like "propensity overfitting" [215] due to unobserved variables, which can hinder POF-based recommendation performance. In contrast, SCM constructs a causal graph using structural equations that model causal relationships between deterministic variables. The causal graph enables causal reasoning and effect estimation, allowing researchers to address data biases, such as exposure bias [220] and popularity bias [257]. Additionally, SCM supports interventions and the exploration of hypothetical "what if" scenarios through do-calculus [154], enabling counterfactual reasoning in recommendation systems.

2.3 Causal Inference for Recommendation

Recently, increasing research has shown that causal inference can largely accelerate the performance of recommender systems in multi-aspects, such as the data debiasing tasks, recommendation model design, model explainability and model fairness.

2.3.1 Causal Debias Methods

Causal inference has attracted increasing attention in various debias tasks for recommender systems. Dominant approaches adopt two main causal inference approaches, i.e., propensity score [67, 120, 171] and causal embedding [14, 220, 241], to discover true user interests under different types of bias.

The propensity score for an individual is the conditional probability of receiving treatment based on the individual's covariates. It can be used to balance the covariates between the treatment and control groups, thus mitigating the bias between the two groups. The propensity score has been widely applied in the phenomenon of missing not at random (MNAR) [181], i.e., the missing pattern of data depends mainly on whether items are exposed to users, and consequently, the observed user feedback, in fact, is missing not at random. A couple of methods address MNAR [85, 120, 171] by treating missing items from the "exposure" perspective, i.e., indicating whether or not an item is exposed (provided) to a user. For example, ExpoMF [85] resorts modeling the probability of exposure by propensity scores and up-weighting the loss of rating prediction with high exposure probability. Likewise, recent works [120, 171] resort to propensity score to model exposure. For example, the method in [171] uses the propensity score to reweight the observational click data, with the aim of imitating the scenario where the item is randomly exposed and alleviating the exposure bias.

Causal embedding methods aim to first design a specific structural causal model on what determines the final recommendation outcomes, e.g., user interest and user conformity together cause the recommendation outcome [262]. The causal graph under the structural causal model is then utilized to guide the learning of the embedding for each component separately. In particular, the causal graph usually shows the presence of confounders that cause biased estimations. For example, the social network is, in fact, a confounder of exposure bias, since it influences both users' choice of movie viewing and their ratings [115]. The confounder introduces confounding effects, the ignorance of which misguides the learning of the user's true intent. A widely used solution for mitigating the confounding effects is learning the user embedding that is forced to be

independent of the confounder. For example, some works design a regularizer for the independence constraint using statistical measures such as L_1-inv, L_2-inv, and distance correlation [134]. By explicitly disentangling the cause of a confounder, the embedding uncovering the users' true intent can be learned for final recommendations. More recently, the work in [241] has resorted to balancing learning with a Middle-point Distance Minimization (MPDM) strategy to learn causal embeddings that are free from selection bias. A few state-of-the-art works [115, 203, 220, 257, 262] inspect the cause-effect of bias generation and designs a specific causal graph that attributes bias issues to confounders. For instance, Wang et al. [220] mitigate exposure bias in the partially observed user-item interactions by regarding the bias as the confounder in the causal graph. They propose a decounfonded model that performs Poisson factorization on substitute confounders (i.e., an exposure matrix) and partially observed user ratings. Zheng et al. [262] relate the user conformity issue in recommendations with popularity bias and use a causal graph to guide the disentangled learning of user interest embeddings. Wang et al. [209] define a causal graph that shows how users' true intents are related to item semantics, i.e., attributes. They propose a framework that produces disentangled semantics-aware user intent embeddings, in which each model component corresponds to a specific node in the causal graph. The learned embeddings are able to disentangle users' true intents towards specific item semantics, which reveal which item attributes are favored by users.

2.3.2 Causality-assisted Model Design

Causal inference can be applied to improve the recommendation accuracy from *model architecture refinement* and *model optimization* perspectives. In particular, causal neural network that refines neural network architecture with causal relations, and counterfactual policy learning that designs unbiased model optimization objectives, are two effective solutions in the model accuracy improvement task.

2.3.2.1 Causal Neural Network

Using causality to refine neural network architecture is still at its early stage with limited prior work. Xia et al. propose a Neural-Causal Connection theory [231] that introduces a special type of structural causal model called the neural causal model. They show that the proposed neural causal model is equally expressive as a neural model learned from existing neural networks, while the expressiveness, learnability, and inference of the neural causal model can be achieved by encoding structural constraints necessary

for performing causal inferences. Designed for Graph Neural Networks (GNNs), Zevcevic et al. [249] propose a neural causal model for GNN-based neural models using the theory proposed by Xia et al. Unfortunately, in the recommendation scenario, existing works do not consider the use of neural causal models to refine the learning process of recommendation models, e.g., collaborative filtering-based models. In addition, the first neural causal model that parameterizes each structural equation under the recommendation modeling as trainable neural networks has not yet been established.

2.3.2.2 Counterfactual Policy Learning

Reinforcement learning largely relies on on-policy and off-policy learning methods for policy optimization [218]. However, recent studies [30, 133, 199] show that both on-policy and off-policy learning methods can suffer from biased recommendations. Whereas both on-policy and off-policy learning methods require a logging policy, the bias occurs due to the distribution discrepancy between the previously deployed logging policy and the new target recommendation policy. The target recommendation policy includes rewards for all user-item pairs, while the rewards recorded by the logging policy are partially observed, i.e., only those items that have been recommended own the rewards, but not for non-recommended ones. Consequently, this reward mismatch issue leads to a data distribution discrepancy, causing a severe selection bias. To alleviate the selection bias, the basic idea is to consider rewards for both recommended and non-recommended items. However, those rewards for non-recommended items, also termed counterfactual data, are naturally not available. Thus, learning an unbiased policy without access to the counterfactual data (i.e., *Counterfactual Policy Learning*) has become an emerging topic.

Counterfactual risk minimization (CRM) methods [168] is an effective method for counterfactual policy learning. The core idea of CRM is to understand the counterfactual question: "*how would the new target recommendation policy have performed if it had been used instead of the policy that logged the data*?" Existing CRM methods largely rely on the inverse propensity scoring (IPS) technique. For example, Swaminathan et al. [187] leverage the IPS weighting technique to model the policy discrepancy in the assignment mechanism, thus providing unbiased estimates of the counterfactual risk (expected reward/loss) throughout a class of policies. This enables learning by optimizing the estimated counterfactual risk, potentially subject to variance regularization. Joachims et al. [97] propose a BanditNet that includes an additional self-normalization IPS (SNIPS) term while they extend the off-policy learning to deep neural networks, which is a recent notable extension. Chen et al. [32] extend a policy gradient-based method with

a Top-K IPS estimator and show significant gains from exploiting bandit feedback in online experiments. Ma et al. [133] formulate the target policy of the two-stage recommender system as the composition of the candidate generation policy and the ranker policy, meanwhile deriving the importance weight based on the IPS to correct the candidate generation policy. However, policy corrections based on IPS often have a large variance due to propensity estimation inaccuracy; thus, IPS-based works easily suffer from biased predictions and mislead the model building. An interesting open question in bias corrections for off-policy and on-policy learning may lie in the accurate estimation of correction values.

2.3.3 Counterfactual Explainable Recommendation

As discussed in Section 2.1.2.1, counterfactual explanations advance from conventional explanations by two aspects: 1) Faithful to the prediction: counterfactual explanations secure the factors in the explanations that could flip the recommendations. For example, the item brand "Apple" is used as the explanation and removing the "Apple" would cause the user to buy another phone. 2) Minimal for the explanation: counterfactual explanation consists of a minimal set of factors that are most influential to the recommendations. Thus, the complexity of the explanation can be largely reduced to the most deterministic ones. For example, if we know using "Apple" could explain the user's preference, then using "Apple" and "California" as the explanation could be redundant.

Counterfactual explanations have been considered as satisfactory explanations [228] and elicit causal reasoning in humans [19]. Successful works on the counterfactual explanation for recommendations can be categorized into search-based and optimization-based approaches. The first category of search-based approaches [98, 236] performs the greedy search for counterfactual explanations. Kaffes et al. [98] perturb user-item interactions by deleting items from user interaction queues to generate perturbed user interactions as counterfactual explanations. Then, a breadth-first search is used to find counterfactual explanations that achieve the highest normalized length and candidate impotence scores. Xiong et al. [236] propose constrained feature perturbations on the features of items and consider the perturbed item features as counterfactual explanations. The second category of optimization-based approaches [34, 61, 189, 193] optimizes explanation models to find counterfactual explanations with minimal changes. Ghazimatin et al. [61] perform random walks over a heterogeneous information network and calculate the PageRank scores after removing user action edges from the graph. Those minimal sets of user actions that change PageRank scores are deemed counterfactual explanations. Tran et

al. [193] identify a minimal set of user actions that updates the parameters of neural models. Tan et al. [189] modify item aspect scores to observe user preference changes based on a user-aspect preference matrix.

2.3.4 Counterfactual Explainable Fairness Recommendation

As discussed in Section 2.1.2.2, most fairness-aware recommendation research focuses on defining fairness measurements [117, 242] and developing feasible approaches [1, 54] to reduce the fairness disparity of recommendation results. Despite the great efforts, fairness-aware recommendations mitigate user and item unfairness in a black-box manner but do not explain why the unfairness appears. Understanding the "why" is desirable for both model transparency [118] and facilitates data curation to remove unfair factors [222]. This answers the fundamental problem: "*what causes unfair recommendation results*?". Fairness explanation, as an important aspect of fairness research, would enhance the design of fairness-aware recommendation approaches by tracking unfair factors for model curation.

However, limited pioneering studies have been conducted to explain fairness. Begley et al. [8] estimate Shapley values of input features to search which features contribute more to the model unfairness. Deldjoo et al. [37] uses regression-based explanatory modeling to investigate the relationship between data characteristics and the fairness of recommendation models. They define data characteristics as explanatory variables (EVs) and the recommendation outcome as dependent variables (DVs). Through statistical significance tests with informed p-values, they determine the EVs that are significant for recommendation models. However, estimating Shapley values and p-values should consider all features/characteristics across possible coalitions [118]. This poses a challenge of exponentially growing search space when the number of features/characteristics increases. More importantly, these two factual explainable methods only consider the relative importance of features/characteristics to the model fairness, while their removal does not guarantee a reduction in the model's unfairness. This limitation makes the explanations less reliable regarding their effectiveness in mitigating unfairness.

Counterfactual explainable methods, on the contrary, learn fairness explanations as the "minimal" change of the input features that guarantee the reduction of unfairness [55, 197]. They provide "what-if" explanations to determine the most vital and essential (i.e., minimal) factors that change model fairness [197]. Unlike factual explainable methods with high complexity, counterfactual explanations have the advantage of being minimal w.r.t. the generated explanations and are faithful to model fairness

changes. Unfortunately, there is only one counterfactual explainable method for recommendation fairness. Ge et al. [56] develop a counterfactual explainable fairness model for recommendation to explain which item aspects influence item exposure fairness. They perform perturbations on item aspect scores, and then apply perturbed aspect scores on two pre-defined matrices to observe fairness changes.

CHAPTER 3

THEORETICAL FOUNDATIONS

This chapter first introduces the structural causal model (SCM) [154], and the confounding and colliding structure are identified as the source of biased causal structure. Then, this chapter introduces the ladder of causation [154] in terms of intervention and counterfactual, which are two fundamental hierarchies for the estimation of causal effects. The final section discusses the preliminaries of effective causal approaches for solving RecSys research questions, i.e., data bias, model accuracy, model explainability, and model fairness.

3.1 Structural Causal Model

Unlike the potential outcome framework, the structural causal model (SCM) [154] explicitly represents the causal relationships between variables using a visualized causal graph. The causal graph is established on the basis of prior knowledge, drawing on assumptions from domain experts before analyzing the causal effect. For instance, the prior knowledge that good quality causes purchase behavior is represented as the directed edge *good quality* $\rightarrow$ *purchase* in the causal graph. Causal graphs are essential in the SCM framework because they provide a clear representation of the relationships among variables in a given state of knowledge. They serve as a carrier of conditional independence relationships, allowing us to confirm whether the criteria for applying certain causal inference methods are satisfied.

Definition 3.1.1 (**Structural Causal Model**)**.** A structural causal model is expressed as $\mathcal{M} = \langle \mathcal{V}, Z, \mathcal{F}, \mathbb{P}(Z) \rangle$ and includes a collection of structural equations $\mathcal{F}$ on endogenous variables $\mathcal{V}$ and a distribution $\mathbb{P}(Z)$ over exogenous variables Z. A structural equation $f_U \in \mathcal{F}$ for a variable $u \in \mathcal{V}$ is a mapping from u's parents and exogenous variables of u:

$$u \leftarrow f_U(pa(u), Z_u), Z_u \sim \mathbb{P}(Z) \tag{3.1}$$

where $pa(u) \subseteq \mathcal{V} \setminus u$ is u's parents that are the causes of u. $Z_u \in Z$ is a set of exogenous variables connected with u.

The causal graph for the structural causal model provides a friendly graphic for estimating causal effects.

Definition 3.1.2 (**Causal Graph**)**.** A causal graph is a directed acyclic graph (DAG) $G = (\{\mathcal{V}, Z\}, \mathcal{E})$ representing causal relations between endogenous and exogenous variables. $\mathcal{V}$ is a set of endogenous variables of interest, e.g., users and items. Z is a set of exogenous variables outside the model, for example, item exposure. $\mathcal{E}$ is edge set denoting causal relations among G. A directed edge $x \rightarrow y$ denotes that there is a causal relationship from x to y.

3.1.1 Construct SCM for Recommender System

It is valuable to provide a step-by-step guide for constructing the structural causal model (SCM) for specific applications in recommendation systems. Consider a commonly studied item feature-based recommendation scenario, where item features serve as the primary signals for recommendation. In this setting, there exists a causal relationship between the observed item features X and the user-item interaction Y, represented as $X \rightarrow Y$. Using this causal relationship $X \rightarrow Y$, existing models aim to directly learn a function that maps X to the space of Y. However, this approach inherently relies on the following assumptions:

Assumption 3.1.3 (Unconfoundedness)**.** Given a set of observed covariates, the assignment of treatment is independent of the potential outcomes. In other words, there are no unobserved confounders that simultaneously affect both treatment and outcome, ensuring that any differences in outcomes between the treated and untreated groups can be attributed to treatment itself rather than hidden biases.

Assumption 3.1.4 (Stable Unit Treatment Value Assumption (SUTVA))**.** The potential outcomes for any unit do not vary with the treatment assigned to others. For each unit,

there is only one version of treatment level that leads to one potential outcome. Here, the independence of each unit and well-defined treatment levels are emphasized.

However, in real-world recommendation scenarios, additional item features, denoted as A, may also influence user-item interactions Y in addition to X, forming the causal relationship $\{A, X\} \rightarrow Y$. X could directly affect the happening of an interaction in practice. For instance, videos with short lengths are easier to finish, and news articles with attractive titles or cover images are easier to click, even though the user does not like the content actually. These features result in the observed interactions not faithfully reflecting user preference. In addition, since A and X describe the same item, they are inevitably affected by some (hidden) factor Z, e.g., item exposure, the intention of the video creator. When learning recommender models on the interaction data, clearly, the effect of A and Z will be counted in the model prediction. Based on the above analysis, the structural causal model for real-world item feature-based recommendation is formulated as the causal graph in Figure 3.1.

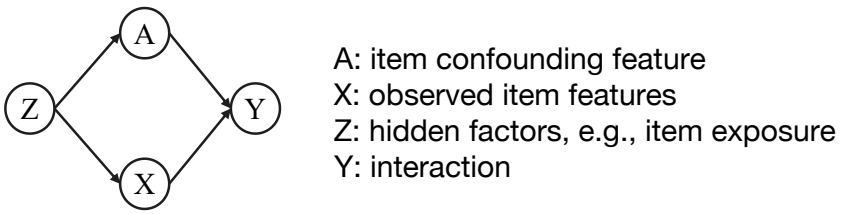


Figure 3.1: Example causal graph for item feature-based recommendation. $X \leftarrow Z \longrightarrow A \longrightarrow Y$ is the backdoor path that brings spurious correlations between X and Y.

The next step is to use the backdoor criterion to scrutinize Figure 3.1. Notably, the feature A opens the backdoor path $X \leftarrow Z \longrightarrow A \longrightarrow Y$, bringing some spurious correlations between X and Y. Formally,

Definition 3.1.5 (Backdoor Path)**.** Given a pair of treatment and outcome variables (t, y), we say a path connecting t and y is a back-door path for (t, y) iff it satisfies that (1) it is not a directed path, and (2) it is not blocked (it has no collider).

Definition 3.1.6 (Backdoor Criterion)**.** Given a treatment-outcome pair (t, y), a set of features $\boldsymbol{x}$ satisfies the back-door criterion of (t, y) iff conditioning on $\boldsymbol{x}$ can block all back-door paths of (t, y).

Clearly, $X \longleftarrow Z \longrightarrow A \longrightarrow Y$ forms a backdoor path as defined in Definition 3.1.5, where the presence of A introduces confounding bias in model estimation. Given these

observations, it is essential to design a recommendation model that eliminates the influence of A by blocking the backdoor path, ensuring unbiased and reliable predictions.

3.1.2 Three Typical Causal Structures

As shown in Figure 3.2, there are three classic structures in causal graphs: *chain*, *fork*, and *collider*, for each of which we give an example of recommender systems.

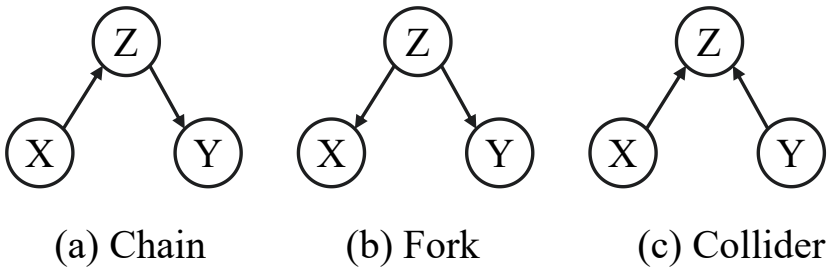


Figure 3.2: Illustration of three typical DAGs.

- **Chain:** In the chain structure, X affects Y via the mediator Z. In the recommendation scenario, the user features affect the user preferences and the user preferences affect the user's click behavior. Thus, the causal relation, *user features* → *preference* → *click*, forms up the chain structure.
- **Fork:** In the fork structure, Z affects both X and Y. In the recommendation scenario, the quality of an item can affect the item price and the preferences of users towards the item, i.e., *price* ← *quality* → *preference*.
- **Collider:** In the collider structure, Z is affected by both X and Y. For example, both user preference and item popularity affect users' clicks, i.e., *preference* → *click* ← *popularity*.

The next section discusses bias issues related to the three typical causal structures.

3.2 Bias Issues in Causal Structures

To relate causal structures to biases, the foundation is to understand the concept of dependency-separation (i.e., d-Separation), which is the base for conditional independence. The d-separation is a criterion used to determine whether a set X of variables is

independent of another set Y, given a third set Z. The idea is to associate the "dependence" of variables with the "connectedness" (i.e., a connecting path) within the causal graph. A path is a sequence of consecutive edges, regardless of their directionality. We consider the flow of dependency between variables connected by paths such as *blocked* when certain conditions are met. In the case of a chain (Figure 3.2 (a)) or a fork (Figure 3.2 (b)), the path between X and Y will be blocked by conditioning to Z. In a collider (Figure 3.2 (c)), conditioning on Z introduces a correlation between X and Y. The formal definition of d-separation or blocking is defined as follows.

Definition 3.2.1 (D-Separation (Block)). A path is said to be d-separated (or blocked) by conditioning on a set of nodes A if and only if one of the two conditions is satisfied:

1. The path contains a chain $X \to Z \to Y$ or a fork $X \leftarrow Z \to Y$ such that the middle node Z is in A;
2. The path contains a collider $X \to Z \leftarrow Y$ such that the middle node Z is not in A and such that no descendant of Z is in A.

In conclusion, X is d-separated from Y given Z if Z blocks all paths between X and Y. In other words, d-separation means that you cannot go from X to Y without passing through a chain or fork whose middle node is in Z or passing through a collider whose middle node (or any of the descendants of the middle node) is not in Z.

The next sections will introduce two types of bias issues relating to the fork and collider structure, respectively. Note that the chain structure does not cause bias issues, as there are propagated causal effects from X, Z to Y, even though $X \to Y$ is blocked by Z. That is, the influence of X on Y is mediated entirely by Z.

3.2.1 Confounding Bias in Fork Structure

Confounding bias can occur in a fork structure (a.k.a., a confounder structure) within a causal graph, i.e., $X \leftarrow Z \to Y$. Here, Z is a common cause (*confounder*) of both X and Y. This structure can lead to confounding bias when trying to estimate the causal effect of X on Y. Because Z affects both X and Y, there can be a *spurious* association between X and Y. This means that X and Y can appear to be related even though X does not cause Y. The main definitions of *confounder* and *confounding bias* are given below.

Definition 3.2.2 (Confounder (Confounding Variable)). Given a pair of treatment and outcome variables (x, y), we say a variable $z \notin \{x, y\}$ is a confounder (confounding variable) iff. it is the central node of a fork, and it blocks (d-separate) the path $x \to y$.

Definition 3.2.3 (Confounding Bias (Confoundness)). Given variables x, y, confounding bias (confoundness) exists for causal effect $x \rightarrow y$ due to the presence of the confounder z. As a result, the statistical association between x and y is not always the same as the corresponding interventional distribution, namely $\mathbb{P}(y \mid x) \neq \mathbb{P}(y \mid do(x))$.

The interventional distribution $\mathbb{P}(y \mid do(x))$ will be discussed in the next section. The following running example in the recommendation scenario is presented to better understand the confounding bias issue.

Running Example. Consider a movie recommendation scenario, we want to understand the effect of a user's viewing history on their likelihood to like a new movie. We have:

- X (Treatment): User's viewing history (e.g., has watched several action movies)
- Y (Outcome): User's rating of a new movie (e.g., likes the new action movie)
- Z (Confounder): User's genre preference (e.g., prefers action movies)

The causal relationships can be represented in a fork structure: $X \leftarrow Z \rightarrow Y$. Here, Z (user's genre preference) is a common cause of both X (user's viewing history) and Y (user's rating of a new movie). This means that the user's genre preference influences both the types of movies they have watched in the past and their likelihood of liking a new movie in that genre. When estimating the causal effect of X on Y, without accounting for the user's genre preference (Z), we might incorrectly infer that watching many action movies (X) directly causes the user to rate a new action movie highly (Y). However, the user's rating of the new movie (i.e., Y) actually stemmed from the user's genre preference Z instead of X. In such a scenario, the issue of confounding bias arises due to the presence of the confounder Z.

3.2.2 Colliding Bias in Collider Structure

The colliding bias (colliding effect) occurs when two variables that do not directly influence each other become conditionally dependent when a third variable, their common effect, is controlled or conditioned on. This effect can be understood through the concept of user conformity bias [262].

Running Example. Consider a book recommendation scenario, we are interested in understanding the factors that influence a user's interest in a new book. We have:

- X (User Interests): Represents the genuine preferences of the user for certain books or content.

- Y (User Conformity): Represents the tendency of a user to interact with books because they are popular or recommended by others, not necessarily because of personal interest.
- Z (User Interactions): The observed behavior of the user, such as clicks, ratings, or purchases, which are influenced by both user interests and conformity.

The causal relationships can be represented in a collider structure as $X \rightarrow Z \leftarrow Y$, as both the interests of the user X and the conformity of the user Y affect the user interaction Z. When we condition on high ratings (user interactions Z), we might mistakenly infer that users who rate this book highly are also genuinely interested in similar books. However, their high rating might be purely due to conformity. This is how colliding bias occurs due to the presence of two common causes of the variable of interest. Some existing literature formulates such a colliding bias in the book recommendation scenario as the *user conformity bias*. Zheng et al. [262] disentangles interest and conformity representations by training on cause-specific data, improving the robustness of user representations.

Any fully specified SCM induces a collection of distributions known as the Pearl Causal Hierarchy (PCH) [154]. The PCH delimits the ladder of causation among three cognitive layers that are associated with the human activities of "seeing" (layer 1 - *association*), "doing" (layer 2 - *intervention*), and "imagining" (layer 3 - *counterfactual*). Each of these layers can be represented as a distinct formal language, and they correspond to queries that help classify various types of causal inferences. The next sections will elaborate on layer 2 - *intervention* and layer 3 - *counterfactual*. Note that layer 1 - association refers to the correlation learning in machine learning models, which is outside the scope of causal inference.

3.3 Intervention

The intervention aims to answer the cognitive question: "*How much would some variables (features or labels) change if we manipulated the value of another variable*?"

The do-calculus $do(\cdot)$, a significant contribution of Pearl's SCM framework, enables researchers to perform interventions by controlling the value of a variable purely through mathematical means rather than physical experiments. For a variable Z, $do(Z = z)$ is to forcefully and externally assign a certain value to the variable Z. Performing the do-calculus $do(Z = z)$ removes any edge from Z's causes to Z, that is, it blocks the effect

of Z's causes on Z. For example, if we apply $do(Z = z)$ to the SCM in Figure 3.3 (a), the SCM can then be transformed into Figure 3.3 (b), which blocks the path $X \rightarrow Z$.

(a) SCM before intervention (b) SCM after intervention

Figure 3.3: Examples of the intervention and do-calculus.

The distribution of the outcome Y after intervention is called the *interventional distribution*, which is expressed as $\mathbb{P}(Y \mid do(Z))$.

3.4 Counterfactual

Counterfactual learning investigates alternative outcomes to answer the "what-if" question: "*what would have happened had some of the initial conditions been different*?" Answering the counterfactual question is naturally challenging. For example, considering the counterfactual learning in the recommendation scenario, we want to know what the user's interaction would be if the recommended items had been different. We should compare the user's interaction under the two opposite conditions, i.e., the item is recommended and non-recommended. The non-recommended condition, which is termed a counterfactual, does not exist in reality and cannot be inferred by do-calculus. Fortunately, Pearl introduces counterfactual notations: $\mathbb{P}(Y(T = 1) \mid T = 0, Y = Y(T = 0))$ indicates the probability of the outcome $Y(T = 1)$ given that the observed treatment value is $T = 0$, and we have observed $Y = Y(T = 0)$ in the data.

In the RecSys domain, there are three wildly used concepts drawn on Pearl's counterfactual notions, i.e., counterfactual explanation, counterfactual explainable fairness and counterfactual estimator.

3.4.1 Counterfactual Explanation

The counterfactual explanation is defined as a minimal perturbation set of the original sample that, if applied, results in the desired prediction of the sample [7, 141]. Formally,

Definition 3.4.1 (Counterfactual Explanation). Suppose that a prediction function $f_R : \mathbb{R}^d \to \mathcal{Y}$ is given. The counterfactual explanation is a minimal perturbation set Δ on the original input $x \in \mathbb{R}^d$. The instance $x_{cf} \in \mathbb{R}^d$ is obtained by applying Δ on x, and is termed the counterfactual instance for x. The solution to the counterfactual explanation Δ for a given input x is computed through an optimization problem:

$$\underset{x_{cf} \in \mathbb{R}^d}{\arg\min} \quad \mathcal{L}(\Delta) = \ell\left(f_R\left(x_{cf}\right), y\right) + C \cdot \theta\left(x_{cf}, x\right) \tag{3.2}$$

where $\mathcal{L}(\Delta)$ is the optimization loss for the counterfactual explanation Δ. $\ell(\cdot)$ denotes a suitable loss function, for example, mean squared error [224], comparing the deviation of the desired prediction y from the counterfactual prediction $f_R(x_{cf})$. $\theta(\cdot)$ is a penalty term for deviations of x_{cf} from the original input x, that is, to encourage Δ as minimal perturbation. $C > 0$ denotes the regularization strength. We refer to a counterfactual explanation Δ and a counterfactual instance x_{cf} given x and $y \sim f_R$ as any solution to the optimization problem.

3.4.2 Counterfactual Explainable Fairness

Fairness explanation aims to fundamentally understand why a model is unfair, i.e., "*what causes unfair recommendation results*?" Counterfactual explainable fairness methods learn fairness explanations as the "minimal" change of the input features that guarantee the reduction of unfairness [55, 197]. They provide "what-if" explanations to determine the most vital and essential (i.e., minimal) factors that change model fairness [197]. Unlike factual explainable methods with high complexity, counterfactual explanations have the advantage of being minimal w.r.t. the generated explanations and are faithful to model fairness changes. Formally,

Definition 3.4.2 (Counterfactual Explainable Fairness). Given a recommender system $f_R : \mathbb{R}^d \to \mathcal{Y}$ and a user or item i with attributes A_i (where A_i includes both protected attributes a_i such as gender, race, etc., and non-protected attributes), counterfactual explainable fairness is defined as a minimal set of attributes $A_i' \in A_i$ that if applied A_i', the fairness disparity of the recommendation result reduced compared with the original recommendation result.

3.4.3 Counterfactual Estimator

The goal of off-policy learning is to maximize the satisfaction of each user with the system by a target policy π_θ, given the historical data logged generated by the historical logging

policy π_0. To achieve this goal, we have to address the counterfactual question: "*how much reward would be received if the new target policy π_θ had been implemented instead of the original logging policy π_0*?" This counterfactual question is not easy to address, as the target policy π_θ is different from the historical logging policy π_0 in the off-policy setting [250]. Counterfactual estimator corrects the policy distribution shift caused by the policy discrepancy through *Counterfactual Risk Minimization* (CRM) [188]. Formally,

Definition 3.4.3 (Counterfactual Risk Minimization). Counterfactual risk minimization (CRM) applies inverse propensity scores (IPS) to the expected cumulative rewards $R(\pi_\theta)$ calculated from the target policy π_θ, so that the distribution shift between π_θ and π_0 is down-weighted.

3.5 Causal Approaches for Recommendation Research Questions

3.5.1 Data Bias

The data bias is caused by the *confoundness* in data, following Definition 3.2.3. The most notable data bias issues in RecSys refer to the *exposure bias* [72] and *popularity bias* [264]. The next sections discuss the reasons why exposure bias and popularity bias arise and present effective causal inference methods that address the two bias issues.

3.5.1.1 Exposure Bias

The power of a recommender system is highly dependent on whether the observed user feedback on items "correctly" reflects the user's preference or not. However, the observed user feedback suffers from the missing issue that needs to be resolved to achieve high-quality recommendations [85, 181, 219]. To handle partially observed feedback, a common assumption for model building is that feedback is *missing at random* (MAR), i.e., the probability of a rating being missing is independent of the value. For example, traditional methods [90] assign a uniform weight to down-weight the missing data, assuming that each missing entry is equally likely to be negative feedback.

However, the MAR assumption usually does not hold in reality, and the missing pattern exhibits the phenomenon *missing not at random* (MNAR). For example, on movie rating websites, highly rated movies are less likely to be missing compared to movies

with lower ratings. The prior study [25] offers compelling evidence to show MNAR can be attributed to *exposure bias* when dealing with implicit feedback. Formally,

Definition 3.5.1 (Exposure Bias). Exposure bias happens as users are only exposed to a part of specific items so that unobserved interactions do not always represent negative preference.

Particularly, the missing pattern of data mainly depends on whether the users are exposed to the items. This may be attributed to the fact that users are only exposed to a part of specific items, so unobserved interactions do not always represent negative preferences. Thus, how to model the missing data mechanism and alleviate the influence brought by the exposure bias becomes a critical research question.

Propensity Score for Exposure Bias. A Recent emerging method, called the *causal debiasing* method, resorted to *propensity score* in causal inference to tackle the MNAR issue [120, 171, 221]. The propensity score introduced in causal inference indicates *the probability that a subject is receiving the treatment or action*. Exposing a user to an item in a recommender system is analogous to exposing a subject to a treatment. Accordingly, the causal debiasing method adopts propensity scores to model the exposure probability and re-weight the prediction error for each observed rating with the inverse propensity score (IPS) [5]. The ultimate goal of using the IPS is to calibrate the MNAR feedback into missing-at-random ones that can be used to guide unbiased rating prediction [130]. Particularly, the propensity score represents the probability of receiving the treatment given covariates X, and it is formulated as follows,

$$e_\pi(X) = \mathbb{P}_\pi(T = 1 \mid X) \tag{3.3}$$

Then, the model uses IPS to assign each sample a weight w based on the propensity score $e(x)$:

$$w = \frac{t}{e(x)} + \frac{1-t}{1-e(x)} \tag{3.4}$$

which represents the inverse likelihood of receiving the observed treatment and control.

There are abundant works that use IPS for causal debiasing. A notable work by Wang et al. [221] views MNAR as the source of confoundness and addresses the confounder problem using propensity scores. Their approach involves first constructing an exposure model to estimate propensity scores. These scores are then used to estimate substitutes for the unobserved confounders. The final outcome model is a rating model based on matrix factorization and is trained conditional on these estimates. Schnabel et al. [171]

compute the propensity from user ratings or indirectly through user and item covariates, and propose an empirical risk minimization approach to learning the unbiased estimators from biased rating data. Alternatively, Liang et al. [120] capture the propensity score using user exposure, i.e., what the user sees. Then, the inverse propensity score is leveraged to train a click model (i.e., what the user clicks on) via a Bayesian model to correct exposure bias. These works re-weight the observational click data as though it came from an "experiment" where users are randomly shown items.

3.5.1.2 Popularity Bias

As discussed in Section 1.2.1, modern recommendation models are largely side-information-aware. They incorporate diverse side information such as social networks [191], knowledge graphs [204], and heterogeneous information networks [174] into the data source for training the recommendation models. The side information provides rich context (e.g., user and item demographics) to capture complex user behavior patterns, promoting recommendation accuracy by a large margin. However, the context is unevenly distributed among auxiliary sources, resulting in *popularity bias*. Formally,

Definition 3.5.2 (**Popularity Bias**). Popular items are recommended even more frequently than their popularity would warrant.

For instance, in a Heterogeneous Information Network (HIN), item attributes (e.g., brand, type) are not evenly distributed. Some items have popular attributes (e.g., the brand Nike), attracting more attention from the recommender system. Conversely, items with less popular attributes receive less attention. This skewed distribution of context (e.g., attributes) can bias the prediction model towards items with popular attributes, leading to recommendations based on attribute popularity rather than true user preferences. As a result, only popular items are recommended, potentially undermining user satisfaction as users' actual preferences remain undiscovered.

Disentangle Learning for Popularity Bias. From a causal perspective, users' actual preferences stem from their intent to interact with certain items. For instance, in movie recommendations, users are more likely to watch movies that belong to the action genre. The movie genre is the user's intent to interact with those movies, rather than the popularity of action movies. Therefore, it is crucial to disentangle users' intent from complex auxiliary side information to mitigate the influence of popularity bias.

Disentangle learning shows to be an effective solution to disentangling users' true intents [142]. It aims to learn disentangled representations (latent factors/embeddings)

that are k individual chunked representations to reveal k user intents, where each chunk reveals a piece of user intent, such as the user's preference towards items' brands. Previous studies have demonstrated that disentangled representations are more robust, i.e., enhanced generalization ability can be achieved by uncovering latent factors. Ma et al. [134] propose to differentiate latent factors of learned user and item embeddings into macro and micro ones. The disentangled macro and micro latent factors are highly independent by the defined Kullback-Leibler (KL) divergence term; thus, the developed recommender is less likely to preserve the bias issues of the factors mistakenly. Recent works use disentangled learning to enhance the accuracy of recommendations. For instance, NeuACF [75] proposes to disentangle users' aspect-level latent factors with the collaborative filtering model to improve the top-N recommendation. DisHAN [223] learns disentangled user and item aspects based on different meta-path types in a HIN, these aspect-aware embeddings are then used to guide the top-N recommendation.

Moreover, such disentangled representation is also superior regarding interpretability, which can be directly applied to various recommendation tasks that require rich semantic information. For example, in the transparent advertising task, Adreveal [125] incorporates massive disentangled user behavior logs into their designed contextual model and develops an analysis framework for transparent advertising. In the explainable recommendation task, TriRank [79] enriches the user-item binary relation to a user-item-aspect ternary relation by incorporating disentangled items' aspect embeddings and provides the explanation on identifying what aspects of users cared most about. EFM [258] disentangles users' opinions about various aspects of products from the reviews and then uses this disentangled information to generate explanations.

3.5.2 Model Accuracy

This section introduces the causal neural network that uses causal relations to refine the neural network architecture, and counterfactual policy learning that designs unbiased model optimization objective for off-policy learning models.

3.5.2.1 Causal Neural Network

According to the neural-causal connection [231] theory, the connection between deep neural networks and causal models is done by establishing a neural causal model.

Definition 3.5.3 (Neural-Causal Connection)**.** A *Neural Causal Model* [231] is defined as $\mathscr{M}(\theta)$ and is parameterized for the structual causal model $\mathscr{M}$ in Definition 3.1.1.

Each structural equation in $\mathscr{M}$ is defined as a feedforward neural network in $\mathscr{M}(\theta)$, e.g., Multi-layer perceptron (MLP). Exogenous variables Z are mapped into hidden vectors $\mathbf{Z}$ that follow the Gaussian distribution $\mathscr{N}(0, \mathbf{I}_K)$.

The NCM $\mathscr{M}(\theta)$ is expressive [231], such that it generates distributions associated with the Pearl Causal Hierarchy (PCH) [155], i.e., modeling "observational" (layer 1), "interventional" (layer 2) and "counterfactual" (layer 3) distributions.

3.5.2.2 Counterfactual Policy Learning

Counterfactual risk minimization(CRM) [188] corrects the discrepancy between the target policy π_θ and logging policy π_0, thus to answer the counterfactual question: "*how much reward would be received if a new target policy π_θ had been deployed instead of the original logging policy π_0?*" The target policy π_θ is optimized to maximize user satisfaction by the cumulative rewards:

$$R(\pi_\theta) = \mathbb{E}_{s_0 \sim \rho(s), a_t \sim \pi_\theta(a|s_t), s_{t+1} \sim \mathbb{P}(s|s_t,a_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{3.5}$$

where $\rho(s)$ is the initial distribution of user states, $\mathbb{P}(s \mid s_t, a_t) \in \mathscr{P}$ is the state transition probability. $r(s_t, a_t)$ is the reward (for example, clicks or time to watch) for a particular pair of state-actions $s_t - a_t$. γ^t is the decay factor for a reinforcement learning step $t \in T$. Following Definition 3.4.3, inverse propensity scores (IPS) are applied to the cumulative rewards in Equation (3.5) to correct the discrepancy between π_θ and π_0.

However, the IPS estimator suffers from the "propensity overfitting" issue due to the uncertainty on rare actions [95, 133]; when directly optimizing IPS within a learning algorithm, the results tend to have a large variance. To reduce variance, we can resort to a clipped estimator that caps the propensity ratios (that is, importance weight) to a maximum value [17]. The core idea is to regulate large weights necessarily associated with actions that are different to the logging policy. The clipped estimator (cIPS) is represented as,

$$L_{\text{cIPS}}(\pi_\theta) = \frac{1}{T} \sum_{t=1}^{T} r_t \min \left\{ \frac{\pi_\theta(a_t \mid s_t^{u \to a})}{\pi_0(a_t \mid s_t^{u \to a})}, c \right\} \tag{3.6}$$

where c is a constant that serves as the regulator for constraining the importance weight $\frac{\pi_\theta(a_t|s_t)}{\pi_0(a_t|s_t)}$ to at most c, smaller value of constant c reduces variance in the gradient estimate but introduces larger bias. We thus follow Joachims et.al [97] to prevent the additional

bias by adding an empirical variance penalty term λ as,

$$L^{\lambda}_{\text{cIPS}}(\pi_\theta) = \frac{1}{T}\sum_{t=1}^{T}(r_t - \lambda_t)\min\left\{\frac{\pi_\theta\left(a_t \mid s_t^{u\to a}\right)}{\pi_0\left(a_t \mid s_t^{u\to a}\right)}, c\right\} \tag{3.7}$$

where λ_i regulates the corresponding reward r_i at each interaction. By plugging Equation (3.7) into objective function Equation (3.5), the optimization function becomes:

$$\begin{aligned} R(\pi_\theta) &= \mathbb{E}_{\pi_\theta}\left[\gamma^t L^{\lambda}_{\text{cIPS}}(\pi_\theta)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t\left(r(s_t^{u\to a}, a_t) - \lambda_t\right)\min\left\{\frac{\pi_\theta\left(a_t \mid s_t^{u\to a}\right)}{\pi_0\left(a_t \mid s_t^{u\to a}\right)}, c\right\}\right] \end{aligned} \tag{3.8}$$

Using the learning objective function in Equation (3.8), the recommendation model can achieve counterfactual policy learning that is free of bias.

3.5.3 Model Explainability and Fairness

3.5.3.1 Counterfactual Explanation

Counterfactual explanations have been considered satisfactory explanations [228] and elicit causal reasoning in humans [19]. Successful works on the counterfactual explanation for recommendations can be categorized into search-based and optimization-based approaches. The first category of search-based approaches [98, 236] performs the greedy search for counterfactual explanations. Kaffes et al. [98] perturb user-item interactions by deleting items from user interaction queues to generate perturbed user interactions as counterfactual explanations. Then, a breadth-first search is used to find counterfactual explanations that achieve the highest normalized length and candidate impotence scores. Xiong et al. [236] propose constrained feature perturbations on the features of items and consider the perturbed item features as counterfactual explanations. The second category of optimization-based approaches [34, 61, 189, 193] optimizes explanation models to find counterfactual explanations with minimal changes. Ghazimatin et al. [61] perform random walks over a heterogeneous information network and calculate the PageRank scores after removing user action edges from the graph. Those minimal sets of user actions that change PageRank scores are deemed counterfactual explanations. Tran et al. [193] identify a minimal set of user actions that updates the parameters of neural models. Tan et al. [189] modify item aspect scores to observe user preference changes based on a user-aspect preference matrix.

Existing approaches either focus on user action [34, 61, 193] or item aspect explanations [189]. However, item attribute-based counterfactual explanations using real-world item demographics are unexplored.

Item Attribute-based Counterfactual Explanations. Item attribute-based counterfactual explanations could benefit both users' trust and recommendation performance. This is because item attribute-based counterfactual explanations are usually more intuitive and persuasive when seeking users' trust. For example, users prefer to know detailed information, e.g., which item attribute caused the film "Avatar" not to be recommended anymore? Is it the director of "Avatar"?

This thesis focuses on finding the attribute-based counterfactual explanations, in which the minimal perturbation set Δ is defined as a set of item attributes, e.g., *genre*, *brand*. Following Definition 3.4.1, we can define f_R as a Top-K recommendation model that gives the recommendation list $\boldsymbol{Q}_u$ with length K for a user u, and we say $i \in \boldsymbol{Q}_u$ if item i is recommended. For a user-item pair (u,i), the aim is to search for a minimal item attributes set $\Delta_{ui} = \{a^1_{ui}, \cdots, a^r_{ui}\}$. Each $a^i_{ui} \in \Delta_{ui}$ is an attribute entity selected from a collaborative knowledge graph, which contains real-world semantics that describes the item. With f_R and Δ_{ui}, our optimization goal is to estimate whether applying the Δ_{ui} on the original recommended item i results in replacing the i with a counterfactual item j. If the optimization goal is met, Δ_{ui} is termed an *attribute-based counterfactual explanation* that flips the recommendation result, and j is the *counterfactual item* for the original item i. The above intuition is formalized as the following definition:

Definition 3.5.4 (Item Attribute-based Counterfactual Explanation). Given f_R as the prediction function of a Top-K recommendation model, and the original recommended item $i \in \boldsymbol{Q}_u \sim f_R$ for a user u. An attribute-based counterfactual explanation for (u,i) is defined as Δ_{ui}, such that applying the Δ_{ui} on i results in replacing the i with a counterfactual item j for user u. Meanwhile, the counterfactual explanation Δ_{ui} is *minimal* such that there is no smaller set Δ'_{ui} satisfying $|\Delta'_{ui}| < |\Delta_{ui}|$ when Δ'_{ui} also meets the optimization goal. With the optimized Δ_{ui}, we can generate the counterfactual explanation for recommending item i to user u, which takes the following form:

> Had a minimal set of attributes [Δ_{ui}(s)] been different for item i, the recommended item would have been j instead.

3.5.3.2 Counterfactual Explainable Fairness

To date, there is only one counterfactual explainable method for recommendation fairness. Ge et al. [56] explain which item feature changes the item exposure fairness of the feature-based recommendation models. They perturb feature scores within pre-defined user-aspect and item-aspect matrices and feed the perturbed matrices into a recommendation

model. Those perturbed features that change the fairness disparity are considered fairness explanations. However, the method proposed by Ge et al. [56] is primarily designed for feature-based models and is not well-suited for handling discrete attributes in attribute-aware recommendation models.

Including context-based filtering [174, 176] and knowledge-aware systems [204], modern recommendation models are largely attribute-aware and rely on explicit attributes to generate recommendations. Those attributes are user and item demographics, which are generated through data tagging [68] on discrete attributes, e.g., genre, language, and location. However, the feature-level optimization in [56] can only deal with continuous values and cannot be directly applied to discrete attributes. For example, assigning a continuous value, such as *gender*=0.19, to the discrete *gender* attribute is impractical in constructing explanations and provides no valuable clue to improve the explanation. Consequently, feature-level optimizations have limited capability in handling discrete attributes that are frequently encountered in recommendation scenarios.

Attribute-level Counterfactual Fairness Explanations. Unlike previous studies, this thesis focuses on exploring *attribute-level counterfactual explanations* for attribute-aware recommendation models. The question is to answer "*what the fairness would be if a minimal set of attributes had been different*?" Driven by recent successes in Heterogeneous Information Networks (HINs)-enhanced recommendations [88, 174, 176], this thesis proposes to leverage real-world attributes from a HIN for counterfactual reasoning when dealing with discrete attributes. In contrast to value-based features, real-world attributes residing in HINs are represented as discrete nodes, with edges denoting their connections. By utilizing attributes from HINs, we can overcome the limitation of feature-level optimizations to directly measure whether the removal of specific attributes changes the model's fairness.

In particular, given user-item interaction $\boldsymbol{Y}$, user attribute set $\mathcal{V}_U$, and item attribute set $\mathcal{V}_I$ extracted from an external Heterogeneous Information Network (HIN) $\mathcal{G}$. Suppose that there exists a recommendation model that produces the recommendation result $H_{u,K}$ for user u. Given all user-item pairs (u,v) in $H_{u,K}$, the goal is to find a minimal attribute set $\mathcal{V}^* \subseteq \{\{e_u, e_v\} \mid e_u \in \mathcal{V}_U, e_v \in \mathcal{V}_I\}$. Each attribute in $\mathcal{V}^*$ is an attribute entity from HIN $\mathcal{G}$, e.g., the user's gender and item's genre. With a minimal set of $\mathcal{V}^*$, counterfactual reasoning seeks to answer: What would the fairness disparity be if $\mathcal{V}^*$ is applied to the recommendation model. $\mathcal{V}^*$ is recognized as valid *counterfactual explanation for fairness*, if after applied $\mathcal{V}^*$, the fairness disparity of the intervened recommendation result $\Delta(H^{cf}_{u,K})$ reduced compared with original $\Delta(H_{u,K})$. In addition, $\mathcal{V}^*$ is *minimal* such

that there is no smaller set $\mathcal{V}^{*'} \in \mathcal{G}$ satisfying $|\mathcal{V}^{*'}| < |\mathcal{V}^{*}|$ when $\mathcal{V}^{*'}$ is also valid. Formally, attribute-level counterfactual fairness explanations are defined as,

Definition 3.5.5 (Attribute-level Counterfactual Fairness Explanations). For item exposure unfairness in recommendations, given the recommendation result $H_{u,K}$, counterfactual explanations at the attribute level aim to find the "minimal" changes in attributes that reduce the fairness disparity $\Delta(H_{u,K})$ of item exposure.

CHAPTER 4

ALLEVIATE DATA BIASES USING CAUSAL INFERENCE

This chapter introduces two research works: DENC model (De-bias Network Confounding in Recommendation) that mitigates the exposure bias under the missing not at random (MNAR) phenomenon, and the CaDSI model (Causal Disentanglement for Semantics-Aware Intent Learning) that mitigates popularity bias under context information-aware recommendation scenario.

4.1 Exposure Bias Mitigation with Propensity Scores

4.1.1 Overview

Research Question. Exposure bias happens when the observational data exhibits missing not at random (MNAR) phenomenon. Particularly, exposure bias occurs because users are free to choose which items to rate, so the observed ratings are not representative of all ratings. That might be because users are only exposed to a part of specific items, so unobserved interactions do not always represent negative preference. How to model the missing data mechanism and debias the rating prediction performance forms the main research question of this work.

Research Objective. Traditional methods [90] use the uniformity assumption to combat exposure bias, which entails assigning a uniform weight to missing data by assuming that each missing entry is equal to negative feedback. This is a strong assumption that limits the adaptability of models in real-world scenarios. Thus, instead of simply performing

shallow-level data re-weighting measures, the objective of this research is to develop a robust and unbiased rating prediction model by approaching the fundamental exposure bias issue from a novel causal inference perspective. As introduced in Section 3.2.1, the exposure bias is attributed to the presence of confounders that affect both the treatment assignments (i.e., exposure) and the outcomes (i.e., rating). Towards this end, the objective is to analyze the causal effect of confounders on rating and exposure and, in turn, fundamentally eliminate the exposure bias to predict valid ratings.

Motivations. This research work aims to address the MNAR issue in recommendation using causal inference, to attain a robust and unbiased rating prediction model. From a causal perspective, the exposure bias in the recommender system is attributed to the presence of confounders. As explained in Figure 4.1, confounders are factors (or variables) that affect both the treatment assignments (exposure) and the outcomes (rating). For example, friendships (or social network) can influence both users' choice of movie watching and their further ratings. Users tend to consume and rate the items that they like and the items that have been consumed by their friends. So, the motivation is the social network is indeed a confounding factor that affects which movie the user is exposed to and how the user rates the movie. The confounding factor results in a distribution discrepancy between the partially observed ratings and the complete ratings, as shown in Figure 4.2. Without considering the distribution discrepancy, the rating model trained on the observed ratings fails to generalize well on the unobserved ratings. Thus, our idea is to analyze the confounder effect of social networks on rating and exposure and fundamentally alleviate the MNAR problem to predict valid ratings.

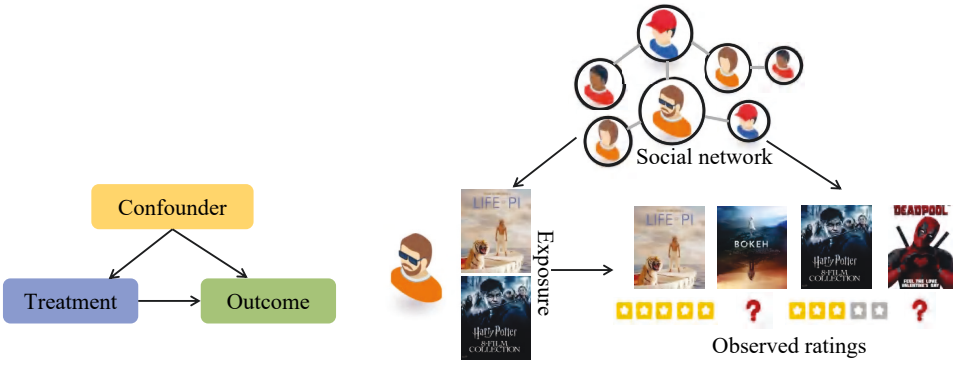


Figure 4.1: The causal view for MNAR problem: treatment and outcome are terms in the theory of causal inference, which denote an action taken (e.g., exposure) and its result (e.g., rating), respectively. The confounder (e.g., social network) is the common cause of treatment and outcome.

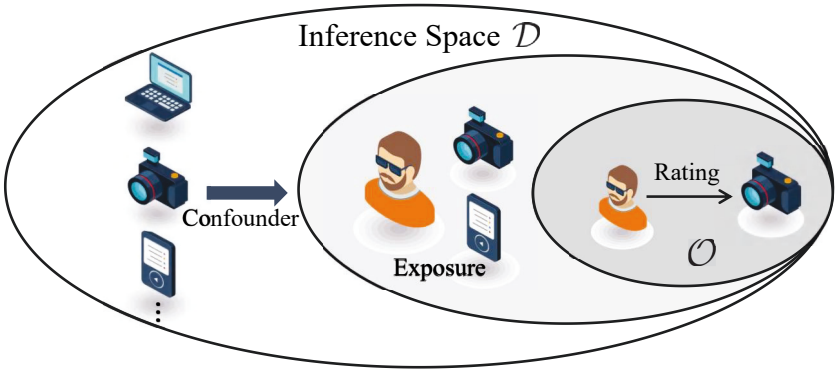


Figure 4.2: The training space of conventional recommendation models is the observed rating space $\mathcal{O}$, whereas the inference space is the entire exposure space $\mathcal{D}$. The discrepancy of data distribution between $\mathcal{O}$ and $\mathcal{D}$ leads to exposure bias in conventional recommendation models.

The Proposed Approach. We propose an unbiased and robust method called DENC (De-bias Network Confounding in Recommendation). To sufficiently consider the exposure bias in MNAR, we model the underlying factors (i.e., inherent user-item information and social network) that can generate observed ratings. In light of this, as shown in Figure 4.4, we construct a causal graph-based recommendation framework by disentangling three determinants for the ratings, i.e., *inherent factors*, *confounder* and *exposure*. Each determinant accordingly corresponds to one of three specific components in DENC: *deconfonder model*, *social network confounder* and *exposure model*, all of which jointly determine the rating outcome.

Contributions. In summary, the key contributions of this research are as follows:

- Fundamentally different from previous works, DENC is the first method for unbiased rating prediction through disentangling determinants of exposure bias from a causal view.

- The proposed *exposure model* is capable of revealing the exposure assignment and accounting for the confounder factors derived from the *social network confounder*, which thus remedies exposure bias in a principled manner.

- We develop a *deconfonder model* via the balanced representation learning that embeds inherent factors independent of the exposure, therefore mitigating the distribution discrepancy between the observed rating and inference space.

- We conduct extensive experiments to show that our DENC method outperforms state-of-the-art methods. The generalization ability of our DENC is also validated by verifying different degrees of confounders.

4.1.2 Causal Graph for Exposure Bias

Exposing a user to an item in recommendation is analogous to exposing a patient to treatment in a medical study. In both tasks, we have only partial observations of how much certain users (patients) prefer (benefit from) certain items (treatments). We are interested in the question "*If the user (patient) had exposed (adopted) to other items (treatments), how much would the user (patient) prefer (benefit from)*?". Following this principle, we aim to answer such an interventional question in the MNAR scenario.

Prior to that, we first give the notations. We assume that $Y \in \mathbb{R}^{m \times n} = [\dot{y}_{ui}]$ is the user-item rating matrix, in which $\dot{y}_{ui}$ is the rating given by user u to item i. In addition, for every user-item pair (u, i), we have a binary exposure $a_{ui} \in \{1, 0\}$ indicates that the item i is exposed to user u or not. Let G denote user-user social graph among users where $G_{kj} = 1$ if u_k has a relation to u_j and zero otherwise. Let $N_s(u)$ be the set of users to whom u is directly connected.

Figure 4.3: The causal graph in the recommendation.

We resort to a causal graph to answer the interventional question. As introduced in Definition 3.1.2, the causal graph can describe the generation mechanism of recommendation results and guide the design of recommendation methods. In our work, we investigate social networks as a confounder that is a common cause of item exposure A and rating Y. In particular, we abstract a structural causal graph, as shown in Figure 4.3, to explicitly analyze the causal relations in the conventional recommender system. The causal graph consists of four variables: confounder Z, exposure A, inherent factor I and rating Y. Every directed edge represents a causal relation between two variables. The rationality of causal relations in Figure 4.3 can be explained as follows.

- $Z \rightarrow A$: the social network information of users affects users' choice of movie. For example, a user's social network might affect the movies he is exposed to.
- $Z \rightarrow Y$: a user's social network can affect the user's preference for items. Similarly, social networks can affect how much the user likes the movies he watches.
- $(Z, A) \rightarrow Y$: observed ratings are generated as results of *which items are exposed to user* and *the user's preference for each of those items*.
- $I \rightarrow Y$: inherent factors I affects the recommendation outcome Y. For example, I refers to the inherent factors that are acquired from demographic features of users and items. For example, user ID and item genre.

As indicated by Figure 4.3, the social network is a confounder variable that affects both the user's exposure to items and the user's rating. To resolve the impact of the confounder, we propose a novel approach called DENC to disentangle determinants on rating outcome guided by the causal graph. The overall framework of our DENC is shown in Figure 4.4 and includes three components: *social network confounder*, *exposure model* and *deconfonder model*. In the following, we will elaborate on each component and the debiasing process for rating prediction.

4.1.3 Methodology

4.1.3.1 Social Network Confounder

To control the exposure bias arising from the external social network, we propose a confounder representation model that quantifies the common biased factors affecting both the exposure and rating.

Let G present the social relationships among users U, where an edge denotes there is a friend relationship between users. We resort to node2vec [66] method and learn network embedding from diverse connectivity provided by the social network. To mine the deep social structure from G, for every source user u, node2vec generates the network neighborhoods $N_s(u) \subset G$ of node u through a sampling strategy to explore its neighborhoods in a breadth-first sampling as well as a depth-first sampling manner. The representation Z_u for user u can be learned by minimizing the negative likelihood of preserving network neighborhoods $N_s(u)$:

$$\mathscr{L}_z = -\sum_{u \in G} \log P(N_s(u)|Z_u) = \sum_{u \in G} \left[\log \sum_{v \in G} \exp(Z_v \cdot Z_u) - \sum_{u_i \in N_s(u)} Z_{u_i} \cdot Z_u \right] \tag{4.1}$$

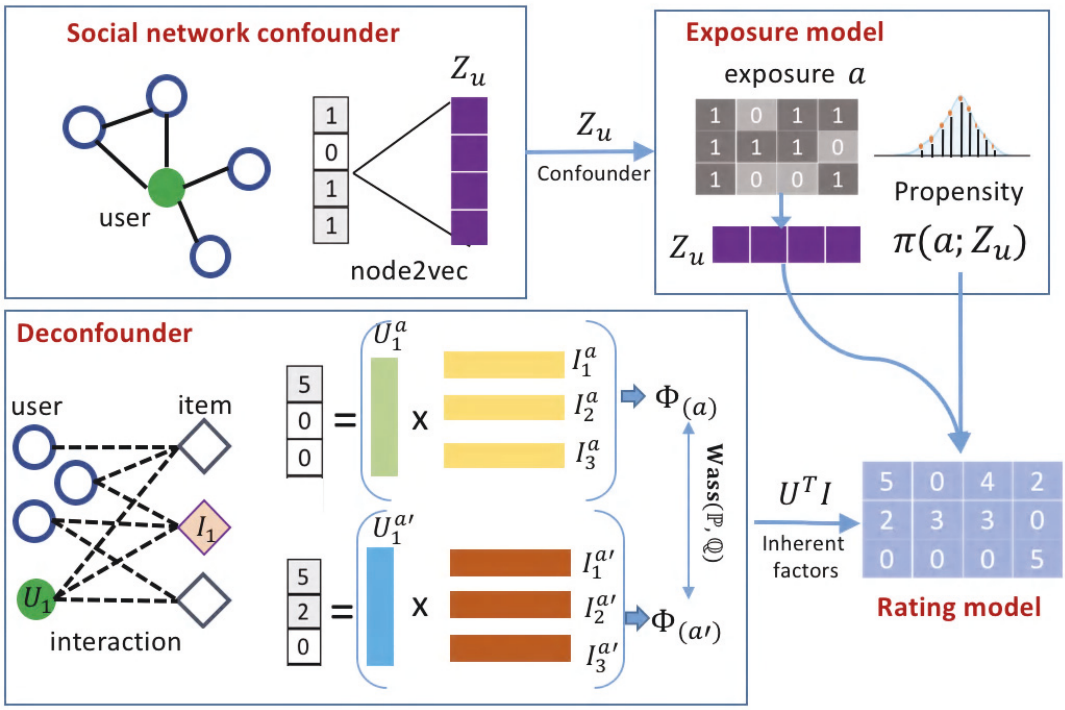


Figure 4.4: Our DENC method consists of social network confounder, exposure model, deconfonder model and rating model.

The final output $Z_u \in \mathbb{R}^d$ explores diverse neighborhoods of each user, which represents to what extent the exposure for a user is influenced by his friends in G.

4.1.3.2 Exposure Model

Guided by the treatment assignment mechanism, we propose a novel exposure model that computes the probability of exposure variables specific to the user-item pair. This model is beneficial to understanding the generation of the MNAR pattern in ratings, which thus remedies exposure biases in a principled manner. We consider the effect of the social network confounder for the treatment assignment, i.e., exposure. For example, the user goes to watch the movie because of his friend's strong recommendation. Thus, we propose to mitigate the exposure bias by exploiting the network connectivity information that indicates *to which extent the exposure for a user will be affected by its neighbors*.

To begin with, we are interested in the binary exposure a_{ui} that defines whether the item i is exposed ($a_{ui} = 1$) or unexposed ($a_{ui} = 1$) to user u, i.e., $a_{ui} = 1$. Based on the informative confounder learned from social network, we propose the notation of *propensity* to capture the exposure from the causal inference language.

Definition 4.1.1 (Propensity). Given an observed rating $y_{ui} \in$ rating and confounder Z_u in (4.1), the propensity of the corresponding exposure for user-item pair (u,i) is defined as,

$$\pi(a_{ui}; Z_u) = P(a_{ui} = 1 | y_{ui} \in \text{rating}; Z_u) \tag{4.2}$$

In view of the foregoing, we model the exposure mechanism by the probability of a_{ui} being assigned to 0 or 1.

$$P(a_{ui}) = \prod_{u,i} P(a_{ui}) = \prod_{(u,i)\in\mathcal{O}} P(a_{ui} = 1) \prod_{(u,i)\notin\mathcal{O}} P(a_{ui} = ?) \tag{4.3}$$

where $\mathcal{O}$ is an index set for the observed ratings. The case of $a_{ui} = 1$ can result in an observed rating or unobserved rating: 1) for the observed rating represented by $y_{ui} \in$ rating, we definitely know the item i is exposed, i.e., $a_{ui} = 1$; 2) an unobserved rating $y_{ui} \notin$ rating may represent a negative feedback (i.e., the user is not reluctant to rating the item) on the exposed item $a_{ui} = 1$. In light of this, based on (4.2), we have

$$\begin{aligned} P(a_{ui} = 1) &= P(a_{ui} = 1, y_{ui} \in \text{rating}) + P(a_{ui} = 1, y_{ui} \notin \text{rating}) \\ &= \pi(a_{ui}; Z_u) P(y_{ui} \in \text{rating}) + W_{ui} P(y_{ui} \notin \text{rating}) \end{aligned} \tag{4.4}$$

where $W_{ui} = P(a_{ui} = 1 | y_{ui} \notin \text{rating})$. The exposure a_{ui} that is unknown follows the distributions as,

$$P(a_{ui} = ?) = 1 - P(a_{ui} = 1) \tag{4.5}$$

By substituting Eq. (4.4) and Eq. (4.5) for Eq. (4.3), we attain the exposure assignment for the overall rating data as,

$$P(a_{ui}) = \prod_{(u,i)\in\mathcal{O}} \pi(a_{ui}; Z_u) \prod_{(u,i)\notin\mathcal{O}} (1 - W_{ui}) \tag{4.6}$$

Inspired by [151], we assume uniform scheme for W_{ui} when no side information is available. According to most causal inference methods [154], a widely-adopted parameterization for $\pi(a_{ui}; Z_u)$ is a logistic regression network parameterized by $\Theta = \{W_0, b_0\}$, i.e.,

$$\pi(a_{ui}; Z_u, \Theta) = \mathbb{I}_{y\in\text{rating}} \cdot \left[1 + e^{-(2a_{ui}-1)(Z_u^\top \cdot W_0 + b_0)}\right]^{-1} \tag{4.7}$$

Based on Eq. (4.7), the overall exposure $P(a_{ui})$ in Eq. (4.6) can be written as the function of parameters $\Theta = \{W_0, b_0\}$ and Z_u, i.e.,

$$\mathcal{L}_a = \sum_{u,i} -\log P(a_{ui}; Z_u, \Theta) \tag{4.8}$$

where social network confounder Z_u is learned by the pre-trained node2vec algorithm. Similar to supervised learning, Θ can be optimized through minimization of the negative log-likelihood.

4.1.3.3 Deconfounder Model

Traditional recommendation learns the latent factor representations for user and item by minimizing errors on the observed ratings, e.g., matrix factorization. Due to the existence of exposure bias, such a learned representation may not necessarily minimize the errors on the unobserved rating prediction. Inspired by [173], we propose to learn a balanced representation that is independent of exposure assignment such that it represents inherent or invariant features in terms of users and items. The invariant features must also lie in the inference space shown in Figure 4.2, which can be used to consistently infer unknown ratings using observed ratings. This makes sense in theory: if the learned representation is hard to distinguish across different exposure settings, it represents invariant features related to users and items.

According to Figure 4.3, we can define two latent vectors $U \in \mathbb{R}^{k_d}$ and $I \in \mathbb{R}^{k_d}$ to represent the inherent factor of a user and a item, respectively. Recall that different values for W_{ui} in Eq. (4.6) can generate different exposure assignments for the observed rating data. Following this intuition, we construct two different exposure assignments a and $\hat{a}$ corresponding two settings of W_{ui}. Accordingly, $\Phi_{(a)}$ and $\Phi_{(\hat{a})}$ are defined to include inherent factors of users and items, i.e., $\Phi_{(a)} = \left[U_1^{(a)}, \cdots, U_M^{(a)}, I_1^{(a)}, \cdots, I_M^{(a)}\right] \in \mathbb{R}^{k_d \times 2M}$, $\Phi_{(\hat{a})} = \left[U_1^{(\hat{a})}, \cdots, U_M^{(\hat{a})}, I_1^{(\hat{a})}, \cdots, I_M^{(\hat{a})}\right] \in \mathbb{R}^{k_d \times 2M}$. Figure 4.3 also indicates that the inherent factors of user and item would keep unchanged even if the exposure variable is altered from 0 to 1, and vice versa. That means $U \in \mathbb{R}^{k_d}$ and $I \in \mathbb{R}^{k_d}$ should be independent of the exposure assignment, i.e., $U^{(a)} \perp\!\!\!\perp U^{(\hat{a})}$ or $I^{(a)} \perp\!\!\!\perp I^{(\hat{a})}$. Accordingly, minimizing the discrepancy between $\Phi_{(a)}$ and $\Phi_{(\hat{a})}$ ensures that the learned factors embeds no information about the exposure variable and thus reduce exposure bias. The penalty term for such a discrepancy is defined as,

$$\mathscr{L}_d = \mathrm{disc}\left(\Phi_{(\hat{a})}, \Phi_{(a)}\right) \tag{4.9}$$

Inspired by [143], we employ *Integral Probability Metric* (IPM) to estimate the discrepancy between $\Phi_{(\hat{a})}$ and $\Phi_{(a)}$. $\mathrm{IPM}_{\mathscr{F}}(\cdot,\cdot)$ is the (empirical) integral probability metric defined by the function family $\mathscr{F}$. Define two probability distributions $\mathbb{P} = P(\Phi_{(\hat{a})})$ and $\mathbb{Q} = P(\Phi_{(a)})$, the corresponding IPM is denoted as,

$$\mathrm{IPM}_{\mathscr{F}}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathscr{F}} \left| \int_S f d\mathbb{P} - \int_S f d\mathbb{Q} \right| \tag{4.10}$$

where $\mathscr{F} : S \to \mathbb{R}$ is a class of real-valued bounded measurable functions. We adopt $\mathscr{F}$ as

1-Lipschitz functions that lead IPM to the Wasserstein-1 distance, i.e.,

$$Wass(\mathbb{P},\mathbb{Q}) = \inf_{f\in\mathscr{F}} \sum_{\mathbf{v}\in \mathrm{col}_i(\Phi_{(\hat{a})})} \|f(\mathbf{v})-\mathbf{v}\|\mathbb{P}(\mathbf{v})d\mathbf{v} \tag{4.11}$$

where $\mathbf{v}$ is the i-th column of $\Phi_{(\hat{a})}$ and the set of push-forward functions that can transform the representation distribution of the exposed $\Phi_{(\hat{a})}$ to that of the unexposed $\Phi_{(a)}$. Thus, $\|f(\mathbf{v})-\mathbf{v}\|$ is a pairwise distance matrix between the exposed and unexposed user-item pairs. Based on the discrepancy defined in (4.11), we define $C(\Phi) = \|f(\mathbf{v})-\mathbf{v}\|$ and reformulate penalty term in (4.9) as,

$$\mathscr{L}_d = \inf_{\gamma\in\Pi(\mathbb{P},\mathbb{Q})} \mathbb{E}_{(\mathbf{v},f(\mathbf{v}))\sim\gamma} C(\Phi) \tag{4.12}$$

We adopt the efficient approximation algorithm proposed by [173] to compute the gradient of (4.12) for training the deconfounder model. In particular, a mini-batch with l exposed and l unexposed user-item pairs is sampled from $\Phi_{(\hat{a})}$ and $\Phi_{(a)}$, respectively. The element of distance matrix $C(\Phi)$ is calculated as $C_{ij} = \|\mathrm{col}_i(\Phi_{(\hat{a})}) - \mathrm{col}_j(\Phi_{(a)})\|$. After computing $C(\Phi)$, we can approximate f and the gradient against the model parameters [1]. In conclusion, the learned latent factors generated by the deconfounder model embed no information about exposure variable. That means all the confounding factors are retained in social network confounder Z_u.

4.1.3.4 Learning

Rating prediction. Having obtained the final representations U and I by the deconfounder model, we use an inner product of $U^\top I$ as the inherent factors to estimate the rating. As shown in the causal structure in Figure 4.4, another component affecting the rating prediction is the social network confounder. A simple way to incorporate these components into recommender systems is through a linear model as follows.

$$\hat{y}_{ui} = \sum_{u,i\in\mathscr{O}} U^\top I + {W_u}^\top Z_u + \epsilon_{ui}, \quad \epsilon_{ui}\sim\mathscr{N}(0,1) \tag{4.13}$$

where W_u is a coefficient that describes how much the confounder Z_u contributes to the rating. To define the unbiased loss function for the biased observations y_{ui}, we leverage the IPS strategy [171] to weight each observation with *Propensity*. By Definition 4.1.1, the intuition of the inverse propensity is to down-weight the commonly observed ratings while up-weighting the rare ones.

$$\mathscr{L}_y = \frac{1}{|\mathscr{O}|} \sum_{u,i\in\mathscr{O}} \frac{(y_{ui}-\hat{y}_{ui})^2}{\pi(a_{ui};Z_u)} \tag{4.14}$$

[1] For a more detailed calculation, refer to Algorithm 2 in the appendix of prior work [173]

Optimization. To this end, the objective function of our DENC method to predict ratings could be derived as follows:

$$\mathscr{L} = \mathscr{L}_y + \lambda_a \mathscr{L}_a + \lambda_z \mathscr{L}_z + \lambda_d \mathscr{L}_d + \mathscr{R}(\Omega) \tag{4.15}$$

where Ω represents the trainable parameters and $\mathscr{R}(\cdot)$ is a squared l_2 norm regularization term on Ω to alleviate the overfitting problem. λ_a, λ_z and λ_d are trade-off hyper-parameters. To optimize the objective function, we adopt Stochastic Gradient Descent(SGD) [16] as the optimizer due to its efficiency.

4.1.3.5 Computational Complexity Analysis

DENC consists of three key components: Social Network Confounder, Exposure Model, and Deconfounder Model. Below, we analyze the time complexity of each component separately and then discuss the overall complexity of DENC. The Social Network Confounder learns network embeddings using node2vec [66], which generates random walk sequences and applies the Skip-Gram model to learn embeddings. The computational complexity of generating r random walks of length l per node for a graph with N nodes and E edges is $O(r \cdot l \cdot N)$. For skip-gram optimization, given a window size w and embedding dimension d, the time complexity of skip-gram with negative sampling is $O(N \cdot d + E \cdot w \cdot d)$. Thus, the overall complexity of Social Network Confounder is $O(r \cdot l \cdot N + N \cdot d + E \cdot w \cdot d)$. The Exposure Model estimates propensity scores to model item exposure probability, in which a logistic regression with the complexity $O(M \cdot K \cdot d)$ is applied over historical interaction data. The Deconfounder Model estimates the discrepancy between exposed and unexposed items using the Integral Probability Metric (IPM). The time complexity depends on the specific divergence measure used. Here, we apply Wasserstein Distance, with a complexity of $O(K^2 \log K)$. The final rating prediction model is implemented via Matrix Factorization (MF), for a user-item matrix of size $M \times K$ with rank r, the complexity is: $O(MKr + r^3)$. Summing the individual complexities; thus, the total time complexity of DENC is: $O(r \cdot l \cdot N + N \cdot d + E \cdot w \cdot d + MKd + K^2 d + MKr + r^3)$.

4.1.4 Experiments

To more thoroughly understand the nature of MNAR issue and the proposed unbiased DENC, experiments are conducted to answer the following research questions:

- **RQ1**. How confounder bias caused by the social network is manifested in real-world recommendation datasets?

- **RQ2**. Does our DENC method achieve the state-of-the-art performance in debiasing recommendation task?
- **RQ3**. How does the embedding size of each component (e.g., social network confounder and deconfounder model) in our DENC method impact the debiasing performance?
- **RQ4**. How do the missing social relations impact the debiasing performance of our DENC method?

4.1.4.1 Setup

Evaluation Metrics. We adopt two popular metrics including Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to evaluate the performance. Since improvements in MAE or RMSE have a significant impact on the quality of the Top-K recommendations [106], we also evaluate our DENC with Precision@K and Recall@K for the ranking performance[2].

Datasets. We conduct experiments on three datasets including one semi-synthetic dataset and two benchmark datasets `Epinions` [3] and `Ciao` [190] [4]. We maintain all the user-item interaction records in the original datasets instead of discarding items that have sparse interactions with users.[5] The semi-synthetic dataset is generated by incorporating the social network into `MovieLens`[6] dataset. The statistics of datasets are summarized in Table 4.1.

Table 4.1: Statistics of datasets.

	`Epinions`	`Ciao`	`MovieLens-1M`
# `users`	22,164	7,317	6,040
# `items`	296,277	104,975	3,706
# `ratings`	922,267	283,319	1000,209
`density-R` (%)	0.0140	0.0368	4.4683
# `relations`	355,754	111,781	9,606
`density-SR` (%)	0.0724	0.2087	0.0263

[2]We consider items with a rating greater than or equal to 3.5 as relevant

[3]http://www.cse.msu.edu/ tangjili/trust.html

[4]http://www.cse.msu.edu/ tangjili/trust.html

[5]Models can benefit from the preprocessed datasets in which all items interact with at least a certain amount of users, for such preprocessing will reduce the dataset sparsity.

[6]https://grouplens.org/datasets/movielens

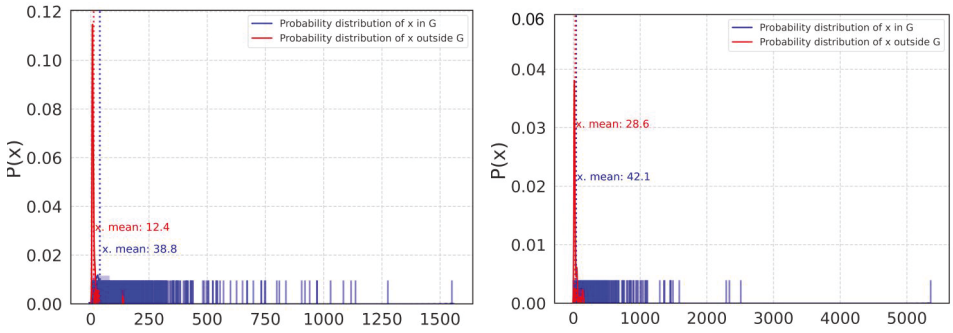
Baselines. We compare our DENC against three groups of methods for rating prediction: (1) Traditional methods, including NRT [113] and PMF [139]. (2) Social network-based methods, including GraphRec [46], DeepFM+ [70], SocialMF [93], SREE [116] and SoReg [132]. (3) Propensity-based methods, including CausE [14] and D-WMF [220].

Parameter Settings. We implement all baseline models on a Linux server with Tesla P100 PCI-E 16GB GPU. Datasets for all models except CausE [7] are split as training/test sets with a proportion of 80/20, and 20% of the training set are validation set. We optimize all models with Stochastic Gradient Descent(SGD) [16]. For fair comparisons, a grid search is conducted to choose the optimal parameter settings, e.g., dimension of user/item latent vector k_{MF} for matrix factorization-based models and dimension of embedding vector d for neural network-based models. The embedding size is initialized with the Xavier and searched in $[8, 16, 32, 64, 128, 256]$. The batch size and learning rate are searched in $[32, 64, 128, 512, 1024]$ and $[0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]$, respectively. The maximum epoch N_{epoch} is set as 2000, an early stopping strategy is performed. Moreover, we employ three hidden layers for the neural components of NRT, GraphRec and DeepFM+. Like our DENC method, DeepFM+ uses node2vec to train the social network embeddings. Hence, the embedding size of its node2vec is set as the same as in our DENC for a fair comparison. Without specification, unique hyperparameters of DENC are set as: three coefficients λ_a, λ_z and λ_d are tuned in $[0.2, 0.4, 0.6, 0.8, 1]$. The dimension of node2vec embedding size k_a and the dimension of inherent factor k_d are tuned in $[8, 16, 32, 64, 128, 256]$, and their influences are reported in Section 4.1.4.4.

4.1.4.2 Understanding Social Confounder (RQ1)

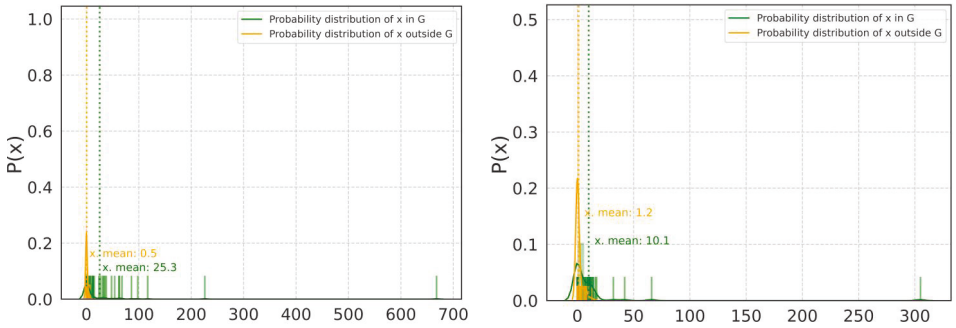
We initially conduct an experiment to understand to what extent the confounding bias caused by social networks is manifested in real-world recommendation datasets. We claim that social networks, as confounders, biases the interactions between users and items. We aim to verify two kinds of scenarios: (1) Users in the social network interact with more items than users outside the social network. (2) The pair of user-neighbors in the social network has more commonly interacted items than the pair of user-neighbors outside the social network. Intuitively, an unbiased platform should expect users to interact with items broadly, which indicates that interactions are likely to be evenly distributed. Thus, we investigate the social confounder bias by analyzing the statistics of interactions in these two scenarios in `Epinions` and `Ciao` datasets.

[7] As in CausE, we sample 10% of the training set to build an additional debiased dataset (mandatory in model training), where items are sampled to be uniformly exposed to users.



(a) Distribution on Ciao. (b) Distribution on Epinions.

Figure 4.5: Scenario (1): the distribution of x (the number of items interacted by a user). The smooth probability curves visualize how the number of items is distributed.



(a) Distribution on Ciao. (b) Distribution on Epinions.

Figure 4.6: Scenario (2): the distribution of x (the number of items commonly interacted by a user-pair).

For the first scenario, we construct two user sets within or outside the social network, i.e., $\mathcal{U}_G$ and $\mathcal{U}_{\setminus G}$. Specially, $\mathcal{U}_G$ is constructed by randomly sampling a set of users in social network G, and $\mathcal{U}_{\setminus G}$ is randomly sampled out of G. The size of $\mathcal{U}_G$ and $\mathcal{U}_{\setminus G}$ is the same and defined as n. Following the above guidelines, we sample $n = 7{,}000$ users for $\mathcal{U}_G$ and $\mathcal{U}_{\setminus G}$. Figure 4.5 depicts the distributions of the interacted items by users in $\mathcal{U}_G$ and $\mathcal{U}_{\setminus G}$. The smooth curves are continuous distribution estimates produced by the kernel density estimation. Apparently, the distribution for $\mathcal{U}_{\setminus G}$ is significantly skewed: most of the users interact with few items. For example, on Ciao, more than 90% of users interact with fewer than 40 items. By contrast, most users in the social network tend to interact with items more frequently. In general, the distribution curve of $\mathcal{U}_G$ is quite different from $\mathcal{U}_{\setminus G}$, which reflects that the social network influences the interactions between users and items. In addition, the degree of bias varies across different datasets: Epinions is less biased than Ciao.

Table 4.2: Performance comparison.

		Traditional		**Social network-based**					**Propensity-based**			
Dataset	Metrics	PMF	NRT	SocialMF	SoReg	SREE	GraphRec	DeepFM+	CausE	D-WMF	**DENC**	**improv.**
Epinions	MAE	0.9505	0.9294	0.8722	0.8851	0.8193	0.7309	0.5782	0.5321	0.3710	**0.2684**	38.2%
	RMSE	1.2169	1.1934	1.1655	1.1775	1.1247	0.9394	0.6728	0.7352	0.6299	**0.5826**	8.1%
Ciao	MAE	0.8868	0.8444	0.7614	0.7784	0.7286	0.6972	0.3641	0.4209	0.2808	**0.2487**	12.9%
	RMSE	1.1501	1.1495	1.0151	1.0167	0.9690	0.9021	0.5886	0.8850	0.5822	**0.5592**	4.1%
MovieLens-1M	MAE	0.8551	0.8959	0.8674	0.9255	0.8408	0.7727	0.5786	0.4683	0.3751	**0.2972**	26.2%
$\Delta(Z_u) = -0.35$	RMSE	1.0894	1.1603	1.1161	1.1916	1.0748	0.9582	0.6730	0.8920	0.6387	**0.5263**	21.4%
MovieLens-1M	MAE	0.8086	0.8801	0.8182	0.8599	0.7737	0.7539	0.5281	0.4221	0.3562	**0.2883**	23.4%
$\Delta(Z_u) = 0$	RMSE	1.0034	1.1518	1.0382	1.1005	0.9772	0.9454	0.6477	0.8333	0.6152	**0.5560**	10.6%
MovieLens-1M	MAE	0.7789	0.7771	0.7969	0.8428	0.7657	0.7423	0.3672	0.4042	0.3151	**0.2836**	11.1%
$\Delta(Z_u) = 0.35$	RMSE	0.9854	0.9779	1.0115	1.0792	0.9746	0.9344	0.5854	0.8173	0.5962	**0.5342**	9.6%

For the second scenario, based on $\mathcal{U}_G$ and $\mathcal{U}_{\setminus G}$, we further analyze the number of commonly interacted items by the user-pair. Particularly, we randomly sample four one-hop neighbours for each user in $\mathcal{U}_G$ to construct user-pairs. Since users in $\mathcal{U}_{\setminus G}$ have no neighbours, for each of them, we randomly select another four users[8] in $\mathcal{U}_{\setminus G}$ to construct four user-pairs. Recall that $\mathcal{U}_G$ and $\mathcal{U}_{\setminus G}$ both have 7,000 users, then we totally have $4 \times 7{,}000$ user-pairs for $\mathcal{U}_{\setminus G}$ and $\mathcal{U}_{\setminus G}$, respectively. Figure 4.6 represents the distribution of how many items are commonly interacted by the users in each pair.[9] Figure 4.6 indicates most user-neighbour pairs in the social network have fewer than 20 items in common. However the user-pairs outside the social network nearly have no items in common, i.e., less than 1. We can conclude that social networks can encourage users to share more items with their neighbours, compared with users who are not connected by any social networks.

4.1.4.3 Performance Comparison (RQ2)

We compare the rating prediction of DENC with nine recommendation baselines on three datasets including `Epinions`, `Ciao` and `MovieLens-1M`. Table 4.2 demonstrates the performance comparison, where the confounder $\Delta(Z_u)$ in `MovieLens-1M` is assigned with three different settings, i.e., -0.35, 0 and 0.35. The improvements and statistical significance test are performed between DENC and the strongest baselines (highlighted with underline). Analyzing Table 4.2, we have the following observations.

- Overall, our DENC consistently yields the best performance among all methods on five datasets. For instance, DENC improves over the best baseline model w.r.t.

[8]According to the statistics, we discover that 90% of users have at least four one-hop neighbours in `Ciao` and `Epinions`

[9]For example, $\{user1, user2, user3, user4\}$ are one-hop neighbours of $user5$. If the number of commonly items interacted by $user1$ and $user5$ is 3, then $x = 3$ in the x-axis of Figure 4.6 is nonzero.

MAE/RMSE by 38.2%/8.1%, 12.9%/4.1%, and 26.2%/21.4% on Epinions, Ciao and MovieLens-1M ($\Delta(Z_u)$=-0.35) datasets, respectively. We can conclude that the improvements of our DENC are statistically significant with all $p < 0.01$. These results indicate the effectiveness of DENC on the task of rating prediction, which has adopted a principled causal inference way to leverage both the inherent factors and auxiliary social network information for improving recommendation performance.

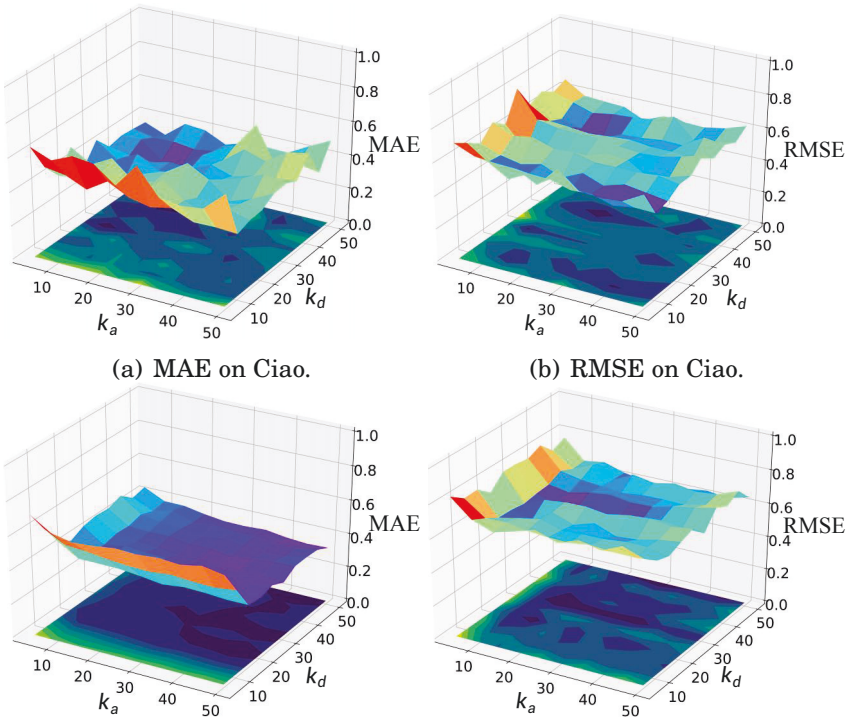
- Among the three kinds of baselines, propensity-based methods serves as the strongest baselines in most cases. This justifies the effectiveness of exploring the missing pattern in rating data by estimating the propensity score, which offers better guidelines to identify the unobserved confounder effect from ratings. However, propensity-based methods perform worse than our DENC, as they ignore the social network information. It is reasonable that exploiting the social network is useful to alleviate the confounder bias to rating outcome. The importance of social networks can be further verified by the fact that most of the social network-based methods consistently outperform PMF on all datasets.

- All baseline methods perform better on Ciao than on Epinions, because Epinions is significantly sparser than Ciao with 0.0140% and 0.0368% density of ratings. Besides this, DENC still achieves satisfying performance on Epinions and its performance is competitive with the counterparts on Ciao. This demonstrates that its exposure model of DENC has an outstanding capability of identifying the missing pattern in rating prediction, in which biased user-item pairs in Epinions can be captured and then alleviated. In addition, the performance of DENC on three Movielens-1M datasets is stable w.r.t. different levels of confounder bias, which verifies the robust debiasing capability of DENC.

4.1.4.4 Ablation Study (RQ3)

In this section, we conduct experiments to evaluate the parameter sensitivity of our DENC method. We have five important hyperparameters: k_a and k_d that correspond to the embedding size in loss function $\mathscr{L}_a$ and $\mathscr{L}_d$, respectively; λ_a, λ_z and λ_d that correspond to the trade-off parameters for $\mathscr{L}_a$, $\mathscr{L}_z$ and $\mathscr{L}_d$, respectively. Based on the hyperparameter setup, we vary the value of one hyperparameter while keeping the others unchanged.

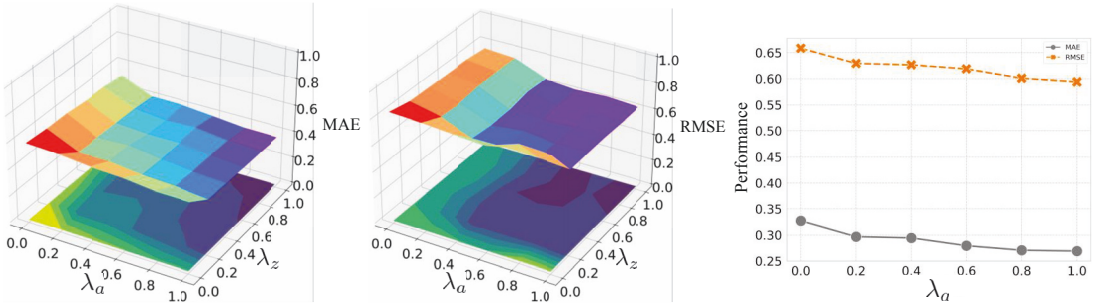
Figure 4.7 lays out the performance of DENC with different embedding sizes. For both datasets, the performance of our DENC is stable under different hyperparameters



(a) MAE on Ciao. (b) RMSE on Ciao.

(c) MAE on Epinions. (d) RMSE on Epinions.

Figure 4.7: Our DENC: Parameter sensitivity of k_a and k_d against (a) MAE (b) RMSE on `Ciao` and `Epinions` dataset.



(a) MAE on λ_a and λ_z. (b) RMSE on λ_a and λ_z. (c) Performance on λ_d.

Figure 4.8: Our DENC's sensitivity to λ_a, λ_z and λ_d on `Epinions` dataset.

k_a and k_d. The performance of DENC increases while the embedding size increase from approximately 0-15 for k_d; afterwards, its performance decreases. It is clear that when the embedding size is set to approximately k_a=45 and k_d=15, our DENC method achieves the optimal performance. Our DENC is less sensitive to the change of k_a than k_d, since MAE/RMSE values change with a obvious concave curve along k_d=0 to 50 in Figure 4.7, while MAE/RMSE values only change gently with a downward trend along k_a=0 to 50. It is reasonable since k_d controls the embedding size of disentangled user-item representation attained by the deconfounder model, i.e., the inherent factors, while social network embedding size k_a serves as the controller for auxiliary social information, the former can influence the essential user-item interaction while the latter affects the auxiliary information.

Sensitivity to Trade-off Parameter. As defined in objective function (4.15), the three most important trade-off parameters λ_a, λ_z and λ_d balance the contributions of exposure model loss, confounder loss and discrepancy loss, respectively. We evaluate our DENC's sensitivity to these three parameters on `Epinions` dataset. As shown in Figure 4.8, the values of trade-off parameters are chosen from $[0, 0.2, 0.4, 0.6, 0.8, 1]$. Figure 4.8 (a) and (b) present the performance of our model in terms of MAE and RMSE, which are generated by fixing the discrepancy loss weight λ_d and varying the trade-off between the other two parameters. Apparently, our performance is significantly improved compared with the model without λ_a and λ_z, i.e., the errors are reduced. Also, the overall performance on different combinations of hyperparameters of λ_a and λ_z is stable over a large parameter range, which confirms the effectiveness and robustness of debiasing in DENC approach. This conclusion is consistent with our model evaluation results. Figure 4.8 (c) indicates that adding the discrepancy loss to account for the exposure bias can improve the performance in terms of MAE and RMSE compared with only having the estimation of confounder and exposure assignment. This is the main reason why our method performs well when debiasing rating, but propensity-based method with logistic regression predicting the exposure assignment cannot accurately estimate rating.

Convergence Analysis. In Figure 4.9, we plot the convergence of objective loss (4.15) on the training set of `Epinions` and `Ciao`. One can see that the overall loss decreases as the epoch increases on both datasets. Note that the rates of convergences are different in different dataset. For example, the red curve starts to decrease significantly at epoch 10 and converges at epoch 40. While the green curve first converges a bit more slowly and then become stable at around epoch 40.

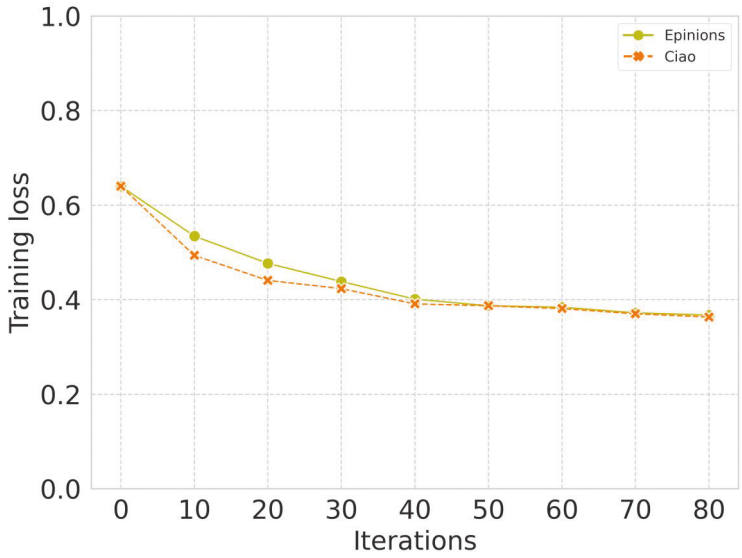
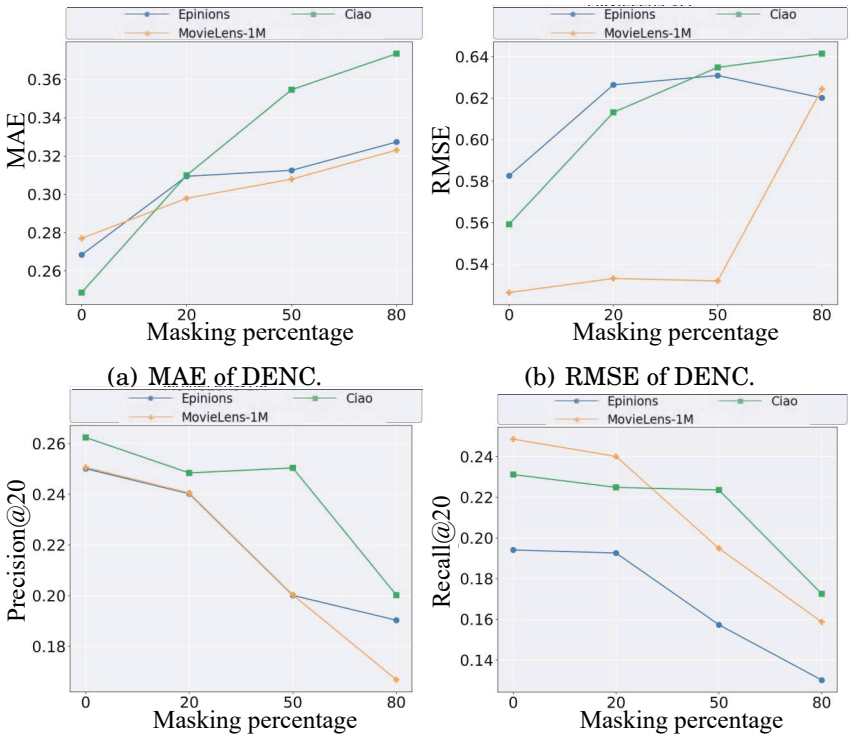


Figure 4.9: Our DENC's loss convergence curves on `Epinions` and `Ciao` datasets.

4.1.4.5 Case Study (RQ4)

We first investigate how the missing social relations affect the performance of DENC. We randomly mask a percentage of social relations to simulate the missing connections in social networks. For `Epinions`, `Ciao` and `MovieLens` dataset, we fix the social network confounder as $\Delta(Z_u) = 0$. Meanwhile, we exploit different percentages of missing social relations including {20%, 50%, 80%}. Note that we do not consider the missing percentage of 100%, i.e., the social network information is completely unobserved. Considering that the social network is viewed as a proxy variable of the confounder, the social network should provide partially known information. Following this guideline, we firstly investigate how the debias capability of our DENC method varies under the different missing percentages. Secondly, we also report the ranking performance of DENC (percentages of missing social relations is set to 0%) under Precision@K and Recall@K with $K = \{10, 15, 20, 25, 30, 35, 40\}$ to evaluate our model thoroughly.

Figure 4.10 illustrates our debias performance w.r.t. different missing percentages of social relations on three datasets. As shown in Figure 4.10, the missing social relations can obviously degrade the debias performance of DENC method. The performance evaluated by four metrics in Figure 4.10 consistently degrades when the missing percentage increases from 0% to 80%, which is consistent with the common observation. This indicates that the underlying social network can play a significant role in a recommendation,

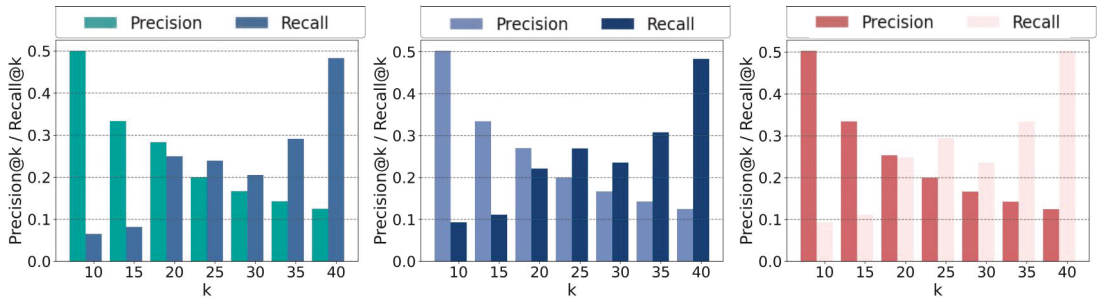


(a) MAE of DENC. (b) RMSE of DENC.

(c) Precision@20 of DENC. (d) Recall@20 of DENC.

Figure 4.10: Our DENC: debias performance w.r.t. different missing percentages of social relation.

because it can capture the preference correlations between users and their neighbours.



(a) `Epinions`. (b) `Ciao`. (c) `MovieLens-1M`($\Delta(Z_u) = 0$).

Figure 4.11: Performance of DENC in terms of Precision@K and RecallG@K under difference K

Based on the evaluation on Precision@K and Recall@K, Figure 4.11 shows that DENC achieves stable performance on Top-K recommendation when K (i.e., the length of ranking list) varies from 10 to 40. Our DENC can recommend more relevant items within top K positions when the ranking list length increases.

4.1.5 Summary

In this research, we have investigated the missing not at random problem in the recommendation and addressed the confounding bias from a causal perspective. Instead of merely relying on inherent information to account for selection bias, we developed a novel social network embedding-based de-bias recommender for unbiased rating, through correcting the confounder effect arising from social networks. We evaluate our DENC method on two real-world and one semi-synthetic recommendation datasets, with extensive experiments demonstrating the superiority of DENC in comparison to state-of-the-art models.

4.2 Popularity Bias Mitigation with Disentangle Learning

4.2.1 Overview

Research Question. Early recommendation models mainly adopt collaborative filtering (CF) methods [108] to model user preferences based on historical user-item interactions (e.g., ratings, clicks). However, such user-item interaction usually exhibits bias that is entangled with users' real interests, ultimately degrading the recommendation performance. For instance, in movie recommendations, users are more likely to watch movies that are watched by many people, which is, however, due to users' conformity to other people rather than stemming from users' real interests [203, 262]. Therefore, it is essential to capture users' pure interests independent of the bias to build high-quality recommender models. To this end, this research aims to disentangle users' true interests (i.e., user intents) under the data in order to mitigate the potential bias issues.

Research Objective. By analyzing existing works on disentangle learning in Section 3.5.1.2, we can find there are two limitations in these works: Firstly, most of them merely focus on user-item interactions, which, however, suffer sparse issues [80] that lead to the difficulty of learning effective user/item representations. Moreover, existing disentangled learning methods merely treat one user-item interaction record as an independent instance and neglect its rich context information. We claim that the rich context information in the form of heterogeneous information can help to disentangle and interpret semantic-aware intents of users for robust recommendations. For instance, higher-order path structure like a meta path *User-Movie-Actor-Movie-User* encodes the

semantics interpretation of "movies starring the same actor are highly likely to be rated by users". Without considering rich context information, disentangled learning fails to offer fine-grained interpretability of item attributes (e.g., *actor*) for recommendation. Therefore, the research objective of this work is to enhance user intent disentangle learning through the augmentation of heterogeneous information.

Motivations. Enhancing user intent disentangle learning with heterogeneous information is not trivial. The heterogeneous information is complicated and consists of various types of data (e.g., user/item aspects), which is usually grouped by its attributes (e.g., *gender*, *actor*). These aspect groups are frequently presented with skewed distributions that cause popularity bias in modeling the user preference and prediction score. The skewed distribution is attributed to missing values of aspects, i.e., the number of non-missing aspects is not evenly distributed in the observational dataset. An empirical study conducted on `Douban Movie` dataset can validate this claim by Figure 4.12: unobserved *Director* aspect accounts for 19.7% of items compared to *Actor* accounting for 7.6%. That is, the skewed distribution of aspects can easily bias the prediction model towards the majority group, even though their items have the same matching level (see example in Figure 4.12). Thus, the motivation of this work is to use the power of heterogeneous information in user intent disentangle learning, at the same time, alleviating the popularity bias bought by these heterogeneous information.

Figure 4.12: An example of bias: movie `Harry Potter` (`HP`) contains only one aspect *Director*, however, *Actor* and *Type* are missing. The high rating of the user on movie `Harry Potter` trains a prediction model towards the user's preference on the director *Steve Kloves*. In contrast, we observed that movie `Racing with the Moon` (`RWM`) with the same director received very low ratings from the same user.

The Proposed Approach. We propose to tackle the challenge from a novel causal perspective, to develop an unbiased and interpretable disentangled approach on heterogeneous information, namely, CaDSI (Causal Disentanglement for Semantics-Aware Intent Learning). To make users' intents semantic-aware, we propose a pre-trained model as the first stage to leverage multiple item facets of heterogeneous information. As a second stage, besides considering items directly interacted by the user, the higher-order interacted items via meta paths are exploited to disentangle user intents in a robust manner. Finally, the pre-trained model, together with the disentangled learning, is subsequently fine-tuned by the causal intervention. Thanks to the development of causal inference, the final module is designed to adopt a causal intervention mechanism to eliminate the bias introduced by heterogeneous information. With these stages, our method can guide the unbiased and semantic-aware representations disentangling user intents for the robust recommendation.

Contributions. The key contributions of this work are fourfold:

- Fundamentally different from previous works, CaDSI is the first method that can disentangle the unbiased user intents from a causal perspective, in the meanwhile endow each user intent with specific semantics under the disentanglement learning task.
- We design a novel causal graph for the qualitative analysis of causal relationships in recommendations, based on which a pre-trained model is developed. With heterogeneous information, the pre-trained model can semantically account for the item aspects influence towards the user intent.
- To eliminate popularity bias stemming from heterogeneous information, we perform the causal intervention on user representations and refine the pre-trained model for unbiased user intent learning.
- We conduct extensive experiments to show that our CaDSI method outperforms state-of-the-art methods. The interpretability of our CaDSI is also validated by our empirical study.

4.2.2 Causal Graph for Popularity Bias

We consider the causal graph [154], as introduced in Definition 3.1.2, to reveal the true causal relations in recommendations. The basic idea of our proposed approach is to disentangle and interpret users' intent based on heterogeneous information. From a

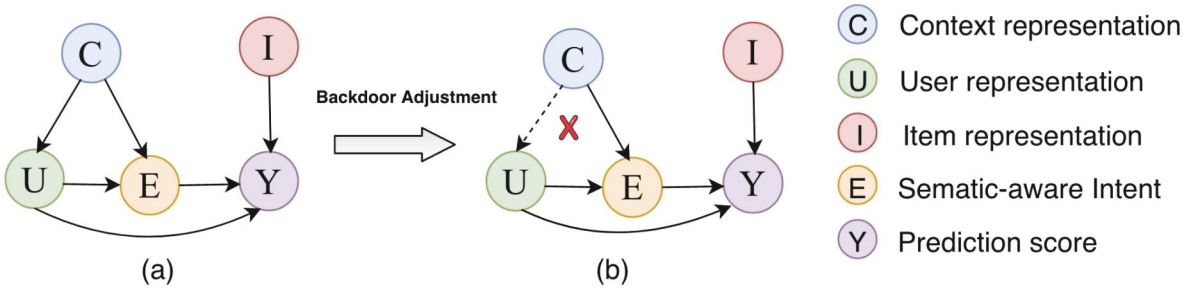


Figure 4.13: Causal disentanglement model for recommendation. We apply a backdoor adjustment to remove the effect of confounder C for U, as indicated by the red cross.

causal perspective, Figure 4.13 (a) demonstrates the illustrative causal graph that offers an interpretable representation for a disentangled recommender system, which consists of four variables including $\{U, C, E, Y\}$.

- C as a confounder is the item aspects (e.g., *actor*) acquired from HINs. The representation of C can be learned by a pre-trained model from an HIN, which retains semantic information on item aspects.
- U denotes the user representation which essentially reveals k user intents. U is presented in the form of k chunked intent representation, where each chunk reveals a piece of user intent, such as the user's preference towards the items' brand.
- I is the item representation and each I denotes the embedding of one item attribute (e.g. *actor*).
- E is the semantic-aware intent representation generated by the context information from C and the user representation U. E retains the information about the user's intent towards different item aspects.
- $Y \in [0, 1]$ is the recommendation probability of the user-item pair.

The directed edge represents the causal relation between two variables; in particular, the rationality of causal relations can be explained as follows.

- $C \rightarrow U$: The prior knowledge C of item aspects affects user representation U, which is reflected by the fact that users prefer items that have particular attributes (e.g., *actor*).

- $(C, U) \rightarrow E$: Item context C and user representation U consist of the semantic-aware user intent representation.
- $I \rightarrow Y$: Item representation by I affects the recommendation probability Y.
- $U \rightarrow Y$: User's preference represented by U affects the recommendation probability Y.
- $U \rightarrow E \rightarrow Y$: The recommendation probability of the item could be high if the user shows interest in the context of the item, e.g., the item type rather than the item. For example, items whose type is "Lipstick" are more likely to be purchased by the user u whose gender is female.

4.2.2.1 Adjusting Confounding Bias via Intervention

From this causal graph, the semantic knowledge C is a confounder between user representation U and recommendation outcome Y, since C is the common cause of U and Y by definitions in causal theory. The presence of confounder C leads to the spurious correlation between U and Y if we ignore to account its causal effect into modeling, which is equal to the estimation of $P(Y \mid U)$. In semantic knowledge-aware recommendation, the confounder C (i.e., semantic knowledge) makes $P(Y \mid U)$ biased towards items that have dominant item aspects. For example, as illustrated in Figure 4.12, we expect that the rating prediction of `RWM` is caused by both of the three item attributes, but not only the dominant *director*, which has a majority attribute popularity (i.e., the majority group). In the language of causal inference, the correlation $P(Y \mid U)$ fails to capture the true causality between U and Y since the prediction of Y conditions not only U, but also the spurious correlations of (1) $C \rightarrow U \rightarrow Y$, i.e., prior knowledge C determines the prediction likelihood through user representation U. For example, undesirable and low-quality items in specific aspect groups will not attract users' intent. (2) $C \rightarrow E \rightarrow Y$. i.e., the semantic-ware user intent representation E derived from C affects the prediction Y. Once users' future interest in item aspect groups changes (i.e., user interest drift), the recommendations will be biased.

To pursue the true causality between U and Y, we should propose a causal intervention method $P(Y \mid do(U))$ to remove the confounding bias from C. The $do(\cdot)$ operation [154] is to forcibly and externally assign a certain value to the variable U, which can be intuitively seen as removing the edge $C \rightarrow U$ and blocking the effect of C on U (as shown in Figure 4.13 (b)). As a result, the prediction likelihood can be independent of its causes so as to generate robust recommendations that are free from confounding bias.

Table 4.3: Key notations and descriptions.

Notation	Description
$\mathscr{G}$	Heterogeneous Information Network (HIN)
$\mathscr{V}$	node set of HIN
$\mathscr{A}$	node type set of HIN
$\mathscr{P}$	meta paths set of HIN
$\mathbf{p}$	a meta path in $\mathscr{P}$
$\boldsymbol{y} \in \mathbb{R}^{m \times n}$	user item interaction matrix
$\hat{\boldsymbol{y}}_{ui}$	predicted interaction likelihood of user u on item i
$\boldsymbol{c_u}$	semantic-aware embedding for user u
$\boldsymbol{c_i}$	semantic-aware embedding for item i
$\boldsymbol{c_a}$	context embedding for aspect a
k	user intent number
$\boldsymbol{l}$	iteration number of graph disentangling module
L	L-th layer of graph disentangling module
$\mathbf{S}_k(u,i)$	intent score of u and i on intent k
$\boldsymbol{u}^u$	intent-aware embedding for u
$\boldsymbol{i}^i$	intent-aware embedding for item i

4.2.3 Methodology

4.2.3.1 Notions and Problem Definition

We formulate our task as disentangling and interpreting users' intent based on the Heterogeneous Information Network (HIN). The important concepts of HIN and the formal definition of our problem are given as follows.

Definition 4.2.1 (Heterogeneous Information Network). A Heterogeneous Information Network (HIN) is denoted as $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with a node type mapping function: $\phi : \mathscr{V} \rightarrow \mathscr{A}$ and an edge type mapping function: $\psi : \mathscr{E} \rightarrow \mathscr{R}$, where $\mathscr{A}$ and $\mathscr{R}$ are the node type set and edge type set of $\mathscr{G}$, respectively. Each node $v \in \mathscr{V}$ and edge $e \in \mathscr{E}$ in a HIN belongs to one particular type with $\phi(v) \in \mathscr{A}$ and $\psi(e) \in \mathscr{R}$, where $|\mathscr{A}| + |\mathscr{R}| > 2$ (i.e., heterogeneity).

Definition 4.2.2 (Meta Path). Meta path $\mathbf{p}$ is a path defined on the network schema $T_{\mathscr{G}} = (\mathscr{A}, \mathscr{R})$, and is denoted in the form of

$$\mathbf{p} \triangleq (\mathscr{A}_1 \overset{\mathscr{R}_1}{\rightarrow} \mathscr{A}_2 \overset{\mathscr{R}_2}{\rightarrow} \ldots \overset{\mathscr{R}_l}{\rightarrow} \mathscr{A}_{o+1})$$

which defines a composite $\mathscr{R} = \mathscr{R}_1 \mathscr{R}_2 \ldots \mathscr{R}_o$ between type $\mathscr{A}_1$ and $\mathscr{A}_{o+1}$. For simplicity, we use node type names to denote the meta path if no multiple relations exist between type

pairs, as $\mathbf{p} = (\mathscr{A}_1\mathscr{A}_2...\mathscr{A}_{o+1})$. Commonly, a HIN contains multiple meta paths, the meta path set is defined as $\mathscr{P}$ where each $\mathbf{p} \in \mathscr{P}$.

Based on these important concepts, we collect the key notations in Table 4.3 and formulate the problem to be solved as follows.

Definition 4.2.3 (Problem Definition). Given user and item sets, we define an interaction matrix $\boldsymbol{y} \in \mathbb{R}^{m \times n}$ where entry $\boldsymbol{y}_{ui} = 1$ indicates a user u in user set interacts with an item i in item set, otherwise $\boldsymbol{y}_{ui} = 0$. We also have additional contextual information about users and items, e.g., social relationships between users or item brands and categories, absorbing in $\mathscr{G}$ of Definition 4.2.1. Thus, we aim to learn the prediction function P parameterized by Θ, such that $\hat{\boldsymbol{y}}_{ui} = P(u, i|\boldsymbol{y}, \mathscr{G}; \Theta)$, where $\hat{\boldsymbol{y}}_{ui}$ denotes the probability that user u will engage with item i conditional on the given $\boldsymbol{y}$ and $\mathscr{G}$.

We now introduce our proposed model, which includes *Causal Disentanglement Model* and *Causal Intervention* as shown in Figure 4.14.

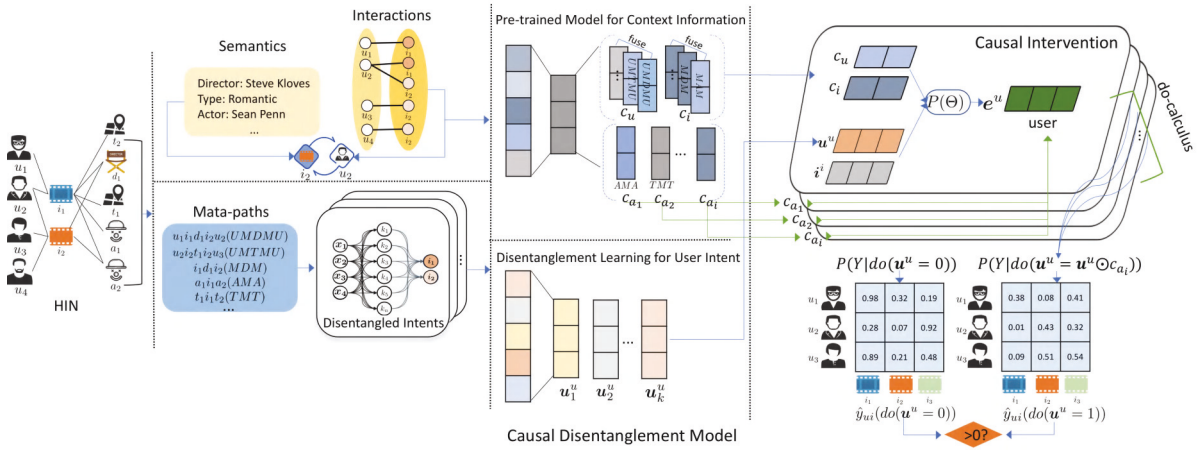


Figure 4.14: Overview of the proposed CaDSI. CaDSI takes HIN as the input, and passes the causal disentanglement model for learning semantic-aware intent representations (*cf.* Section 4.2.3.4); then uses the causal intervention (*cf.* Section 4.2.3.5) for controlling the confounding bias.

4.2.3.2 Pre-trained Model for Learning Context Information

The pre-trained model for learning context representation C is a key component in the causal disentanglement model, which aims to leverage a given HIN to construct semantic-aware representations for users, items and aspects directly. Specifically, given a

HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and its corresponding meta paths set $\mathcal{P}$, we aim to learn semantic-aware representations (a.k.a., embedding) $\boldsymbol{c_u}$ for each user node $u \in \mathcal{V}$, $\boldsymbol{c_i}$ for each item node $i \in \mathcal{V}$ and $\boldsymbol{c_a}$ for a specific type of aspect a (e.g., *Actor*).

A *Heterogeneous Skip-Gram with Meta Path Based Random Walks* is designed to output a set of multinomial distributions, while each distribution corresponds to one type of node (i.e., *User*, *Item* and *Aspect* type); The *Meta Path Based Random Walks* is used to generate node sequences that capture complex semantics under a given HIN, while *Heterogeneous Skip-Gram* takes the generated node sequences as inputs and catches heterogeneous neighbors of a node to output the semantic-aware representations. Finally, the semantic-aware representations for users, items and aspects are given by aggregating every node representation under different meta paths by an *Embedding Fusion*.

Meta Path Based Random Walks. The *Meta Path Based Random Walks* [41] generates node sequences that can capture both the semantic and structural correlations between different types of nodes. The basic idea is to put random walkers [66] in a HIN to generate paths that constitute multiple types of nodes. Specifically, given $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \psi)$, the node sequence $\mathbf{n_p} = \{v_1, \cdots, v_{i+1}\}$ under a specific meta path $\mathbf{p} = (\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_{o+1})$ is generated according to the following distribution:

$$P(v_{i+1} \mid v_i, \mathbf{p}) = \begin{cases} \frac{1}{\left|\mathcal{N}_{v_i}^{(\mathcal{A}_{o+1})}\right|}, & (v_i, v_{i+1}) \in \mathcal{E} \text{ and } \phi(v_{i+1}) = \mathcal{A}_{o+1} \\ 0, & \text{otherwise} \end{cases} \tag{4.16}$$

where $\mathcal{N}_{v_i}^{(\mathcal{A}_{o+1})}$ is the first-order neighbor set for node v_i whose type is $\mathcal{A}_{o+1}$. The v_{i+1} is the $i+1$-th node whose type is $\mathcal{A}_{o+1}$, and v_i is the i-th node in the walk which belongs to type $\mathcal{A}_o$. By regulating $v_i \in \mathcal{A}_o$ while $v_{i+1} \in \mathcal{A}_{o+1}$, the node types sampled by random walkers are conditioned on the pre-defined meta path $\mathbf{p}$.

Following the pre-defined meta paths in Table 4.5, by performing the *Meta Path Based Random Walks* on each meta path $\mathbf{p} \in \mathcal{P}$, we can obtain node sequences set for all meta paths as $\mathbf{n}^{\mathcal{P}} = \{\mathbf{n}_1, \cdots, \mathbf{n}_{|\mathcal{P}|}\}$. As we care about semantic-aware embeddings for users, items and aspects, we select meta paths starting with *user*, *item* or *Aspect* type and reorganize its corresponding node sequences into *user* type-specific set $\mathbf{n}(U)$ as $\mathbf{n}(U) = \{\mathbf{n}_1, \cdots, \mathbf{n}_m\}$, *item* type-specific set $\mathbf{n}(I)$ as $\mathbf{n}(I) = \{\mathbf{n}_1, \cdots, \mathbf{n}_n\}$ and aspect type-specific set $\mathbf{n}(A) = \{\mathbf{n}_1, \cdots, \mathbf{n}_h\}$. We then use *Heterogeneous Skip-Gram* to generate semantic-aware embeddings of node sequences in $\mathbf{n}(U)$, $\mathbf{n}(I)$ and $\mathbf{n}(A)$.

Heterogeneous Skip-Gram. Given $\mathbf{n}(U)$, $\mathbf{n}(I)$ and $\mathbf{n}(A)$ generated from Eq. (4.16), we aim to leverage *Heterogeneous Skip-Gram* [41] to learn semantic-aware user rep-

resentation $\boldsymbol{c}_u$ for sequence $\mathbf{n}_u$ in $\mathbf{n}(U)$. Analogously, $\boldsymbol{c}_i$ is the semantic-aware item representation for sequence $\mathbf{n}_i$ in $\mathbf{n}(I)$ while aspect representation $\boldsymbol{c}_a$ is learned for sequence $\mathbf{n}_a$ in $\mathbf{n}(A)$. Specifically, a *Heterogeneous Skip-Gram* is designed to learn node representations by aggregating node neighbors in node sequences [41]. Given each user node sequence $\mathbf{n}_u$ in $\mathbf{n}(U)$, the *Heterogeneous Skip-Gram* model learns the semantic-aware embedding $\boldsymbol{c}_u$ of $\mathbf{n}_u$ by maximizing the probability of having the heterogeneous neighbors $\mathcal{N}_u$ given a node u:

$$\mathcal{L}_\theta = \sum_{u \in \mathcal{V}} \sum_{u_c \in \mathcal{N}_u^{\mathcal{A}_i}} \sum_{\mathcal{A}_i \in \mathcal{A}} \left(\sigma\left({\boldsymbol{c}_u}^T \boldsymbol{c}_{u_c} \right) \prod_{w=1}^{W} \sigma\left({\boldsymbol{c}_u}^T \boldsymbol{c}_w \right); \theta \right) \tag{4.17}$$

where $\mathcal{N}_u^{\mathcal{A}_i}$ denotes u's neighbors whose type is $\mathcal{A}_i$ and u_c is one node in the neighbors set $\mathcal{N}_u$. The $\boldsymbol{c}_u$ and $\boldsymbol{c}_{u_c}$ are latent vectors that correspond to the target node and context node representations of u and u_c, and $\sigma(x) = 1/1 + \exp(-x)$. The W is a parameter that determines the number of negative examples to be drawn per a positive example. The $\boldsymbol{c}_w$ is the sampled node's representation within W negative samples and θ is the model parameters of *Heterogeneous Skip-Gram*. Finally, $\boldsymbol{c}_u$ for each node sequence $\mathbf{n}_u$ in $\mathbf{n}(U)$ are estimated by applying gradient descent algorithms. Analogously, we can build semantic-aware item representation $\boldsymbol{c}_i$ and the aspect representation $\boldsymbol{c}_a$.

Embedding Fusion. Since we have multiple node sequences in $\mathbf{n}(U)$, $\mathbf{n}(I)$ and $\mathbf{n}(A)$, we should perform embedding fusion to aggregate every representations $\boldsymbol{c}_u$, $\boldsymbol{c}_i$ and $\boldsymbol{c}_a$ of sequences in $\mathbf{n}(U)$, $\mathbf{n}(I)$ and $\mathbf{n}(A)$ into an uniform manner. The reason why we fuse the individual embedding of each node sequence is straightforward. Firstly, in a recommender system, the optimization goal is to learn effective representations for users and items. Hence, it requires a principled fusion way to transform node embeddings w.r.t. different meta paths relating *user* type or *item* type into a suitable form for later recommendation tasks. Secondly, the context information across meta paths starting with an aspect type should be further arranged into an uniform embedding space, representing one piece of semantic factors. The embedding fusion is implemented as a liner combination function defined as follows:

$$
\begin{aligned}
\boldsymbol{c_u} &\leftarrow \frac{1}{|\boldsymbol{c}_u(U)|} \sum_{j=1}^{|\boldsymbol{c}_u(U)|} \left(\mathbf{M} \cdot \boldsymbol{c}_u^j + b\right) \\
\boldsymbol{c_i} &\leftarrow \frac{1}{|\boldsymbol{c}_i(I)|} \sum_{j=1}^{|\boldsymbol{c}_i(I)|} \left(\mathbf{M} \cdot \boldsymbol{c}_i^j + b\right) \\
\boldsymbol{c_a} &\leftarrow \frac{1}{|\boldsymbol{c}_a(A)|} \sum_{j=1}^{|\boldsymbol{c}_a(A)|} \left(\mathbf{M} \cdot \boldsymbol{c}_a^j + b\right)
\end{aligned}
\tag{4.18}
$$

where $\boldsymbol{c}_u(U)$ is the user node representation set that absorbs the node representations of user u and $\boldsymbol{c}_u(U) = \{\boldsymbol{c}_u^1, \cdots, \boldsymbol{c}_u^m\}$. Correspondingly, the item and aspect node representation sets $\boldsymbol{c}_i(I) = \{\boldsymbol{c}_i^1, \cdots, \boldsymbol{c}_i^n\}$ and $\boldsymbol{c}_a(A) = \{\boldsymbol{c}_a^1, \cdots, \boldsymbol{c}_a^h\}$ are established for item i and aspect a. $\mathbf{M}$ is a linear combination transformation matrix [226] and $\boldsymbol{b}$ is the error term. Through Eq. (4.18), $\boldsymbol{c_u}$, $\boldsymbol{c_i}$ and $\boldsymbol{c_a}$ can be learned as final semantic-aware representations for a user u and an item i and context representation for aspect a, respectively.

4.2.3.3 Disentanglement Learning for User Intent

Inspired by recent achievements on GNNs [10, 80, 205], we propose a GNN-based disentangling module to learn user representations that can essentially reveal k user intents. Specifically, the L-layer disentangling module exploits the high-order connectivities among the user-item interaction graph and initializes intent-ware embeddings by separating each user/item embedding into k chunks. Then, an interactive update rule that computes the importance scores of intent-aware user-item interactions is designed to refine intent-aware embeddings, so as to disentangle the holistic interaction graph into k intent-ware sub-graphs. Thereafter, each intent-aware embedding chunk are stacked by embedding propagation in the current layer, serving as the holistic intent-aware embedding $\boldsymbol{u}^u$ and $\boldsymbol{i}^i$ for user u and item i, where each intent-aware embedding $\boldsymbol{u}^u$ and $\boldsymbol{i}^i$ is composed of k independent chunks:

$$
\boldsymbol{u}^u = [\boldsymbol{u}_1^u, \cdots, \boldsymbol{u}_k^u], \quad \boldsymbol{i}^i = [\boldsymbol{i}_1^i, \cdots, \boldsymbol{i}_k^i]
\tag{4.19}
$$

Each chunked representation $\boldsymbol{u}_k^u \in \mathbb{R}^{\frac{d}{k}}$ and $\boldsymbol{i}_k^i \in \mathbb{R}^{\frac{d}{k}}$is built upon the intent-aware interactions between user u and its preferred items under intent i. We ultimately sum up intent-aware representations at each intent k of all L layers, the final layer outputs the k-th chunked intent-aware representations $\boldsymbol{u}_k^u$ and $\boldsymbol{i}_k^i$:

$$\boldsymbol{u}_k^u = \boldsymbol{u}_k^u(1) + \cdots + \boldsymbol{u}_k^u(L), \quad \boldsymbol{i}_k^i = \boldsymbol{i}_k^i(1) + \cdots + \boldsymbol{i}_k^i(L) \tag{4.20}$$

The detailed operations are given as follows.

Initialization. As ID embedding captures the intrinsic characteristics of users, we separate the ID embedding $\boldsymbol{x}$ of user u into k chunks and associate each chunk with a latent intent. Then the $\boldsymbol{x}$ serves as the initialization of $\boldsymbol{u}^u$ in Eq. (4.19):

$$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k] \tag{4.21}$$

Thereafter, we initialize the importance score $\mathbf{S}_k(u,i)$. The $\mathbf{S}_k(u,i)$ is the importance score of the interaction between u and i concerning the k-th intent. Such $\mathbf{S}_k(u,i)$ can be seen as the indicator of whether u should interact with i under a specific intent k. Thus, by learn $\mathbf{S}_k(u,i)$ for aspect $k \in \{1, \cdots, k\}$, we can construct an intent-aware sub-graph for intent k. Such that the embedding propagation can output intent-aware representations at intent k for all users based on the corresponding intent-aware sub-graph. We uniformly initialize importance score $\mathbf{S}_k(u,i)$ as $\mathbf{S}_k(u,i) = \frac{1}{k}$ which presumes the equal contributions of intents at the start of modeling.

Iterative Update Rule. An iterative update rule is then designed to update the importance score $\mathbf{S}_k(u,i)$ within $\boldsymbol{l}$ iterations, so as to disentangle intent sub-graphs to refine each intent-aware embedding chunk $\boldsymbol{x}_k$ in Eq. (4.21). Note that $\boldsymbol{x}_k \in \boldsymbol{x}$ in Eq. (4.21) serves as the initialized representation chunk of $\boldsymbol{u}_k^u \in \boldsymbol{u}^u$ in Eq. (4.19) and is used to memorize the updated value during iteration. The final $\boldsymbol{x}_k$ is assigned to each $\boldsymbol{u}_k^u$ as the final intent-aware representation chunk of user u.

In particular, we set $\boldsymbol{l}$ iterations in the interactive update. At each iteration, for the target interaction (u,i), we firstly normalize the score vector $\mathbf{S}_k(u,i) \mid \forall k \in \{1, \cdots, k\}\}$ over all intents into $\tilde{\mathbf{S}}_k$ via a softmax function:

$$\tilde{\mathbf{S}}_k^{\boldsymbol{l}}(u,i) = \frac{\exp\left(\mathbf{S}_k^{\boldsymbol{l}}(u,i)\right)}{\sum_{k'=1}^{k} \exp\left(\mathbf{S}_{k'}^{\boldsymbol{l}}(u,i)\right)} \tag{4.22}$$

which is capable of illustrating which intent should get more attention in the user-item interaction (u,i).

We then learn the representation $\boldsymbol{x}_k$ of the intent-aware graph whose adjacency matrix is $\tilde{\mathbf{S}}_k$ via embedding propagation. The weighted sum aggregator of the embedding propagation is defined as:

$$\boldsymbol{x}_k^{\boldsymbol{l}} = \sum_{i \in \mathcal{N}_u} \frac{\tilde{\mathbf{S}}_k^{\boldsymbol{l}}(u,i)}{\sqrt{D_k^{\boldsymbol{l}}(u) \cdot D_k^{\boldsymbol{l}}(i)}} \cdot \boldsymbol{i}_k^i \tag{4.23}$$

where $\boldsymbol{x}_k^{\boldsymbol{l}}$ is $\boldsymbol{u}_k^u$'s temporary representation that memorizes information acquired from $\mathcal{N}_u$ at the $\boldsymbol{l}$-th iteration. The $D_k^{\boldsymbol{l}}(u) = \sum_{i' \in \mathcal{N}_u} \tilde{\mathbf{S}}_k^{\boldsymbol{l}}\left(u, i'\right)$ and $D_k^{\boldsymbol{l}}(i) = \sum_{u' \in \mathcal{N}_i} \tilde{\mathbf{S}}_k^{\boldsymbol{l}}\left(u', i\right)$ are the degrees of user u and item i, respectively.

Then we interactively update intent-aware graphs. Intuitively, historical items of users driven by the same intent tend to have similar chunked representations. This goal can be achieved by encouraging users and items among the same intent sub-graph to have stronger relationships. We hence iteratively update the interaction importance score $\mathbf{S}_k^{\boldsymbol{l}}(u,i)$ to strengthen the degree between the centroid u and its neighbor i under intent k, as follows:

$$\mathbf{S}_k^{\boldsymbol{l}+1}(u,i) = \mathbf{S}_k^{\boldsymbol{l}}(u,i) + \boldsymbol{x}_k^{\boldsymbol{l}\top} \tanh\left(\boldsymbol{i}_k^i\right) \tag{4.24}$$

where $\boldsymbol{x}_k^{\boldsymbol{l}\top} \tanh\left(\boldsymbol{i}_k^i\right)$ considers the affinity between $\boldsymbol{x}_k^{\boldsymbol{l}}$ and $\boldsymbol{i}_k^i$, and $tanh$ is a nonlinear activation function that increases the representation ability of model.

After $\boldsymbol{l}$ iterations, $\boldsymbol{u}^u(1) = \boldsymbol{x}^{\boldsymbol{l}}(1)$ for user u is obtained in the first layer, where each chuncked representation $\boldsymbol{u}_k^u(1) = \boldsymbol{x}_k^{\boldsymbol{l}}(1)$ denotes the chunked embedding of $\boldsymbol{u}^u(1)$ on the intent k corresponds to the k-th dimension of Eq. (4.19).

Layer Combination. To explore high-order connectivities between users and items, we recursively formulate the representation of L-th layer as:

$$\boldsymbol{u}_k^u(L) = g\left(\boldsymbol{x}_k^{\boldsymbol{l}}(L-1), \left\{\boldsymbol{i}_k^i(L-1) \mid i \in \mathcal{N}_u\right\}\right) \tag{4.25}$$

where $\boldsymbol{u}_k^u(L)$ and $\boldsymbol{i}_k^i(L)$ are the representations of user u and item i on the k-th intent at layer L. The $\boldsymbol{x}_k^{\boldsymbol{l}}(L-1)$ is the chunked embedding of k-th intent at $L-1$-th layer of $\boldsymbol{u}_k^u(L-1)$. Note that $g(\cdot)$ is a fully connection layer memorizes the information propagated from the $(L-1)$-order neighbors of u.

Finally, after L layers, we sum up intent-aware representations at different layers as the final representation of the k-th chunk of Eq. (4.19), as $\boldsymbol{u}_k^u = \boldsymbol{u}_k^u(1) + \cdots + \boldsymbol{u}_k^u(L)$, where $\boldsymbol{u}_k^u$ donates the intent-aware representation for user u at intent k. Analogously, we can establish the final intent-aware embedding chunk $\boldsymbol{i}_k^i$ following the definition in Eq. (4.20).

4.2.3.4 Semantic-aware Intent Learning

Having obtained the intent-aware embeddings $\boldsymbol{u}^u$, $\boldsymbol{i}^i$ from *Disentanglement learning for User Intent*, our method takes semantics factors $\boldsymbol{c_u}$ and $\boldsymbol{c_i}$ from *Pre-trained Model for Context Information* as the auxiliary inputs for the semantic-aware recommendation. Specifically, we design an operator to instantiate a semantic-aware intent representation $\boldsymbol{e}$ that incorporates intent-aware embeddings and semantics factors. The intent representation $\boldsymbol{u}^u$ is updating through training together with $\boldsymbol{e}$ to capture the effect of semantics factors. The learned $\boldsymbol{u}^u$ retains the information from semantic factors and can be easily plugged into the backdoor adjustment to alleviate bias. The second-order Factorization Machine (FM) [160] operator for instantiating $\boldsymbol{e}$ is defined as:

$$\boldsymbol{e} = \sum_{a=1}^{d} \sum_{b=1}^{d} \boldsymbol{u}_a^u \boldsymbol{c}_{\boldsymbol{i}_a} \odot \boldsymbol{c}_{\boldsymbol{u}_b} \boldsymbol{i}_b^i \tag{4.26}$$

where $\odot$ denotes the element-wise product that is used for capturing interactions between intent representations and semantics factors.

Next, the semantic-aware intent representation $\boldsymbol{e}$ can be incorporated into recommender models as one additional user representation. Formally, we use the collaborative filtering to calculate the prediction score $\hat{\boldsymbol{y}}_{ui}$ given user and item ID representations, as follows:

$$\hat{\boldsymbol{y}}_{ui} = f(\boldsymbol{u}, \boldsymbol{i}, \boldsymbol{e}) = \delta \boldsymbol{u}^\top \boldsymbol{i} + (1-\delta) \boldsymbol{e}^\top \boldsymbol{i} \tag{4.27}$$

where $\boldsymbol{u}$ and $\boldsymbol{i}$ are the ID embeddings given by Multi-OneHot [254], and δ is the coefficient that describes how much each component contributes to the prediction score.

Then we use the pairwise BPR loss [162] to optimize the model parameters Θ. Specifically, BPR loss encourages the prediction likelihood of a user's historical items to be higher than those of unobserved items:

$$\mathscr{L}_{\text{BPR}} = \sum_{(u,i,j) \in O} -\ln \sigma \left(\hat{\boldsymbol{y}}_{ui} - \hat{\boldsymbol{y}}_{uj} \right) + \lambda \|\Theta\|_2^2 \tag{4.28}$$

where $\boldsymbol{u}$, $\boldsymbol{i}$ and $\boldsymbol{j}$ are the ID embeddings of user u, item i and item j, O denotes the training dataset involving the observed interactions O^+ and unobserved counterparts O^- and $O = \left\{ (u,i,j) \mid (u,i) \in O^+, (u,j) \in O^- \right\}$. The $\sigma(\cdot)$ is the sigmoid function, and λ is the coefficient controlling regularization.

4.2.3.5 Causal Intervention for Debiasing

Contextual information is a confounder that affects both user representations and prediction scores. To make context information beneficial for semantic-aware intent learning, we resort to a causal inference technique, namely, backdoor adjustment[154], to adjust the user representations in the disentangled causal model. By doing so, we aim to produce unbiased user representations.

Backdoor Adjustment. Note that previous recommenders build predictive models $P(Y \mid U)$ from the passively collected interaction dataset, which neglect the effect of confounder C and lead to spurious correlations between users and items. The spurious correlation is harmful since the items in the majority group are likely to dominate the recommendation list and narrow down the user interests. According to the theory of backdoor adjustment [154], we aim to remove the harmful effect of context information C on user representation U. Instead of $P(Y \mid U)$, we formulate the predictive model as $P(Y|do(U = u))$ to account for the effect of confounder. Based on the graph in Figure. 4.13, we first need to formulate the causal effect between variables by causal intervention, which is denoted by $do(\cdot)$ operation [154]. The operation $do(U = u)$ is defined to externally assign a certain value to U. Namely, $do(U = u)$ intuitively removes the edge $C \rightarrow U$ so as to block the effect of C on U, making the value of U independent of its causes (*cf.* Figure. 4.13). By applying do operation, we can estimate the effect of C on Y by:

$$P(Y|do(U = u)) - P(Y|do(U = 0)) \tag{4.29}$$

where $P(Y|do(U = 0))$ denotes the null intervention, e.g., the baseline compared to $U = u$. In the physical world, $P(Y|do(U = u))$ corresponds to actively manipulating the aspects or attributes in the item.

Our implementation is inspired by two inherent properties of any Heterogeneous Network Embedding method (e.g., metapath2vec++ [41]). First, by passing a meta-path into the pre-trained context model in Eq. (4.18), we have the context embedding $\boldsymbol{c_a}$ for an aspect (e.g., *actor*). Aggregating context embeddings for all aspects into the aspect representation set C, we can obtain the unified aspect representation C, where each element of C, i.e., $\boldsymbol{c_a}$, representing one semantic aspect. As C is no longer correlated with $\boldsymbol{u}^u$ by removing the edge $C \rightarrow U$, the causal intervention makes $\boldsymbol{u}^u$ has a fair opportunity to incorporate every context $\boldsymbol{c_a}$ into the prediction of $\hat{\boldsymbol{y}}_{ui}$, subject to a prior $P(C = \boldsymbol{c_a})$. Second, prevailing pre-trained models use specific tasks (in our method is the semantic-aware intent learning in Section 4.2.3.4) as the objective, the representations

trained from it can be considered as the distilled information $\boldsymbol{u}^u$ that waits to be adjusted by $do(U = \boldsymbol{u}^u \odot C)$. By far, we have the context information C for all aspects, where each $\boldsymbol{c_a} \in C$ represents the context embedding of one aspect. We also have the refined intent representation $\boldsymbol{u}^u$ that waits to be adjusted from Eq. (4.28) as U. Next, we will detail the causal intervention by providing the implementation for Eq. (4.29).

The overall backdoor adjustment is achieved through:

$$
\begin{aligned}
& P(Y \mid U, do(U = u)) - P(Y \mid do(U = 0)) \\
&= \sum_{C} \left(P\left(Y \mid do\left(U = \boldsymbol{u}^u \odot C\right)\right) - P\left(Y \mid do(U = 0)\right) \right) P\left(C = \boldsymbol{c_a}\right) \\
&= \frac{1}{N} \sum_{i=1}^{N} \left(P\left(\hat{\boldsymbol{y}}_{ui} \mid \boldsymbol{u}^u \odot C\right) - P\left(\hat{\boldsymbol{y}}_{ui} \mid \boldsymbol{u}^u\right) \right)
\end{aligned}
\tag{4.30}
$$

where each component in Eq. (4.30) is designed by:

- $C = \boldsymbol{c_a}$ indicates the confounder C is set as one context embedding $\boldsymbol{c_a}$.
- $P(Y|U, do(U = u)) = P\left(\hat{\boldsymbol{y}}_{ui} \mid \boldsymbol{u}^u \odot C\right)$. The selected context embedding C is concatenated with $\boldsymbol{u}^u$ by the element-wise product $\odot$, which serves as the adjustment value for the prediction of $\hat{\boldsymbol{y}}_{ui}$.
- $P(C = \boldsymbol{c_a})$ is the prior distribution of different item aspect, by defining $P(C = \boldsymbol{c_a}) = 1/N$, we can assume a uniform prior for the adjusted features, where N is the total number of aspect type.

We then utilize an inference strategy to adaptively fuse the prediction scores from the $P\left(\hat{\boldsymbol{y}}_{ui} \mid \boldsymbol{u}^u \odot C\right)$ and $P\left(\hat{\boldsymbol{y}}_{ui} \mid \boldsymbol{u}^u\right)$. Specifically, we first train the recommender models by $P\left(Y \mid do\left(\hat{\boldsymbol{y}}_{ui} = \boldsymbol{u}^u \odot C\right)\right)$ and $P\left(Y \mid do\left(\hat{\boldsymbol{y}}_{ui} = 0\right)\right)$ and obtain $\hat{\boldsymbol{y}}_C$ and $\hat{\boldsymbol{y}}$, respectively. Then, the adjusted prediction score $\hat{\boldsymbol{y}}_C$ and unadjusted prediction score $\hat{\boldsymbol{y}}$ are automatically fused to regulate the impact of backdoor adjustment. We define an indicator function I_a that determines whether to include $\boldsymbol{c_a}$ into the user intent $\boldsymbol{u}^u$ or not.

$$
\mathrm{I}_a := \begin{cases} \boldsymbol{c_a} & \text{if } \tanh\left(\hat{\boldsymbol{y}}_C - \hat{\boldsymbol{y}}\right) > 0 \\ 1 & \text{if } \tanh\left(\hat{\boldsymbol{y}}_C - \hat{\boldsymbol{y}}\right) < 0 \end{cases}
\tag{4.31}
$$

where $\tanh(\hat{\boldsymbol{y}}_C - \hat{\boldsymbol{y}}) > 0$ indicates that the backdoor adjustment leads a positive impact on recommendation result by considering the aspect $\boldsymbol{c_a}$. Otherwise, $\boldsymbol{c_a}$ leads a negative

impact, which should be removed from the user intent representation. Based on Eq. (4.31), the semantic-aware user intent representation can be refined as follows:

$$\boldsymbol{e_a} = \boldsymbol{u}^u \odot \mathrm{I}_a \tag{4.32}$$

To define the unbiased loss function for the observation $\hat{\boldsymbol{y}}_{ui}$, we aim to maximize the discrepancy between the final adjusted and the unadjusted representation $\boldsymbol{u}^u$ under the guidance of $\boldsymbol{e_a}$, that is,

$$\mathscr{L}_d = \arg\min_{\bar{\theta}} \sum_{(u,i,\boldsymbol{y}_{ui})\in O} (\boldsymbol{y}_{ui}, f(\boldsymbol{u}, \boldsymbol{i}, \prod_{a}^{N} \boldsymbol{e_a})) \tag{4.33}$$

where $\boldsymbol{u}$, $\boldsymbol{i}$ are the ID embeddings for user u and item i, $f(\cdot)$ is defined in Eq. (4.27), and $\boldsymbol{y}_{ui}$ is the ground-truth for the interaction between user u and item i.

4.2.3.6 Optimization

Our model ultimately has three loss functions, i.e., $\mathscr{L}_d$ of unbiased preference score estimation loss given in Eq. (4.33), $\mathscr{L}_\theta$ of *Heterogeneous Skip-Gram* model given in Eq. (4.17), and the BPR loss given in Eq. (4.28). To this end, the objective function of our CaDSI method could be derived as:

$$\mathscr{L} = \lambda_d \mathscr{L}_d + \lambda_\theta \mathscr{L}_\theta + \lambda_z \mathscr{L}_{BPR} + \mathscr{R}(\Omega) \tag{4.34}$$

where Ω represents the trainable parameters and $\mathscr{R}(\cdot)$ is a squared L_2 norm regularization term on Ω to alleviate the overfitting problem. λ_d, λ_θ, λ_z are trade-off hyper-parameters of the three separate loss functions respectively. During the training, we optimize the objective function in Eq. (4.34) via gradient descent algorithm.

4.2.3.7 Computational Complexity Analysis

CaDSI consists of multiple components, each performing different tasks for learning semantic-aware intent representations in Heterogeneous Information Networks (HINs). The pre-trained model takes a HIN as input and learns node representations using Meta Path-Based Random Walks and Heterogeneous Skip-Gram. Let N be the number of nodes and E the number of edges in the HIN. Let r be the number of random walks per node and l be the walk length, the cost of performing random walks for all nodes is $O(r \cdot l \cdot N)$. The

Heterogeneous Skip-Gram takes generated node sequences and learns node embeddings. Let d be the embedding size and w be the context window size, the computational cost of Skip-Gram with negative sampling is $O(N \cdot d + E \cdot w \cdot d)$. For the GNN-based disentangling module, the cost per L layer for N nodes and E edges is $O(EH + NH^2)$. The FM operator in semantic-aware intent learning models pairwise feature interactions, for M users and K items, the cost is $O(MKd)$. Backdoor adjustment refines the semantic-aware intent representations by deconfounding causal effects. Assume B represents the number of confounding variables, the cost is $O(B^2 d)$. Thus, the overall computational complexity is $O\left(r \cdot l \cdot N + N \cdot d + E \cdot w \cdot d + m \cdot N \cdot d + L\left(EH + NH^2\right) + MKd + B^2 d\right)$.

4.2.4 Experiments

To more thoroughly evaluate the proposed methd, experiments are conducted to answer the following research questions:

- **RQ1**. How confounding bias caused by the context information is manifested in real-world recommendation datasets?
- **RQ2**. How does our model perform compared with state-of-the-art models for Top-K recommendation?
- **RQ3**. How does key components in our model impact the recommendation performance (i.e., disentanglement learning task, causal intervention)? How do hyper-parameters in our model impact recommendation performance?
- **RQ4**. How does our model interprets user intents for recommendations?

We first present the experimental settings for good reproducibility and answer the four research questions above.

4.2.4.1 Setup

Datasets. We evaluate our model on three publicly accessible datasets for the Top-K recommendation. The statistics of the datasets are summarized in Table 4.4, and the selected meta paths for all data sets are shown in Table 4.5. To ensure the quality of all the datasets, we use the core settings, i.e., we transform explicit ratings into implicit data, where each interaction between users and items is marked as 0 or 1, indicating whether the user has rated the item or not; we retaining users and items with at least five interactions and each user has at least five friends for both of the datasets. In the

Table 4.4: Statistics of datasets.

Dataset (Density)	Node	Relation A-B	Avg.Degree of A/B
MovieLens-HetRec (4.0%)	#User(U): 2,113 #Movie(M): 10,109 #Actor(A): 38,044 #Director(D): 4,031 #Country(C): 72 #Genre(G): 20	#U-M: 855,598 #U-U: 0 #M-A: 95,777 #M-D: 10,068 #M-C: 10,109 #M-G: 20,670	#U/M: 405.0/84.6 #U/U: 0/0 #M/A: 9.5/2.5 #M/D: 1.0/2.5 #M/C: 1.0/140.0 #M/G: 2.0/1033.5
Douban Book (0.27%)	#User(U): 13,024 #Book(Bo): 22,347 #Group(Gr): 2,936 #Author(Au): 10,805 #Publisher(P): 1,815 #Year(Y): 64	#U-Bo: 792,062 #U-U: 169,150 #U-Gr: 1,189,271 #Bo-Au: 21,907 #Bo-P: 21,773 #Bo-Y: 21,192	#U/Bo: 60.8/35.4 #U/U: 13.0/13.0 #U/Gr: 91.3/405.1 #Bo/Au: 1.0/2.0 #Bo/P: 1.0/12.0 #Bo/Y: 1.0/331.1
Douban Movie (0.63%)	#User(U): 13,367 #Movie(M): 12,677 #Group(Gr): 2,753 #Actor(A): 6,311 #Director(D): 2,449 #Type(T): 38	#U-M: 1,068,278 #U-U: 4,085 #U-Gr: 570,047 #M-A: 33,587 #M-D: 11,276 #M-T: 27,668	#U-M: 79.9/84.3 #U/U: 1.7/1.8 #U/Gr: 42.7/207.1 #M-A: 2.9/5.3 #M/D: 1.1/4.6 #M/T: 2.2/728.1

Table 4.5: The selected meta paths for datasets.

Dataset	Meta path Schemes
MovieLens-HetRec	UMU, UMAMU, UMDMU, UMCMU, UMGMU MUM, MAM, MDM, MCM, MGM
Douban Book	UBoU, UBoAuBoU, UBoPBoU, UBoYBoU, UBoAuBoU BoUBo, BoPBo, BoYBo, BoAuBo
Douban Movie	UMU, UMDMU, UMAMU, UMTMU MUM, MAM, MDM, MTM

training phase, each observed user-item interaction is treated as a positive instance. We use negative sampling to randomly sample an unobserved item and pair it with the user as a negative instance.

Baselines. To demonstrate the effectiveness, we compare our model with four classes of methods: (I) conventional entangled CF methods; (II) graph-based entangled methods; (III) HIN enhanced entangled methods, which model user-item interaction with rich context information in a HIN; (IV) disentangled methods, which disentangle user intents or item aspects with different mechanisms; (V) causal-based recommendation methods.

- **NeuMF** [81] (I): This method combines deep neural networks with Matrix Factorization (MF) method for modeling user-item interactions.
- **GC-MC** [10] (II): The method organizes user behaviors as a graph, and employs one Graph Convolution Network (GCN) encoder to generate representations based

on first-order connectivity.

- **NGCF** [205] (II): This adopts three Graph Neural Network(GNN) layers to model at most third-order connectivity on the user-item interaction graph.
- **LightGCN** [80] (II): This is a state-of-the-art graph-based recommendation method that learns user/item embeddings by linearly propagating them with neighbors aggregation in the GCN component.
- **IF-BPR** [245] (III): This method leverages meta path-based social relations derived from a HIN, and proposes a social recommmender that can capture the similarity of users.
- **MCRec** [88] (III): This method leverages meta path-based context with co-attention mechanism for Top-K recommendation.
- **NeuACF** [75] (IV): This method disentangles multiple aspects of users and items with a deep neural network for HIN-enhanced recommendations.
- **MacridVAE** [134] (IV): This method disentangle user intents behind user behaviors, assuming that the co-existence of macro and micro latent factors affect user behaviors.
- **DGCF** [207] (IV): This is a state-of-the-art CF-based disentangled recommendation method, which disentangles latent factors of user intents by the neighbor routing and embedding propagation.
- **DICE** [261] (V): This is a state-of-the-art causal-based recommendation method, which aims at disentangling users' interest by controlling the conformity bias using causal embedding.

Evaluation Metrics. We adopt two popular metrics: Recall@K and Normalized Discounted Cumulative Gain(NDCG)@K to evaluate the Top-K recommendation performance of our model. The K is set as 20 by default. In the inference phase, we view the historical items of a user in the test set as the positive, and evaluate how well these items are ranked higher than all unobserved ones. The average results w.r.t. the metrics over all users are reported.

Parameter Settings. We implement all baseline models and our proposed CaDSI model on a Linux server with Tesla P100 PCI-E 16GB GPU. For a fair comparison, datasets for implementing all models are split as train/test/validate set with a proportion of

80%/10%/10% of the dataset, while we optimize all models with Adam. For a fair comparison, a grid search is conducted to choose the optimal parameter settings, e.g., dimension of user/item latent vector k_{MF} for matrix factorization-based models and dimension of embedding vector d for neural network-based models. The embedding size is initialized with the Xavier and searched in $\{8, 16, 32, 64, 128, 256\}$. The batch size and learning rate are searched in $\{32, 64, 128, 512, 1024\}$ and $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, respectively. The maximum epoch N_{epoch} is set as 2000 and an early stopping strategy is performed. Moreover, we employ three hidden layers for the neural components of GC-MC NGCF, LightGCN, MCRec, NeuACF, MacridVAE and DGCF. The hyperparameter specifications of CaDSI are set as: latent intents number k as 4, iteration number of disentangling module $\boldsymbol{l}$ as 2, model depth of disentangling module L as 2, iteration number of causal intervention n as 140, and their influences are reported in Section 4.2.4.4.

4.2.4.2 Understanding Confounders (RQ1)

We initially conduct an experiment to understand to what extent the confounding bias exists in meta paths of real-world recommendation datasets. To this end, we aim to investigate the distribution of nodes among the same meta path. Intuitively, an unbiased HIN-based recommendation method should expect that, for a specific attribute, each user/item should hold an equal number of this attribute (i.e., interactions between nodes and attributes are likely to be evenly distributed). Thus, we investigate the confounding bias by analyzing the statistics of node-attribute interactions of meta paths in `Douban Book`. We randomly sample $n = 100$ books from `Douban Book` and extract their *Author*, *Publisher*, and *Year* attributes. By counting the connections between books and their attributes whose type belongs to *Author*, *Publisher*, and *Year*, respectively, we have the statistical results shown in Figure 4.15. The connected graphs in the left part of Figure 4.15 depict whether the book i connected with the selected attribute. The figures in the right part of Figure 4.15 show the distributions of connected book numbers by a certain attribute, where the y-axis denotes the total amount of the connected books.

Apparently, attributes and books exhibit an unevenly distribution regarding their interactions: the attributes in dataset are partially observed, leaving a larger number of attributes to be unobserved. For example, for *Book-Author* meta path, there are a lot of books that do not connect with any node whose type is *Author*, which means lots of author attributes of books are missing. In conventional recommendation methods, the missing pattern of such attributes is ignored by either regarding them as outliers and padding them with random values, or treating the missing attributes as negative

feedback. Such measure would preserve the confoundings bought by node attributes, degrading the recommendation ultimately.

Another finding is that the distribution for book-attribute connection numbers is significantly skewed: it displays a long-tail phenomenon where the green vertical line separates the top 50% of connection numbers by popularity; these connections outweigh another 50% long-tail connections to the right. For instance, in Figure 4.15 (a), authors in the *Book-Author* relation cumulatively connect with 90% more books than the long-tail authors to the right. For *Book-Publisher* meta path, ideally, *Book-Publisher* has the one-to-one relation from book to the publisher, while every publisher has published an equal number of books. However, some publishers have published at most 510 books, while more than 90% of publishers only published fewer than 10 books. Such long-tail distribution can bias the users' interest on item aspects. i.e., recommendation methods tend to recommend items with the most frequent attribute, while users can only be exposed to recommended items.

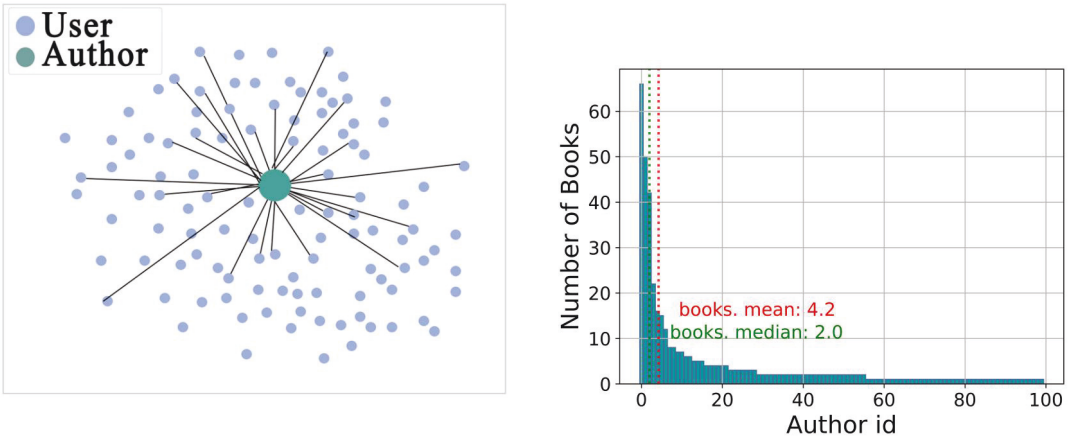
4.2.4.3 Performance Comparison (RQ2)

Table 4.6: Performance comparison.

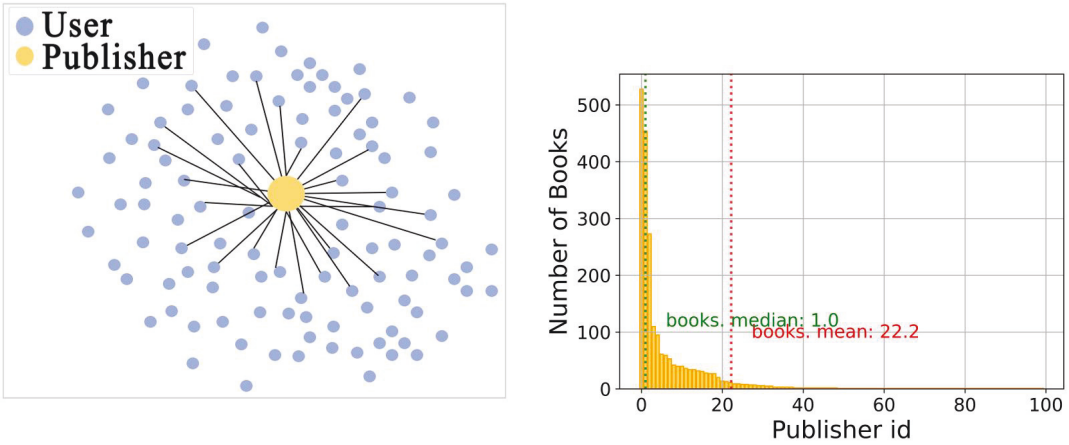
	MovieLens-HetRec				Douban Book				Douban Movie			
	Recall@20	NDCG@20	Recall@40	NDCG@40	Recall@20	NDCG@20	Recall@40	NDCG@40	Recall@20	NDCG@20	Recall@40	NDCG@40
NeuMF	0.0434	0.0557	0.0665	0.0709	0.0339	0.0391	0.0641	0.0682	0.0460	0.0417	0.0708	0.0611
GC-MC	0.0336	0.0404	0.0653	0.0584	0.0458	0.0402	0.0675	0.0643	0.0448	0.0461	0.0602	0.0622
NGCF	0.0365	0.0508	0.0699	0.0615	0.0252	0.0301	0.0707	0.0691	0.0475	0.0498	0.0689	0.0642
LightGCN	0.0466	0.0155	0.0615	0.0498	0.0201	0.0225	0.0531	0.0568	0.0294	0.0331	0.0499	0.0578
IF-BPR	0.0546	0.0510	0.0727	0.0689	0.0396	0.0463	0.0628	0.0601	0.0483	0.0501	0.0652	0.0603
MCRec	0.0352	0.0195	0.0680	0.0677	0.0165	0.0294	0.0481	0.0507	0.0281	0.0336	0.0618	0.0629
NeuACF	0.0236	0.0308	0.0556	0.0684	0.0298	0.0201	0.0601	0.0579	0.0351	0.0438	0.0571	0.0623
MacridVAE	0.0454	0.0290	0.0661	0.0592	0.0309	0.0425	0.0691	0.0645	0.0489	0.0441	0.0729	0.0616
DGCF	0.0229	0.0589	0.0532	0.0708	0.0431	0.0502	0.0649	0.0663	0.0416	0.0527	0.0702	0.0628
DICE	0.0549	0.0499	0.0740	0.0703	0.0577	0.0608	0.0820	0.0799	0.0513	0.0389	0.0811	0.0636
Our model	0.0678*	0.0659*	0.0765*	0.0736*	0.0708*	0.0722*	0.1466*	0.1194*	0.0583*	0.0547*	0.0918*	0.0647*
%improv.	**23.5%**	**11.9%**	**3.4%**	**3.8%**	**22.7%**	**18.8%**	**78.8%**	**49.4%**	**13.6%**	**3.8%**	**13.2%**	**0.8%**

We compare the Top-K recommendation performance of CaDSI with ten recommendation baselines on three datasets: `MovieLens-HetRec`, `Douban Book` and `Douban Movie`. Table 4.6 demonstrates the performance comparison and we have the following observations:

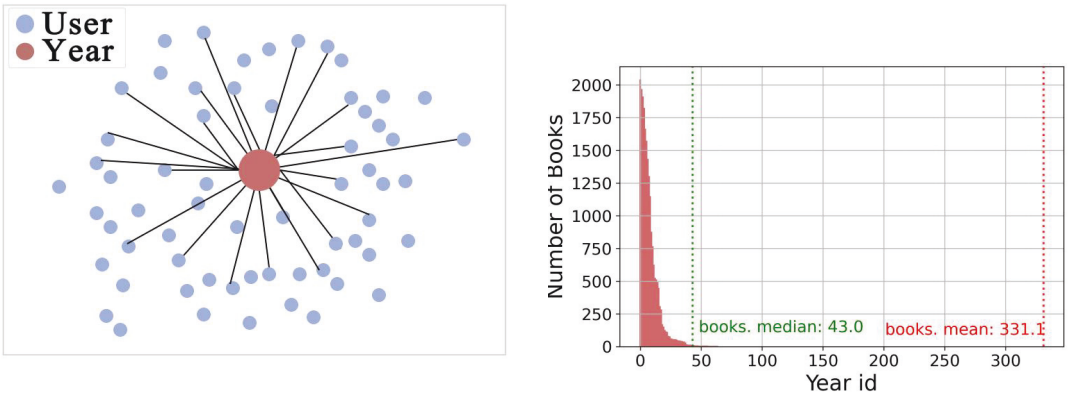
- Our CaDSI consistently yields the best performance among all methods on three datasets. In particular, CaDSI improves over the strongest baselines w.r.t. Recall@20 by 23.5%, 22.7%, 13.6% , NDCG@20 by 11.9%, 18.8%, 3.8%, Recall@40 by 3.4%, 78.8%, 13.2% and NDCG@40 by 3.8%, 49.4%, 0.8% on `MovieLens-HetRec`, `Douban Book` and `Douban Movie` respectively. CaDSI outperforms all baseline



(a) Distribution of $Book-Author$.



(b) Distribution of $Book-Publisher$.



(c) Distribution of $Book-Year$.

Figure 4.15: The distributions of $Book-Author$, $Book-Publisher$ and $Book-Year$ of `Douban Book` dataset.

methods on Top-K recommendation task, which validates that the semantic-aware user intents representation can enhance the recommendation performance.

- In virtue of user-item interaction graph and meta paths, GNN-based (GC-MC, NGCF and LightGCN) and HIN-based (IF-BPR, MCRec) recommendation methods can achieve better performance than conventional MF methods (NeuMF) in most cases. However, they ignore controlling the bias existing in the context information. On the contrary, our CaDSI adopts a principled causal inference way to easing such confounding bias. So it outperforms GNN-based and HIN-based recommendation methods on both of the datasets. For instance, our CaDSI outperforms the most competitive HIN-based recommender IF-BPR w.r.t. Recall@20/Recall@40 by 24.2%/5.2% and NDCG@20/NDCG@40 by 29.2%/6.8% on `MovieLens-HetRec`.

- By performing unbiased disentanglement via semantics context, our CaDSI can infer user's potential interests of items. However, those user interests could not be well inferred from other disentangled recommendation methods (NeuACF, MacridVAE and DGCF).

- Among the GNN-based recommenders (GC-MC, NGCF and LightGCN), HIN-based recommenders (IF-BPR, MCRec) and disentangled recommenders (NeuACF, MacridVAE and DGCF), the causal-based disentangled method (DICE) serves as the strongest baseline in most cases. This justifies the effectiveness of easing the confounding bias in context information when estimating disentangled users' interests. However, DICE performs worse than our CaDSI, as it ignores rich semantics information in HIN and fails to ingest semantics aspects when disentangling user interests.

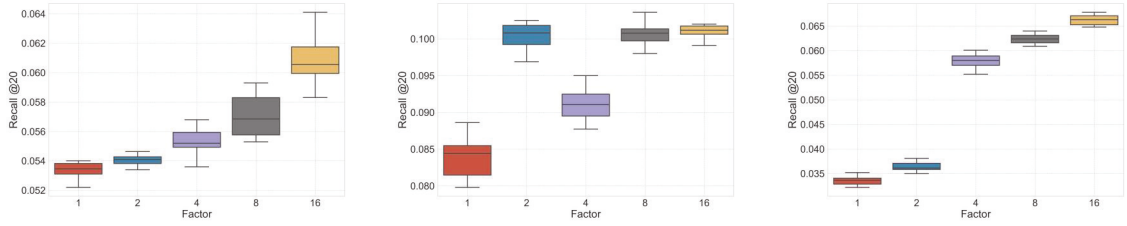
- From movie recommendation datasets, we can find that the improvements on `MovieLens-HetRec` is bigger than that on `Douban Movie`. This is reasonable since `Douban Movie` is much more sparser than `MovieLens-HetRec` with density of 0.63% vs. 4.0%, respectively. However, our CaDSI has better performance than all baselines on `Douban Movie`, because it achieves unbiased evaluation on high-order connectivity and rich semantics. This indicates that CaDSI is robust to the very sparse datasets.

4.2.4.4 Study of CaDSI (RQ3)

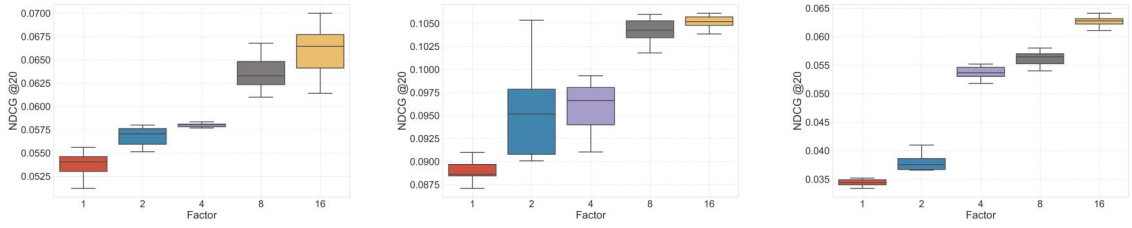
Ablation studies on CaDSI are also conducted to investigate rationality and effectiveness. Specifically, we first attempt to exploit how the *disentangled learning* and *causal intervention* affect our performance. Moreover, the stability of our approach's performance on the Top-K recommendation is validated as well.

We have one fixed parameter $n = 140$ (*cf.* Eq. (4.30)) which denotes the total number of causal intervention times. Three important hyperparameters k (*cf.* Eq. (4.19)), L (*cf.* Eq. (4.25)) and K correspond to: the number of latent factors of user intents, the number of graph disentangling layers and the number of items in Top-K recommendation list, respectively. Based on the hyperparameter setup, for all questions listed above, we vary the value of one parameter while keeping the others unchanged.

Effect of Disentanglement Learning. The intent number k controls the total amount of user intents considered in our model; larger k stands for more fine-grained disentangled user intents. To study the influence, we vary k in the range of $\{1,2,4,8,16\}$ and show the corresponding performance comparison on `MovieLens-HetRec` `Douban Book`, `Douban Movie` in Figure 4.16. We have several observations.



(a) Recall@20 on `MovieLens-HetRec`, `Douban Book` and `Douban Movie`.



(b) NDCG@20 on `MovieLens-HetRec`, `Douban Book` and `Douban Movie`.

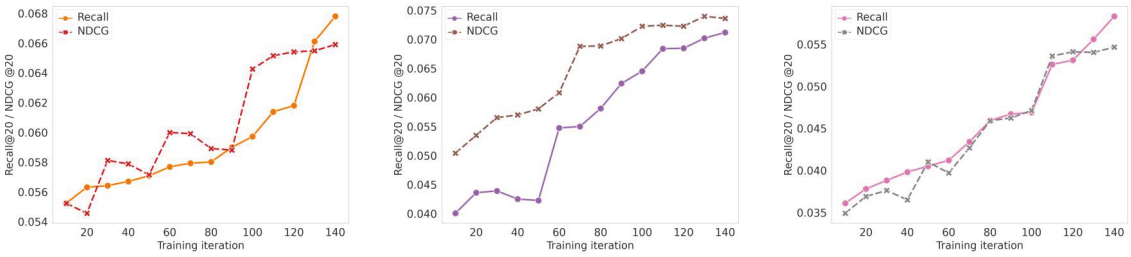
Figure 4.16: The recommendation performance comparison under different latent user intent factors.

- Increasing the intent number from 1 to 16 can significantly enhance the performance, while CaDSI performs the worst when $k = 1$. This indicates learning the

disentangled user intents is effective to capture the real user preferences towards items instead of coupling all preference together.

- The variations are diverse across different datasets. For `MovieLens-HetRec` and `Douban Movie`, the performance of CaDSI increase steadily as the k value increases from 1 to 16, while the performance drops when k is set from 2 to 4 on `Douban Book`. One possible reason is that CaDSI should balance between too fine-grained disentangled intents and the adjustment from causal intervention, such balancing learning is more obvious when dataset size is lager.

Effect of Causal Intervention. To investigate whether CaDSI can benefit from causal intervention, we study the performance of CaDSI by varying the iterations of causal intervention. Figure 4.17 summarizes the experimental results w.r.t. `MovieLens-HetRec`, `Douban Book`, `Douban Movie` and we have the following observations:



(a) MovieLens-HetRec. (b) Douban Book. (c) Douban Movie.

Figure 4.17: Impact of causal intervention on the recommendation performance of our CaDSI along with iterations.

- Clearly, the causal intervention mechanism renders our CaDSI a better recommendation performance: more iterations of causal intervention lead to better recommendation performance before saturation on all datasets, e.g., the Recall@20 and NDCG@20 values generally increase along with training iterations in Figure 4.17.

- When training iterations reach to 130, 70 and 110 for `MovieLens-HetRec`, `Douban Book` and `Douban Movie`, respectively, the performance becomes relatively stable. Moreover, `Douban Book` requires fewer iteration times than the other two datasets. Intuitively, purchasing books is a much more simple behavior than choosing movies. Thus, user intents on book aspects are less diverse, leading to a quick convergence to the optimal interventional representations.

Table 4.7: Impact of multi-order connectivity, i.e., graph propagation layer number L.

	MovieLens-HetRec		Douban Book		Douban Movie	
Layer number	Recall	NDCG	Recall	NDCG	Recall	NDCG
1	0.0505	0.0521	0.0651	0.0684	0.0583	0.0546
2	0.0672	0.0683	0.0712	0.0736	0.0596	0.0570
3	0.0611	0.0624	0.0682	0.0701	0.0562	0.0573

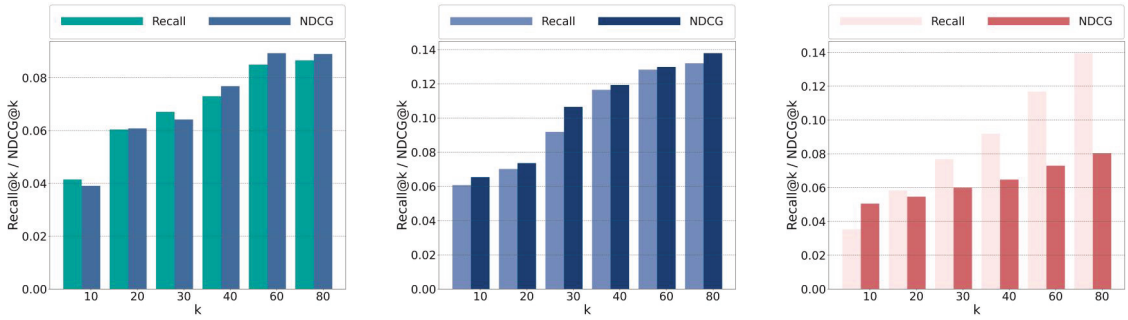
- Some fluctuations appear in the iteration process, especially on MovieLens-HetRec dataset. The size of MovieLens-HetRec is much smaller than the other two datasets, thus leading to instability to intervention process due to the data sparsity. However, when carrying more iterations, small-size datasets such as MovieLens-HetRec can also yield satisfying results.

Effect of Multi-order Connectivity. Since CaDSI is benefited from the higher-order connectivity between complex interactions and context information, we investigate how connectivity degrees affect the CaDSI. Specifically, we search the graph disentangling layer number L in the range of $\{1,2,3\}$, which correspond to first-order connectivity, second-order connectivity and third-order connectivity, respectively. We show the performance comparison in Table 4.7 and below are our observations.

- More graph disentangling layers will collect more information form multi-hop neighbors from a holistic user-item interaction graph. Clearly, the performance of our CaDSI with layer number $L = 2$ is better than that with $L = 1$, since the second-order connectivity can capture significant collaborative signals of users and items.

- When stacking more than 2 layers, the influence of multi-hop neighbors is small and the recommendation performance is degraded. This is reasonable since too informative signals of user-item interactions might introduce additional noises to the representation learning. This again emphasizes the importance of controlling the bias bought by context information.

Top-K Recommendation Performance. Based on Recall@K and NDCG@K, Figure 4.18 shows that CaDSI achieves the stable performance on Top-K recommendation when K (i.e., the length of the ranking list) varies from 10 to 80. This indicates that our CaDSI performs stably on Top-K recommendation task and can recommend more relevant items within Top-K positions when the ranking list length increases.



(a) MovieLens-HetRec. (b) Douban Book. (c) Douban Movie.

Figure 4.18: Performance of CaDSI in terms of Recall@K and NDCG@K under different K.

4.2.4.5 Case Study and Visualization(RQ4)

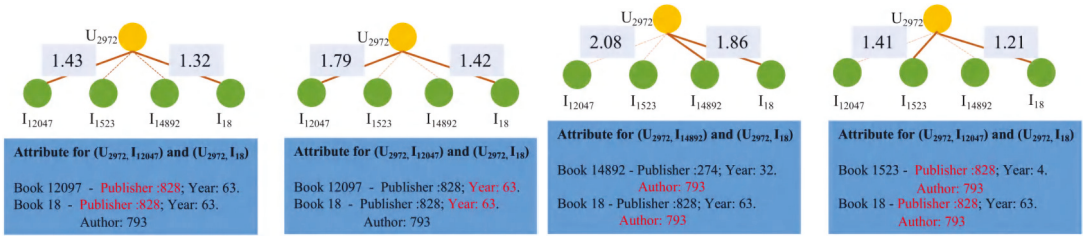


Figure 4.19: Visualization of the disentangled user intent graphs based on score matrices. User-item interactions with the highest scores are marked in solid lines; item attributes with the same values are highlighted in red.

We conduct an experiment to understand the disentanglement of user intents by our CaDSI, then explore whether such intent is related to real-world item semantics. We select a user $u2972$ from `Douban Book` and learn its interaction scores $S(u,i)$ with his/her historical interacted items under our CaDSI. The user intents factor $k = 4$ indicates four distinct user intents. Thereafter, we randomly select four items from the interaction score matrices. For each interaction under different k, we mark the interaction scores with the highest confidence with solid lines and couple the certain item attributes below them. Figure 4.19 shows the visualization results, and we have the following findings:

- Jointly analyzing intent-aware user-item interaction graphs, we can see user preference differs across each graph, reflected by different interaction scores in each intent-aware graph. For example, $u2972$ interacts with $i12047$ with a preference score of 1.43 under intent k_1, while the score changes to 1.79 under intent k_2. This

demonstrates the importance of disentangling user intents in recommendation scenarios.

- We thereafter couple item attributes to investigate whether user intents are related to item semantics. It can be seen that different fine-grained user intents are highly consistent with high-level item semantics. For instance, intent k_3 contributes mostly to interactions $(u2972, i14892)$ and $(u2972, i18)$, which suggests its high confidence as being the intent behind these behaviors. When switching to item attributes of $i12047$ and $i18$, the same *Author* attribute has been id 793 can be found, reflecting the reason why $u2972$ chose to interact with $i12047$ and $i18$. This demonstrated that our CaDSI, which aims at disentangling user intents while assigning specific item semantics to the learned intents, is effective in the disentanglement of user intents towards item aspects.

4.2.5 Summary

This research work has researched the popularity bias issue stemming from different aspects and proposes an unbiased and robust CaDSI model for context-aware recommendation. The CaDSI can provide semantics to fine-grained representations for disentangling user intents, while easing the bias stemming from unevenly distributed item aspects. We evaluate our CaDSI on three real-world recommendation datasets, with extensive experiments and visualizations demonstrate the robustness and interpretability of our semantic-aware user intent representation.

CHAPTER 5

IMPROVING RECOMMENDATIONS USING CAUSAL INFERENCE

This chapter presents research work that improves the accuracy of recommendations by refining 1) the architecture of the neural network and 2) the objective of optimizing the model. The NCGCF model (Neural causal graph Collaborative Filtering) connects the Graph Convolutional Network (GCN) to the Structural Causal Model (SCM) by using a Neural Causal Model, thereby improving the accuracy of the conventional graph collaborative filtering method. The HINpolicy model (HIN augmented off-policy learning) uses counterfactual risk minimization to achieve the unbiased policy optimization of a typical reinforcement learning method, i.e., off-policy learning method.

5.1 Causal Neural Network for Collaborative Filtering

5.1.1 Overview

Research Question. Collaborative Filtering (CF) [170] as an effective remedy has dominated recommendation research for years. Recently, Graph Collaborative Filtering (GCF) has been studied extensively and has become an emerging CF paradigm [89]. GCF enhances traditional CF methods by modeling complex user-item interactions in a graph as well as auxiliary information, e.g., user and item attributes. Thus, GCF has

shown great potential in deriving knowledge (e.g., user behavior patterns) embedded in graphs. Generally, GCF models utilize graph representation learning techniques to derive useful information for downstream CFs. These models use graph neural networks to analyze graph connections and create embeddings, thus improving the optimization of the CF model. Graph Convolutional Networks (GCNs) are among the most widely adopted graph neural networks in GCF models, primarily due to their well-established performance in learning local and global information from large-scale graphs. Therefore, this research aims to investigate leveraging GCNs in the GCF model, thereby capturing users' preferences for improved recommendation performance.

Research Objective. Though promising results are witnessed, leveraging GCN in GCF models presents severe challenges. Compared to traditional methods, such as matrix factorization on pure user-item interactions, graph learning seeks to model complex, multi-hop relationships propagated across graph entities. The complexity of relation modeling increases as the graph becomes more sophisticated by including various types of nodes, e.g., user gender [211], item brands [212]. For example, consider a social network where users are connected based on their friendships, following lists, and followers [216]. Graph learning techniques must not only capture these diverse relationships, but also consider other potential factors, such as user conformity impacts on social content recommendations. However, existing GCN-based GCF methods often fall short of capturing complex graph relationships due to two fundamental drawbacks. Firstly, *they ignore distinguishable node dependencies between neighboring nodes and the target node.* Most GCN-based GCF methods treat all messages from the neighborhood equally, which inevitably overlooks the varying dependencies of neighboring nodes to the target node. However, a user node might have different relations with other neighboring nodes, e.g., item brands. These different relations reflect distinct user preferences, which is the essence of personalized recommendations [215]. By ignoring the difference in relations, the learned user and item embeddings eventually lose expressive power in the recommendation task, i.e., we cannot know which node is the root cause of user interests. Secondly, *they lack an explicit encoding of complex relations between variables in the recommendation.* Most GCN-based GCF methods assume the co-occurrence of users and items is independent. However, user preferences are influenced by various variables in real-world recommendations, such as user conformity caused by user social networks. Discarding these relations leads to the learned embeddings being unable to capture such structural complexity. Therefore, the research objective of this work is to simultaneously capture node dependencies and complex relations between variables to promote more

accurate GCN learning in GCF.

Motivations. Causal modeling sheds light on solving the above drawbacks. On the one hand, causal modeling identifies intrinsic cause-effect relations between nodes and true user preferences [209]. For example, we might treat each neighboring node as the cause (e.g., an item brand) and the user preference as the effect in a causal graph [7]. By estimating the causal effect, we could encode the crucial node dependencies into user and item embeddings to uncover the root causes of user interests. On the other hand, the causal graph is able to model genuine causal relations among the variables in GCFs, capturing variable dependencies inherent in the GCF-based methods. Those causal relations represent the underlying mechanisms driving the recommendation and can be utilized to guide graph learning toward complex user behaviors.

The Proposed Approach. Given the compelling nature of casual modeling in GCN-based GCFs, this paper aims to integrate GCNs and causal models to facilitate causality-aware GCF learning. Motivated by Neural-Causal Connection [231], this paper proposes to connect GCN learning with the Structural Causal Model (SCM). Since the SCM is induced from a causal graph and the GCN works on graph-structured data, the integration of the two models becomes practical. In particular, we first conceptualize the causal graph for the SCM, which is built by revisiting existing CFs and padding their limitations in user preference modeling. Then, we formulate the SCM into a Neural Causal Model, called Neural causal graph Collaborative Filtering (NCGCF). Our NCGCF uses variational inference to approximate structural equations as trainable neural networks, making the learned graph embeddings equally expressive as the causal effects modeled by the SCM. The integration of causal modeling and graph representation learning offers a novel perspective to facilitate accurate recommendations.

Contributions. The contributions of this work are:

- We complete the Neural-Causal Connection for causal modeling of graph convolutional network in recommendations.
- Our proposed NCGCF is the first Neural Causal Model for graph collaborative filtering, which generates causality-aware graph embeddings for enhanced recommendations.
- We validate the effectiveness of our proposed framework through extensive experiments. Our experimental results demonstrate that our approach outperforms existing methods in achieving satisfactory recommendation performance.

5.1.2 Neural Causal Model for Neural Networks

We first provide the motivations for defining our causal graph. We then give our task formulation, which covers detailed steps toward connecting the GCN with the Structural Causal Model.

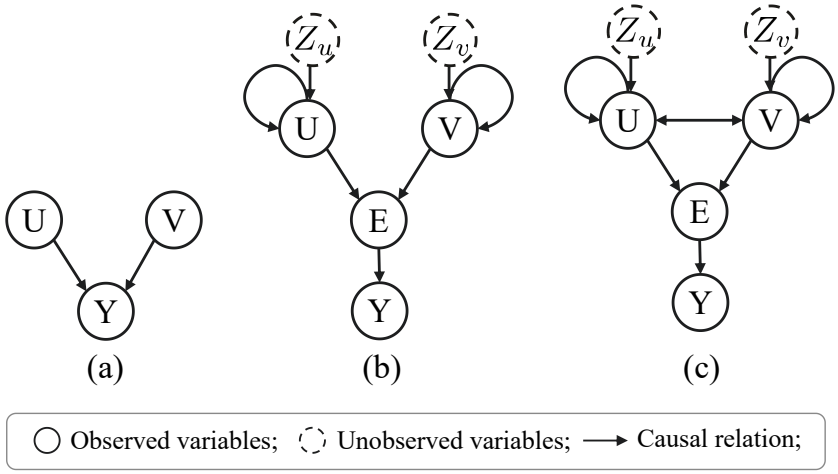


Figure 5.1: Paradigms of user preference modeling in a class of CFs: (a) Early CF, (b) GCF, and (c) Our causality-aware GCF.

5.1.2.1 The Proposed Causal Graph

Following Definition 3.1.2, we start by providing the causal graphs of a class of CF methods, including early CF methods in Figure 5.1 (a) and existing GCF methods in Figure 5.1 (b). Specifically, we aim to show the fundamental drawback shared by these two types of methods: they are fragile in capturing complex user-item relations by assuming the co-occurrence of users and items is independent. We put forward our defined causal graph in Figure 5.1 (c), which considers user-item dependencies for better user preference modeling.

Early CFs largely resort to user-item associative matching [81] and follow the causal graph shown in Figure 5.1 (a), where user node U and item node V constitute a collider to affect the recommendation result Y. For example, matrix factorization typically assumes $P(Y = 1 \mid u, v) \propto \mathrm{u}^\top \mathrm{v}$, where u and v are user and item IDs and the probability of recommendations Y is estimated from the matching of the inner product between u and v. The methods based on latent factors assume $P(Y = 1 \mid u, v) \propto \mathrm{LFM}(u)^\top \mathrm{LFM}(v)$, where LFM is a latent factor model that learns the latent vectors of the user and the

item, and a simple inner product is used for the matching of similarity to determine recommendations.

As shown in Figure 5.1 (b), GCF works on graph-structure data to consider auxiliary information, e.g., user/item attributes, which potentially captures exogenous variables Z_u and Z_v, e.g., user conformity, item exposure. In addition, since user-user and item-item relations are propagated through multihop neighbors within the graph, GCF can capture the inner connections of users and items to model more complex user behavior patterns, e.g., collaborative behavior [205]. However, existing GCF methods still assume the independence between users and items. This is because user and item embeddings are learned separately from the graph representation learning and then subsequently applied to a CF model for user-item associative matching. For example, NGCF [205] assumes $P(Y=1 \mid u,v) \propto E = \mathrm{CF}(\mathrm{agg}(u, z_u, \mathrm{msg}(N_u)), \mathrm{agg}(v, z_v, \mathrm{msg}(N_v)))$, where CF is a CF model for user-item associative matching. N_u and N_v are neighbor sets for users and items; agg and msg are the aggregation and message passing operations, respectively.

Both Figures 5.1 (a) and (b) assume that the co-occurrence of users and items is independent of the observational data, that is, there is no edge $U \rightarrow V$ or $V \rightarrow U$. However, this assumption is unrealistic in the real world because user behaviors are influenced by the recommended items for various reasons. For example, users may be more likely to click on items if recommended [220], which is also known as the item exposure bias problem. In addition, the exposure of the items is determined by the user preferences estimated from the recommendation model, which is the essence of the personalized recommendation. Therefore, we conceptualize the causal relations under GCN-based GCF as the causal graph in Figure 5.1 (c). Our causal graph includes the modeling of $U \longleftrightarrow V$, so that user-item relations can be captured for better user preference modeling. Using the causal graph in Figure 5.1 (c), the directed edge $(u \rightarrow v) \in \mathscr{E}$ captures the causal relationship of a user u to an item v, where $u \in U$ and $v \in V$ and u is a parent node of v, that is, $u \in pa(v)$. G induces a set of causal adjacency vectors A_u and A_v, which specify the neighbors of a user node u and an item node v, respectively. Each element $\mathrm{A}_u^v = 1$ if $v \in pa(u)$, otherwise, $\mathrm{A}_u^v = 0$. Similarly, $\mathrm{A}_v^u = 1$ if $u \in pa(v)$.

5.1.2.2 Task Formulation

The key innovation of this work is to integrate causal modeling into the learning process of a GCN-based GCF model. The problem can be formulated as follows:

Definition 5.1.1 (Task Formulation)**.** Establish the connection between the GCN-based

GCF model and the causal graph depicted in Figure 5.1 (c). Motivated by Neural-Causal Connection [231], the goal is to approximate a Neural Causal Model (NCM) based on the provided causal graph.

To achieve this goal, we first convert the causal graph into a Structural Causal Model (SCM). Subsequently, the NCM is defined on the basis of the SCM, with each structural equation in the SCM corresponding to a neural network in the NCM. To approximate the neural networks trainable in the NCM, we employ a unified framework described in **Section 5.1.3**. This framework enables causal modeling, making the learned graph embeddings as expressive as the causal effects modeled by the SCM. Overall, through the integration with causal modeling, our approach offers a novel perspective on graph representation learning, leveraging the expressive power of the causality-aware graph embeddings to capture complex causal relations in the recommendation.

5.1.2.3 Neural Causal Model

This section evokes the concept of the Structural Causal Model (SCM) and the Neural Causal Model (NCM). The SCM converts causal relations between the causal graph in Figure 5.1 (c) as structural equations; The NCM defines each of the structural equations as a parameterized neural network.

The causal graph in Figure 5.1 (c) has four variables of interest (i.e., endogenous variables): U (user), V (item), E (preference representation) and Y (recommendation). Besides, two exogenous variables Z_u and Z_v are also involved, representing hidden impacts such as user conformity and item exposure. The causal mechanism of modeling the four endogenous variables $\{U,V,E,Y\}$ is done by a SCM. Following Definition 3.1.1 and the causal relations in Figure 5.1 (c), endogenous variables $\{U,V,E,Y\} = \mathcal{V}$ are modeled by structural equations $\{f_U, f_V, f_E, f_Y\} = \mathscr{F}$. Formally,

$$\mathscr{F}(\mathcal{V},Z) := \begin{cases} U \leftarrow f_U(U,V,Z_u) \\ V \leftarrow f_V(U,V,Z_v) \\ E \leftarrow f_E(U,V) \\ Y \leftarrow f_Y(E) \end{cases} \tag{5.1}$$

These structural equations model the causal relation from a set of causes (e.g., $pa(u)$) to a variable (e.g., $u \in U$) accounting for the impact of exogenous variables (e.g., Z_u).

We now formally introduce Neural-Causal Connection [231], i.e., the connection between deep neural networks (e.g., GCNs) and causal models is made by establishing an NCM. In accordance with Definition 3.5.3, we aim to build an NCM $\mathscr{M}(\theta)$ that models

structural equations defined in Eq. (5.1) as parameterized feedforward neural networks. Formally,

$$\mathscr{M}(\theta) \triangleq \begin{cases} \mathrm{Z}_u \sim \mathscr{N}(0, \mathrm{I}_K), \mathrm{Z}_v \sim \mathscr{N}(0, \mathrm{I}_K), \\ \mathrm{u} \propto f_U = q_{\theta_1}(f_{\phi_1}(\mathrm{Z}_u, f_{\varphi_1}(U \mid U, V))), \\ \mathrm{v} \propto f_V = q_{\theta_2}(f_{\phi_2}(\mathrm{Z}_v, f_{\varphi_2}(V \mid U, V))), \\ \mathrm{e} \propto f_E = p_{\theta_3}(\mathrm{u}, \mathrm{v}), \\ \mathrm{y} \sim f_Y = \mathrm{Multinomial}(N, \mathrm{e}) \end{cases} \tag{5.2}$$

- Z_u, Z_v are mapped into low-dimensional hidden vectors Z_u and Z_v using Gaussian distribution $\mathscr{N}(0, \mathrm{I}_K)$.
- $\mathrm{u} \propto f_U$: user representation u is calculated by a user encoder q_{θ_1}. The user encoder takes as input the aggregated (i.e., f_{ϕ_1}) information of user exogenous variables Z_u and user's causality-aware neighbor messages f_{φ_1}.
- $\mathrm{v} \propto f_V$: item representation v is given by an item encoder q_{θ_2}. The item encoder uses aggregated (i.e., f_{ϕ_2}) information of item exogenous variables Z_v and item's causality-aware neighbor messages f_{φ_2}.
- $\mathrm{e} \propto f_E$: user preference probability e is produced by a collaborative filtering decoder p_{θ_3} by using latent representations u and v.
- $\mathrm{y} \sim f_Y$: user interaction y is sampled from a multinomial distribution with the probability e. N is the user's total interaction number.

5.1.3 Methodology

We now introduce our framework, namely, Neural causal graph Collaborative Filtering (NCGCF). We show the NCGCF framework in Figure 5.2, which includes three major components based on the variational autoencoder structure:

- **Causal Graph Encoder**: approximates f_U and f_V. The causal graph encoder includes a user encoder, an item encoder and a semi-implicit generative model. The semi-implicit generative model calculates causal relations between nodes as causality-aware messages. The user encoder and item encoder then use these causality-aware messages to output user representation and item representation, respectively.

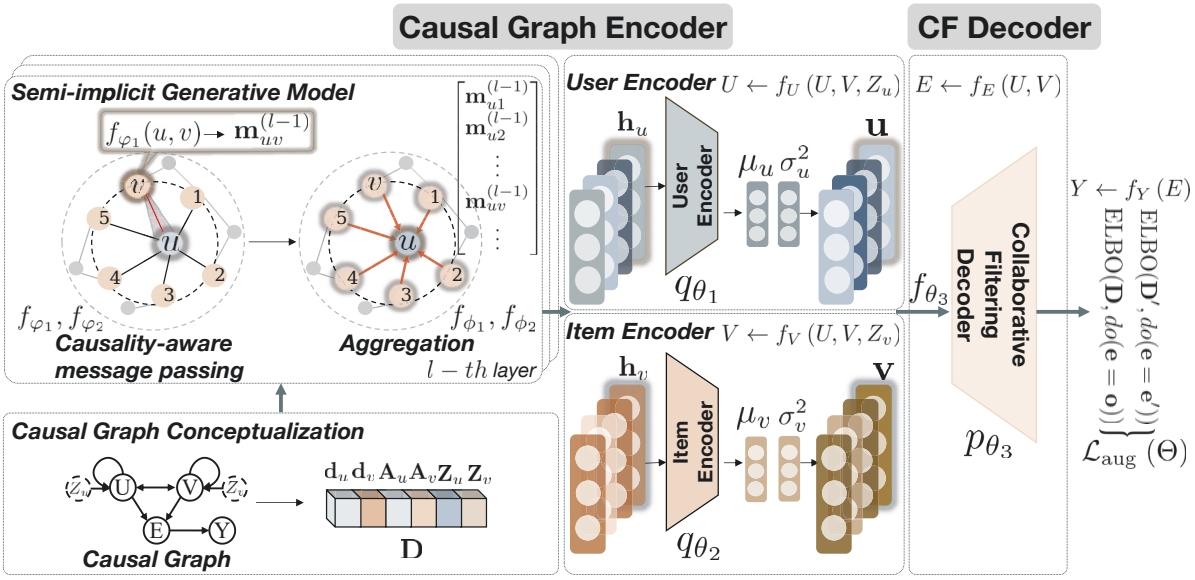


Figure 5.2: NCGCF framework.

- **Collaborative Filtering (CF) Decoder**: approximates f_E using a CF method to estimate user preference.
- **Counterfactual Instances-based Optimization**: optimizes model parameters by implementing f_Y with counterfactual instances to capture user preference shifts.

5.1.3.1 Causal Graph Encoder

The causal graph encoder aims to model f_U and f_V in Eq. (5.2). However, this is not a trivial task as the true posteriors of f_U and f_V do not follow standard Gaussian distributions due to the existence of causal relations between node pairs. Besides, these causal relations should be modeled into causality-aware messages using neural networks. Thus, traditional variational inference [104] that approximates posteriors to simple, tractable Gaussian vectors is not applicable.

Semi-implicit variational inference (SIVI) [243] that models complex distributions through implicit posteriors shows to be an effective alternative. Inspired by SIVI, we devise a semi-implicit generative model on top of the user and item encoder to model implicit posteriors. In particular, the semi-implicit generative model calculates causal relations between nodes as causality-aware messages. Those causality-aware messages are encoded into user and item hidden factors h_u and h_v. Then, the user encoder takes h_u as the input to output the user representation u. Analogously, the item encoder uses h_v to calculate item representation.

Semi-implicit Generative Model. Our semi-implicit generative model contains two operators: *causality-aware message passing* and *aggregation*. The *causality-aware message passing* uses learnable neural networks to model each of the dependency terms for a node and its neighbors within a structural equation. For example, $f_{\varphi_1}(u,v)$ models the dependency between a user node u and his/her neighbor item node v, such that the learned message becomes a descriptor of the causal relation $u \to v$. The *aggregation* uses weighted-sum aggregators to aggregate user/item exogenous variables and the calculated causality-aware neighbor messages. Finally, user and item hidden factors h_u and h_v are output for latter user and item encoder learning.

- *Causality-aware message passing*: For the user encoder, given user u's features d_u and its causal adjacency vector A_u, the messages from u's neighbor v is given by:

$$\begin{aligned} \mathrm{m}_{uv}^{(l-1)} &= f_{\varphi_1}(u,v) = \mathrm{MLP}^{(l)}\left(\mathrm{h}_u^{(l-1)} \| \mathrm{h}_v^{(l-1)}\right) \\ &= \mathrm{ReLU}\left(\mathrm{W}_{\varphi_1}^{(l)}\left(\mathrm{h}_u^{(l-1)} \| \mathrm{h}_v^{(l-1)}\right)\right), \text{ for } l \in \{1,\cdots,L\} \end{aligned} \tag{5.3}$$

  where $\mathrm{m}_{uv}^{(l-1)}$ is the neighbor message at the $l-1$-th graph learning layer [1]. v is a neighbor for u and $v \in N_u \propto \mathrm{A}_u$. $\mathrm{h}_v^{(l-1)}$ and $\mathrm{h}_u^{(l-1)}$ are hidden factors for the neighbor v and the user u at the $l-1$-th layer [2]. $\mathrm{W}_{\varphi_1}^{(l)}$ is the weight matrix for f_{φ_1} at the l-th layer and $\|$ denotes column-wise concatenation. Analogously, for the item encoder, we can calculate the neighbor message $\mathrm{m}_{vu}^{(l-1)}$ for an item v by replacing f_{φ_1} with f_{φ_2} in Eq. (5.3).

- *Aggregation*: For the user encoder, at each graph learning layer l, we perform aggregation operation on the messages $\mathrm{m}_{uv}^{(l-1)}$ from u's neighbors and the user exogenous variables Z_u to obtain the hidden factor $\mathrm{h}_u^{(l)}$:

$$\mathrm{h}_u^{(l)} = \left(\mathrm{h}_u^{(l-1)} \| f_{\phi_1}\left(\left\{\mathrm{W}_{\phi_1}^{(l)} \mathrm{m}_{uv}^{(l-1)} : v \in N_u\right\}\right) \| \mathrm{Z}_u\right) \tag{5.4}$$

  where $\mathrm{h}_u^{(l)}$ is the learned hidden factor for u at the l-th graph learning layer. f_{ϕ_1} is the aggregation operator chosen as weighted-sum, following [4]. $\mathrm{W}_{\phi_1}^{(l)}$ is the weight for f_{ϕ_1} that specifies the different contributions of neighbor messages to the target node at the l-th layer. $\|$ is the column-wise concatenation. Z_u is low-dimensional latent factors for user exogenous variables given by Gaussian distribution $\mathcal{N}(0, \mathrm{I}_K)$. Similarly, for the item encoder, we calculate item v's hidden factors $\mathrm{h}_v^{(l)}$ by using f_{ϕ_2} with W_{ϕ_2} in Eq. (5.4).

[1] The neighbor message at the 0-th layer, i.e., $\mathrm{m}_{uv}^{(0)}$, is initialized from a normal distribution.
[2] $\mathrm{h}_v^{(0)}$ and $\mathrm{h}_u^{(0)}$ are initialized as node features d_v and d_u.

Having obtained the hidden factors $\mathrm{h}_u^{(l)}$ for user u and $\mathrm{h}_v^{(l)}$ for item v at each graph learning layer $l \in \{1, \cdots, L\}$, we adopt layer-aggregation to concatenate vectors at all layers into a single vector:

$$\mathrm{h}_u = \mathrm{h}_u^{(1)} + \cdots + \mathrm{h}_u^{(L)}, \quad \mathrm{h}_v = \mathrm{h}_v^{(1)} + \cdots + \mathrm{h}_v^{(L)} \tag{5.5}$$

By performing layer aggregation, we capture higher-order connectivities of node pairs across different graph learning layers. Finally, our semi-implicit generative model outputs h_u and h_v as hidden factors of users and items.

User and Item Encoder. Given hidden factors h_u for a user u, the user encoder outputs mean and variance in $\mathcal{N}\left(\mu_u, \mathrm{diag}\left(\sigma_u^2\right)\right)$, from which user embedding u is sampled:

$$q_{\theta_1}(\mathrm{u} \mid \mathrm{h}_u) = \mathcal{N}\left(\mathrm{u} \mid \mu_u, \mathrm{diag}\left(\sigma_u^2\right)\right) \tag{5.6}$$

where μ_u and $\mathrm{diag}\left(\sigma_u^2\right)$ are the mean and variance for user u, which are obtained by sending u's hidden factors h_u to a one-layer neural network with the activation function $\mathrm{ReLU}(x) = \max(0, x)$:

$$\mu_u = \mathrm{ReLU}\left(\mathrm{W}_{\theta_1}^{\mu_u} \mathrm{h}_u + \mathrm{b}\right), \sigma_u^2 = \exp\left(\mathrm{ReLU}\left(\mathrm{W}_{\theta_1}^{\sigma_u} \mathrm{h}_u + \mathrm{b}\right)\right) \tag{5.7}$$

where $\mathrm{W}_{\theta_1} = \{\mathrm{W}_{\theta_1}^{\mu_u}, \mathrm{W}_{\theta_1}^{\sigma_u}\}$ is a hidden-to-output weight matrix for the user encoder q_{θ_1}; b is the bias vector. Analogously, the item encoder follows the same paradigm as the user encoder to generate the mean and variance for item v based on v's hidden factors h_v:

$$\begin{gathered} q_{\theta_2}(\mathrm{v} \mid \mathrm{h}_v) = \mathcal{N}\left(\mathrm{v} \mid \mu_v, \mathrm{diag}\left(\sigma_v^2\right)\right), \\ \mu_v = \mathrm{ReLU}\left(\mathrm{W}_{\theta_2}^{\mu_v} \mathrm{h}_v + \mathrm{b}\right), \sigma_v^2 = \exp\left(\mathrm{ReLU}\left(\mathrm{W}_{\theta_2}^{\mu_v} \mathrm{h}_v + \mathrm{b}\right)\right) \end{gathered} \tag{5.8}$$

where $\mathrm{W}_{\theta_2} = \{\mathrm{W}_{\theta_2}^{\mu_v}, \mathrm{W}_{\theta_2}^{\sigma_v}\}$ is the weight matrix for the item encoder q_{θ_2}.

5.1.3.2 Collaborative Filtering Decoder

Collaborative filtering is largely developed based on latent factors. These models involve mapping users and items into latent factors in order to estimate the preference scores of users towards items. We use latent factor-based collaborative filtering in our decoder for modeling the user preference e, which is a probability vector over the entire item set for recommendations. The predicted user interaction vector y is assumed to be sampled from a multinomial distribution with probability e.

Formally, we define a generative function $f_{\theta_3}(\mathrm{u}, \mathrm{v})$ recovering classical latent factor-based CF to approximate user preference vector e:

$$\mathrm{e} = \mathrm{softmax}(f_{\theta_3}(\mathrm{u}, \mathrm{v})) = \mathrm{softmax}(\mathrm{u}^\top \mathrm{v}) \tag{5.9}$$

where u and v are latent factors for a user v and an item v drawn from Eq. (5.6) and Eq. (5.8), respectively. The softmax function transforms the calculated preference scores to probability vector e over the item corpus.

Then, the decoder $p_{\theta_3}(\mathrm{e} \mid \mathrm{u}, \mathrm{v})$ produces interaction probability y by approximating a logistic log-likelihood:

$$\log p_{\theta_3}(\mathrm{y} \mid \mathrm{e}) = \sum_{v} y_{uv} \log \sigma(\mathrm{e}) + (1 - y_{uv}) \log(1 - \sigma(\mathrm{e})) \tag{5.10}$$

where y_{uv} is the historical interaction between u and v, e.g., click. $\sigma(\mathrm{e}) = 1/(1 + \exp(-\mathrm{e}))$ is the logistic function.

5.1.3.3 Counterfactual Instances-based Optimization

We wish our NCGCF to be robust to unseen (unknown) user preference shifts to further enhance the recommendation robustness. Catching user preferences is at the core of any recommendation model; however, user preferences may change over time [215]. For example, a user may once love items with the brand *Nike* but change his taste for liking *Adidas*. Such a user preference shift can be captured by actively manipulating user preferences, i.e., manipulating e.

Since our NCGCF is a Neural Causal Model and is capable of generating "interventional" distributions within the Pearl Causal Hierarchy, the manipulations can be done by performing interventions [7] on the user preference vector e using a do-operator $do(\cdot)$, i.e., $do(\mathrm{e} = \mathrm{e}')$. The data after interventions are called *counterfactual instances* that, if augmented to original training instances, increase the model robustness to unseen interventions (i.e., user preference shifts). Inspired by [252], we optimize NCGCF by considering two data scenarios, i.e., the clean data scenario in which our NCGCF accesses the data without interventions, and the counterfactual data scenario in which the data is generated by known interventions on user preference vectors.

Formally, for the clean data scenario, assuming that NCGCF observes only clean data D during training. In this case, we retain the original value o of user preference e by using $do(\mathrm{e} = \mathrm{o})$. Then, NCGCF is trained by maximizing the likelihood function $\log p_{\theta_3}(\mathrm{y} \mid do(\mathrm{e} = \mathrm{o}))$. Since this marginal distribution is intractable [104], we instead maximize the intervention evidence lower-bound (ELBO) with $do(\mathrm{e} = \mathrm{o})$, i.e.

$\max_{\theta_1,\theta_2,\theta_3} \text{ELBO}(\text{D}, do(\text{e}=\text{o}))$. In particular,

$$
\begin{aligned}
&\text{ELBO}(\text{D}, do(\text{e}=\text{o})) = \\
&\mathbb{E}_\theta \left[\log \frac{p_{\theta_3}(\text{y} \mid do(\text{e}=\text{o}))p(\text{u})p(\text{v})}{q_{\theta_1}(\text{u} \mid \Xi, do(\text{e}=\text{o}))\, q_{\theta_2}(\text{v} \mid \Xi, do(\text{e}=\text{o}))} \right] \\
=&\mathbb{E}_\theta \left[\log p_{\theta_3}(\text{y} \mid do(\text{e}=\text{o})) \right] \\
&- \text{KL}\left(q_{\theta_1}(\text{u} \mid \Xi) \| p(\text{u}), q_{\theta_2}(\text{v} \mid \Xi) \| p(\text{v}) \right)
\end{aligned}
\tag{5.11}
$$

where Ξ represents required parameters for the conditional probability distributions of q_{θ_1}, q_{θ_2} and p_{θ_3}, i.e., $\Xi = \{\text{Z}_u, \text{d}_u, \text{A}_u\}$ for q_{θ_1}, $\Xi = \{\text{Z}_v, \text{d}_v, \text{A}_v\}$ for q_{θ_2} and $\Xi = \{\text{u}, \text{v}\}$ for p_{θ_3}. $\theta = \{\theta_1, \theta_2, \theta_3\}$ is a set of model parameters and $\text{KL}(\cdot)$ is KL-divergence between two distributions.

For the counterfactual data scenario, we assume NCGCF accesses counterfactual data D' generated by known interventions $do(\text{e} = \text{e}')$ on user preference vectors. The counterfactual vectors e' hold the same dimension with e and are drawn from a random distribution. Then, the ELBO of NCGCF with the counterfactual data is,

$$
\begin{aligned}
\text{ELBO}(\text{D}', do(\text{e}=\text{e}')) = &\mathbb{E}_\theta \left[\log p_{\theta_3} \left(\text{y} \mid do(\text{e}=\text{e}') \right) \right] \\
&- \text{KL}\left(q_{\theta_1}(\text{u} \mid \Xi) \| p(\text{u}), q_{\theta_2}(\text{v} \mid \Xi) \| p(\text{v}) \right)
\end{aligned}
\tag{5.12}
$$

Inspired by data augmentation and adversarial training, we augment the clean data with counterfactual instances to enhance the robustness of our NCGCF meanwhile capturing user preference shifts. In particular, the total loss function after augmentation is as below,

$$
\begin{aligned}
\mathscr{L}_{\text{aug}}(\Theta) = &\lambda(\text{ELBO}(\text{D}, do(\text{e}=\text{o})) \\
&+ (1-\lambda)(\text{ELBO}(\text{D}', do(\text{e}=\text{e}'))
\end{aligned}
\tag{5.13}
$$

where $\mathscr{L}_{\text{aug}}(\Theta)$ is the loss function for training our NCGCF and Θ are model parameters. λ is the trade-off parameter between the clean and the counterfactual data scenario. During the training stage, the loss function is calculated by averaging the ELBO over all users.

5.1.3.4 Computational Complexity Analysis

Three major components determine the complexity of our proposed NCGCF: 1) Causal Graph Encoder; 2) Collaborative Filtering (CF) Decoder and 3) Counterfactual Instances-Based Optimization. The causal graph encoder consists of a user encoder and an item encoder, which gives the complexity of $O(L(Ed + Nd^2))$ from L-GNN layers, given N as

the total number of users and items, E as the number of edges, and d as the embedding size. The semi-implicit general model in the causal graph encoder gives the complexity of $O(L(Ed+Nd^2)+Nd^2)$. The CF Decoder estimates user preference using collaborative filtering based on matrix factorization. Given M users and K items, the operation of the dot product for the prediction of preferences gives a complexity of $O(MKd)$. The complexity of counterfactual optimization is $O(NCd)$, given C counterfactual samples per training instance. In total, the computational complexity of NCGCF is $O(L(Ed+Nd^2)+Nd^2+MKd+NCd)$.

5.1.4 Experiments

We thoroughly evaluate the proposed NCGCF for the recommendation task to answer the following research questions:

- **RQ1.** How does NCGCF perform as compared with state-of-the-art recommendation methods?
- **RQ2.** How do different components impact NCGCF's performance?
- **RQ3.** How do parameters in the causal graph encoder affect NCGCF?

5.1.4.1 Setup

Datasets. We conduct experiments on one synthetic dataset and three real-world datasets to evaluate NCGCF. The synthetic dataset is constructed in accordance with the causal graph depicted in Figure 5.1(c). The construction process follows a series of assumptions that reflect causal relations between users and items. For instance, we assume the causal relation between user features and user preferences based on prior knowledge, such as the positive effect of high income on preference over high price. Similar assumptions also apply to item features to user preferences, e.g., the positive effect of the brand "Apple" on the preference for high-priced items. In particular, the **Synthetic** dataset construction is under the following four steps:

1. Feature generation: We simulate $|U| = 1,000$ users and $|I| = 1,000$ items, where each user has one discrete feature (*gender*) and one continuous feature (*income*), while each item has three discrete features, i.e., *type*, *brand* and *price*. For discrete features, their values in $\{0,1\}$ are sampled from Bernoulli distributions. We sample continuous features from random sampling, in which random feature values are

chosen from the minimum (i.e., 0) and the maximum (i.e., 1000) feature values. For both users and items, we assume two exogenous variables (i.e., Z_u and Z_v) drawn from the Gaussian distribution.

2. Causal neighbor sampling: We synthesize the causal relations $U \rightarrow U$ and $V \rightarrow V$ by creating user/item causal neighbors. In particular, we set the causal neighbor number $N_c = 10$. We assume a user u's causal neighbors (i.e., $U \rightarrow U$) are those who have interacted with the same item with the user u. In other words, users who have shown interest in similar items are considered causal neighbors for each other. For item causal neighbor sampling (i.e., $V \rightarrow V$), we first convert items with their features into dense vectors through item2vec, then calculate the Euclidean distances between two items. We assume those items that have the N_c smallest distances from the target item are causal neighbors for the target item.

3. User preference estimation: For each user u and item v, the user preference $\mathrm{u} \in \mathbb{R}^d$ towards item property $\mathrm{v} \in \mathbb{R}^d$ is generated from a multi-variable Gaussian distribution $\mathcal{N}(0, \mathrm{I})$. Then, the preference score y_{uv} between user u and item v is calculated by the inner product of u and v. Besides, we assume the fine-grained causal relations from user/item features to the preference score based on prior knowledge. For example, we assume a positive effect of the "high" *income* on the preference over "high" *price*, thus tuning the preference score to prefer items with high prices. Besides, a user should have similar preference scores toward an item and the item's causal neighbors.

4. User interaction sampling: Once we obtain a user u's preference scores for all items (i.e., I), we normalize preference scores by $\frac{\exp(r_i)}{\sum_{i' \in I} \exp(r_{i'})}$. We select items with k-top scores as the interactions for the user $u \in U$, where k is a constant chosen randomly from range $[20, 100]$.

Apart from the synthetic dataset, we also use three benchmark datasets to test our performance in real-world scenarios. We also assume fine-grained causal relations in these real-world datasets to ensure users interact with items causally.

- **Amazon-Beauty** and **Amazon-Appliances**: two sub-datasets from Amazon Product Reviews [3] [78], which record large crawls of user reviews and product metadata (e.g., *brand*). Following [76], we use *brand* and *price* to build item features since

[3] https://nijianmo.github.io/amazon/index.html

other features (e.g., *category*) are too sparse and contain noisy information. We use co-purchased information from the product metadata to build item-item causal relations, i.e., $V \rightarrow V$. The co-purchased information records item-to-item relationships, i.e., a user who bought item v also bought item i. We assume an item's causal neighbors are those items that are co-purchased together. For user-user causal relation (i.e., $U \rightarrow U$), we assume a user's causal neighbors are those who have similar interactions, i.e., users who reviewed the same item are neighbors for each other.

- **Epinions** [4] [190]: a social dataset recording social relations between users. We convert user/item features from the dataset into one-hot embeddings. We use social relations to build user causal neighbors, i.e., a user's social friends are the neighbors of the user. Besides, items bought by the same user are causal neighbors to each other.

For the three real-world datasets, we regard user interactions with overall ratings above 3.0 as positive interactions. For the synthetic dataset, we regard all user-item interactions as positive as they are top items selected based on users' preferences. The statistics of the four datasets are shown in Table 6.8. For model training, we randomly split samples in both datasets into training, validation, and test sets by the ratio of 70%, 10%, and 20%.

Table 5.1: Statistics of the datasets.

Dataset	Synthetic	Amazon-Beauty	Amazon-Appliances	Epinions
# Users	1,000	271,036	446,774	116,260
# Items	1,000	29,735	27,888	41,269
# Interactions	12,813	311,791	522,416	181,394
# Density	0.0128	0.0039	0.0041	0.0038

Baselines. We compare NCGCF with eight competitive recommendation methods.

- **BPR** [161]: a well-known matrix factorization-based model with a pairwise ranking loss to enable recommendation learning from implicit feedback.

- **NCF** [81]: extends CF to neural network architecture. It maps users and items into dense vectors and feeds user and item vectors into an MLP to predict user preferences.

[4] http://www.cse.msu.edu/ tangjili/trust.html

- **MultiVAE** [121]: extends CF to variational autoencoder (VAE) structure for implicit feedback modeling. It formulates CF learning as a generative model and uses variational inference to model the posterior distributions.
- **NGCF** [205]: a GCF that incorporates two GCNs to learn user and item embeddings. The learned embeddings are passed to a matrix factorization to capture the collaborative signal for recommendations.
- **VGAE** [104]: a graph learning method that extends VAE to handle graph-structured data. We use VGAE to obtain user and item embeddings and inner product those embeddings to predict user preference scores.
- **GC-MC** [10]: a graph-based auto-encoder framework for matrix completion. The encoder is a GCN that produces user and item embeddings. The learned embeddings reconstruct the rating links through a bilinear decoder.
- **LightGCN** [80]: a SOTA graph-based recommendation model that simplifies the GCN component. It includes the essential part in GCNs, i.e., neighbor aggregation, to learn user and item embeddings for collaborative filtering.
- **CACF** [253]: a method that learns attention scores from individual treatment effect estimation. The attention scores are used as user and item weights to enhance the CF.

Evaluation Metrics. We use three Top-K recommendation evaluation metrics, i.e., Precision@K, Recall@K and Normalized Discounted Cumulative Gain(NDCG)@K. The three evaluation metrics measure whether the recommended Top-K items are consistent with users' preferences in their historical interactions. We report the average results with respect to the metrics over all users. The Wilcoxon signed-rank test is used to evaluate whether the improvements against baselines are significant.

Implementation Details. We implement our NCGCF using Pytorch. The code and datasets are released in `https://github.com/Chrystalii/CNGCF`. The latent embedding sizes of neural networks for all neural-based methods are fixed as $d = 64$. The in-dimension and out-dimension of the graph convolutional layer in NCGCF, NGCF, VGAE, GC-MC and LightGCN are set as 32 and 64, respectively. We apply a dropout layer on top of the graph convolutional layer to prevent model overfitting for all GCN-based methods. The hyper-parameters of all methods are chosen by the grid search, including the learning rate l_r in $\{0.0001, 0.0005, 0.001, 0.005\}$, L_2 norm in $\{10^{-5}, 10^{-4}, \cdots, 10^{1}, 10^{2}\}$,

and the dropout ratio p in $\{0.0, 0.1, \cdots, 0.8\}$. The Adam optimizer is applied to all methods for model optimization, where the batch size is fixed as 1024. We set the maximum epoch for all methods as 400 and use the early stopping strategy, i.e., terminate model training when the validation Precision@10 value does not increase for 20 epochs. To ensure a fair comparison, all baseline methods are trained using the same data used in our NCGCF. This includes using causality-enhanced node features and causal relations, such as item-item and user-user relationships, in the training process for all models.

5.1.4.2 Recommendation Performance (RQ1)

Table 5.2: Recommendation performance comparison.

Dataset	Synthetic			Amazon-Beauty			Amazon-Appliances			Epinions		
Method	Precision@10	Recall@10	NDCG@10	Precision@10	Recall@10	NDCG@10	Precision@10	Recall@10	NDCG@10	Precision@10	Recall@10	NDCG@10
BPR	0.5214	0.4913	0.6446	0.3555	0.3319	0.4111	0.3720	0.3574	0.4356	0.3022	0.2895	0.4889
NCF	0.6120	0.6293	0.7124	0.3618	0.3659	0.4459	0.3871	0.3789	0.4771	0.3551	0.3364	0.5432
MultiVAE	0.6248	0.5999	0.8101	0.4418	0.4112	0.4616	0.4544	0.4428	0.5998	0.4229	0.3888	0.5331
NGCF	0.5990	0.5681	0.7477	0.4512	0.4003	0.5188	0.4271	0.3778	0.5555	0.4018	0.3912	0.5012
VGAE	0.5446	0.5572	0.7778	0.3499	0.3812	0.4466	0.3681	0.4014	0.5019	0.3590	0.3460	0.4913
GC-MC	0.6115	0.6226	0.8116	0.4666	0.4615	0.5612	0.4718	0.4518	0.5677	0.4666	0.4218	0.5112
LightGCN	0.6439	0.6719	0.8223	0.4810	0.4778	0.5501	0.4844	0.4652	0.6028	0.4717	0.4544	0.5436
CACF	0.4482	0.4158	0.5555	0.3101	0.3005	0.3888	0.3222	0.3188	0.4215	0.2899	0.2765	0.3445
NCGCF	**0.7952**	**0.6889**	**0.8538**	**0.5148**	**0.5183**	**0.6855**	**0.6510**	**0.5271**	**0.8193**	**0.4990**	**0.5030**	**0.5589**
Improv.%	+23.4%	+2.5%	+3.8%	+7.0%	+8.4%	+22.1%	+34.3%	+13.3%	+35.9%	+5.7%	+10.6%	+2.8%
	Precision@20	Recall@20	NDCG@20	Precision@20	Recall@20	NDCG@20	Precision@20	Recall@20	NDCG@20	Precision@20	Recall@20	NDCG@20
BPR	0.6111	0.5536	0.6338	0.3561	0.3420	0.4062	0.3941	0.3599	0.4322	0.3332	0.3232	0.4689
NCF	0.6678	0.6446	0.7003	0.3699	0.3691	0.4330	0.3999	0.4033	0.4519	0.3719	0.3614	0.5255
MultiVAE	0.6779	0.6136	0.8006	0.4496	0.4200	0.4555	0.4819	0.4716	0.5911	0.4465	0.4055	0.5133
NGCF	0.6233	0.5999	0.7312	0.4612	0.4112	0.5081	0.4666	0.4258	0.5499	0.4223	0.4210	0.4811
VGAE	0.5847	0.5687	0.7613	0.3551	0.3999	0.4410	0.3771	0.4228	0.4761	0.3667	0.3598	0.4781
GC-MC	0.6665	0.6317	0.8091	0.4781	0.4771	0.5582	0.4892	0.4881	0.5514	0.4815	0.4451	0.4999
LightGCN	0.6904	0.6819	0.8108	0.5023	0.4869	0.5306	0.4919	0.4781	0.5613	0.4915	0.4718	0.5221
CACF	0.4567	0.4266	0.5348	0.3186	0.3211	0.3678	0.3418	0.3271	0.4103	0.2747	0.2910	0.3368
NCGCF	**0.8081**	**0.6844**	**0.8603**	**0.5153**	**0.5106**	**0.7123**	**0.6367**	**0.5055**	**0.8501**	**0.5002**	**0.5034**	**0.5667**
Improv.%	+17.0%	+0.3%	+6.1%	+2.5%	+4.8%	+27.6%	+29.4%	+3.5%	+43.8%	+1.7%	+6.6%	+7.8%

We show the recommendation performance of our NCGCF and all baselines on the four datasets in Table 6.9. By analyzing Table 6.9, we have the following findings.

- NCGCF consistently outperforms the strongest baselines on both synthetic and real-world datasets, achieving the best recommendation performance across all three evaluation metrics. In particular, NCGCF outperforms the strongest baselines by 23.4%, 7.0%, 34.3% and 5.7% w.r.t Precision@10 on Synthetic, Amazon-Beauty, Amazon-Appliances and Epinions, respectively. Additionally, NCGCF improves 2.5%/3.8%, 8.4%/22.1%, 13.3%/35.9% and 10.6%/2.8% for Recall@10/NDCG@10 on the four datasets, respectively. The superiority of NCGCF can be attributed to two factors: the power of neural graph learning and the modeling of causality. Firstly, graph learning explicitly models the interactions between users and items as a graph, and uses graph convolutional networks to capture the non-linear relations

from neighboring nodes. This allows graph learning to capture more complex user behavior patterns. Secondly, modeling causal relations allows us to identify the causal effects of different items on users, thus capturing true user preferences on items. By injecting causal modeling into graph representation learning, our NCGCF captures more precise user preferences to produce robust recommendations against baselines.

- NCGCF achieves the most notable improvements (e.g., 35.9% NDCG@10 and 43.8% NDCG@20) on the Amazon-Appliances dataset. Amazon-Appliances dataset is a large-scale dataset with a considerable amount of user behavior data that may be noisy and challenging to model. Nevertheless, NCGCF still outperforms all baselines. We consider the reason is that NCGCF injects causality into graph learning, enabling the model to surpass merely capturing spurious correlations among noisy data. This results in more accurate and reliable modeling of true user preferences.

- NGCF that uses graph representation learning outperforms NCF without graph learning. This is because NGCF models user-item interactions as a graph, and uses graph convolutional networks to capture more complex user-user collaborative behavior to enhance recommendations. In contrast, NCF uses a multi-layer perception to learn user and item similarities, which captures only linear user-item correlations from the interaction matrix. Moreover, GC-MC and LightGCN outperform other graph learning-based baselines (i.e., NGCF, VGAE) in most cases. This is because GC-MC and LightGCN aggregate multiple embedding propagation layers to capture higher-order connectivity within the interaction graph. Similarly, our NCGCF incorporates layer aggregation within our causal graph encoder, enabling us to capture higher-order connectivity and produce better graph representations for improved recommendation performance.

- NCGCF outperforms all graph learning-based baselines, including NGCF, VGAE, GC-MC and LightGCN. This is because NCGCF models causal relations within the graph learning process. Guided by the causal recommendation generation process, NCGCF is able to inject causal relations under the Structural Causal Model into the learning process of the graph convolutional network. This allows NCGCF to uncover the causal effect of items on users and capture user behavior patterns more accurately.

5.1.4.3 Study of NCGCF (RQ2)

Table 5.3: Recommendation performance after replacing the causal graph encoder with different graph representation learning methods.

Variants	Precision@10	Recall@10	NDCG@10
		Synthetic	
NCGCF	0.7952	0.6889	0.8538
NCGCF-GCN	0.5358(-32.7%)	0.5182(-24.7%)	0.7025(-17.7%)
NCGCF-Graphsage	0.5038(-36.8%)	0.5005(-27.4%)	0.7022(17.8%)
NCGCF-Pinsage	0.5819(-26.8%)	0.5498(-20.2%)	0.7446(-12.8%)
		Amazon-Beauty	
NCGCF	0.5148	0.5183	0.6855
NCGCF-GCN	0.4991(-3.04%)	0.5029(-2.97%)	0.4886(-28.68%)
NCGCF-Graphsage	0.5011(-2.67%)	0.5039(-2.78%)	0.5243(-23.55%)
NCGCF-Pinsage	0.5008(-2.72%)	0.5043(-2.70%)	0.5143(-25.01%)
		Amazon-Appliances	
NCGCF	0.6510	0.5271	0.8193
NCGCF-GCN	0.5067(-3.04%)	0.5167(-2.97%)	0.6614(-28.68%)
NCGCF-Graphsage	0.5085(-2.67%)	0.5184(2.78%)	0.6670(- 23.55%)
NCGCF-Pinsage	0.5083(-2.72%)	0.5178(-2.70%)	0.6631(-25.01%)
		Epinions	
NCGCF	0.4990	0.5030	0.5589
NCGCF-GCN	0.4812(-3.55%)	0.4990(-0.79%)	0.5013(-10.28%)
NCGCF-Graphsage	0.4809(-3.62%)	0.4989(-0.81%)	0.4999(-10.52%)
NCGCF-Pinsage	0.4871(-2.38%)	0.4994(-0.71%)	0.4930(-11.74%)

We start by exploring how replacing our causal graph encoder with other graph representation learning methods, i.e., naive GCN [103], Graphsage [74] and Pinsage [244], impact NCGCF's performance. We then analyze the influences of core components, including causality-aware message passing and counterfactual instance-aware ELBO.

Effect of Causal Graph Encoder. The causal graph encoder plays a pivotal role in NCGCF by modeling the causal relations of nodes. To investigate its effectiveness, we replace our causal graph encoder with different encoders built by other graph learning methods. In particular, we use GCN [103], Graphsage [74] and Pinsage [244] to produce user and item embedding vectors for the decoder learning phase, and compare the performance of NCGCF before and after the replacements. We present the experimental results in Table 5.3. We find that both GCN, Graphsage and Pinsage-based encoders downgrade the performance of NCGCF compared to NCGCF equipped with our proposed causal graph encoder. For instance, NCGCF with a GCN-based encoder downgrades the NDCG@10 by 28.68% on the Amazon-Beauty. This is because GCN, Graphsage and Pinsage cannot capture the causal relations of nodes in the interaction graph, leading to

insufficient representations of users and items. On the contrary, our causal graph encoder captures the intrinsic causal relations between nodes using the causality-aware message passing; thus, it learns causality-aware user and item representations to better serve the later decoder learning. Moreover, the GCN-based encoder downgrades the NCGCF performance most severely compared with GraphSage and Pinsage-based encoders. This is because naive GCN performs transductive learning requiring full graph Laplacian, whereas GraphSage and Pinsage perform inductive learning without requiring full graph Laplacian to handle large-scale graph data well. We thus conclude that an inductive learning setting is more desired for our NCGCF, especially when facing large-scale graph data.

Table 5.4: Ablation study on NCGCF.

Variants	Precision@10	Recall@10	NDCG@10
		Synthetic	
NCGCF	0.7952	0.6889	0.8538
¬ CM	0.5806(−31.9%)	0.5491(−20.3%)	0.7179(−16.0%)
¬ CI	0.7781(−2.1%)	0.6654(−3.4%)	0.7573(−11.2%)
		Amazon-Beauty	
NCGCF	0.5148	0.5183	0.6855
¬ CM	0.5007(−2.7%)	0.5060(−2.3%)	0.5383(−20.7%)
¬ CI	0.5101(−0.9%)	0.5081(−2.0%)	0.5738(−15.9%)
		Amazon-Appliances	
NCGCF	0.6510	0.5271	0.8193
¬ CM	0.6357(−2.4%)	0.5050(−4.2%)	0.6864(−16.2%)
¬ CI	0.6445(−1.0%)	0.5143(−2.4%)	0.7956(−2.9%)
		Epinions	
NCGCF	0.4990	0.5030	0.5589
¬ CM	0.4695(−6.0%)	0.4936(−1.9%)	0.4647(−16.9%)
¬ CI	0.4794(−3.9%)	0.5018(−0.2%)	0.5139(−8.1%)

Effect of Causality-aware Message Passing. The causality-aware message passing models the dependency terms between each of the structural equations as the causal relations between nodes. We present NCGCF's performance after removing the causality-aware message passing in Table 6.11. We observe that removing the component downgrades NCGCF's performance, indicating the importance of causality-aware message passing in helping NCGCF achieve favorable recommendation performance. We thus conclude that modeling the causal relations between nodes within the graph-structured data is essential for graph learning-based models to uncover true user preferences for improved recommendations.

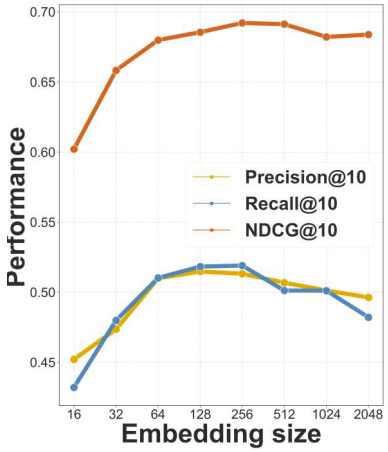
Effect of Counterfactual Instance-aware ELBO. The counterfactual instance-aware ELBO augments counterfactual instances for NCGCF optimization. We present NCGCF's performance after removing the counterfactual instance-aware ELBO in Table 6.11. Apparently, removing the counterfactual instance-aware ELBO leads to the downgraded performance of NCGCF on both datasets. This is because our counterfactual instance-aware ELBO augments counterfactual instances, i.e., the intervened data on user preference vectors, thus facilitating better model optimization to capture user preference shifts.

5.1.4.4 Parameter Analysis of Causal Graph Encoder (RQ3)

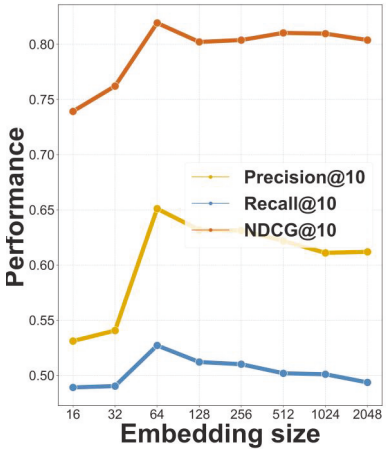
We analyze NCGCF's performance under different embedding sizes n of the semi-implicit generative model in the causal graph encoder. We also investigate the node dropout ratios p of the dropout layer applied in the causal graph encoder.

Effect of Embedding Size. Figure 5.3 (a) (b) (c) report the parameter sensitivity of our NCGCF w.r.t. embedding size n with $n = \{16, 32, 64, 128, 256, 512, 1024, 2048\}$. Apparently, the performance of NCGCF on Amazon-Beauty, Amazon-Appliances and Epinions demonstrates increasing trends from $n = 16$, then reaches the peak when $n = 512$, $n = 64$ and $n = 256$, respectively. This is reasonable since n controls the number of latent vectors of users and items from the semi-implicit generative model, and low-dimensional latent vectors cannot retain enough information for the encoder learning phrase. After reaching the peaks, the performance of NCGCF degrades slightly and then becomes stable. The decrease in performance is due to the introduction of redundant information as the embedding size becomes too large, which can affect the model. Additionally, we observe the largest Amazon-Appliances dataset requires the smallest embedding size of $n = 64$ to reach its peak performance compared to the other two datasets. This is because a larger embedding size brings large-scale datasets a higher computational burden, thus limiting the model's performance.

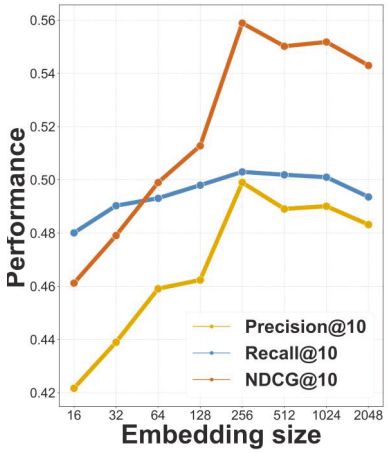
Effect of Dropout Ratio. We employ a node dropout layer in the causal graph encoder to prevent model overfitting. We show the influence of node dropout ratio p on the three datasets in Figure 5.3 (d) (e) (f). We observe that the performance of NCGCF on both Amazon-Beauty, Amazon-Appliances and Epinions exhibits a decreasing trend as we increase the node dropout ratio p from 0.0 to 0.3, but recovers at $p = 0.4$. After $p = 0.4$, the performance of NCGCF decreases as the dropout ratio increases. We believe that the reduced performance could be attributed to the removal of crucial information that the model needs to learn from the data, thus impairing the NCGCF's performance. Nevertheless, the recovered performance at $p = 0.4$ indicates that NCGCF is robust to



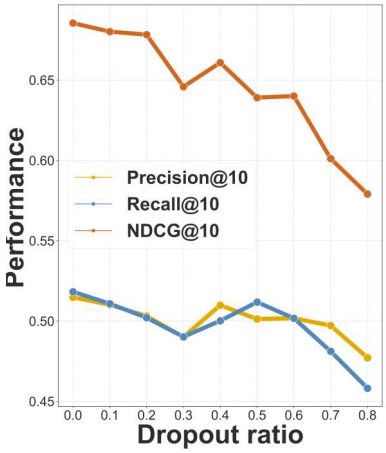
(a) Impact of embedding size on Amazon-Beauty.



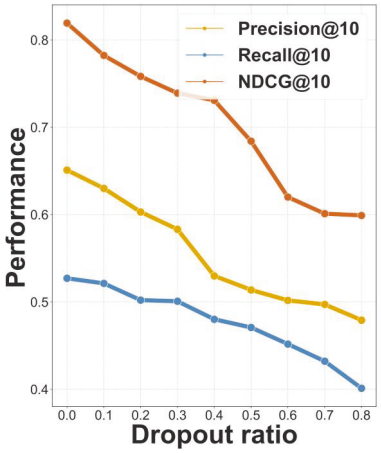
(b) Impact of embedding size on Amazon-Appliances.



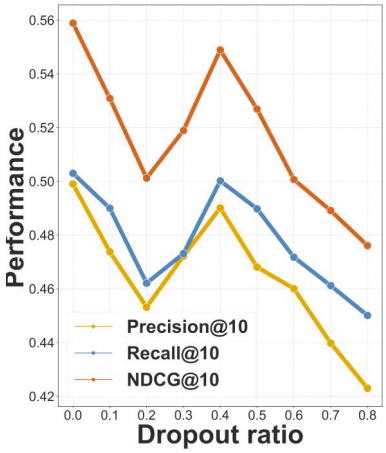
(c) Impact of embedding size on Epinions.



(d) Impact of dropout ratio on Amazon-Beauty.



(e) Impact of dropout ratio on Amazon-Appliances.



(f) Impact of dropout ratio on Epinions.

Figure 5.3: Parameter analysis on causal graph encoder.

balance the loss of information and overfitting.

5.1.5 Summary

In this work, we propose Neural causal graph Collaborative Filtering (NCGCF), the first causality-aware graph representation learning framework for graph collaborative filtering. In particular, NCGCF injects causal relations between nodes into the graph representation learning process. The integration of causal modeling and graph representation learning offers a novel perspective to facilitate accurate recommendations. In the proposed method, we first craft a causal graph to describe the learning process of causality-aware graph representations. Our causal graph abandons the strong assumption of user and item independence in current recommendation models. We then construct a Neural Causal Model to parameterize each of the structural equations under the causal graph as trainable neural networks. The proposed Neural Causal Model completes the first Neural-Causal Connection for the causal modeling of graph convolutional networks in recommendations. Finally, we approximate the Neural Causal Model using variational inference, with a semi-implicit generative model enabling causality-aware message passing for graph learning. As a result, NCGCF effectively models complex node and variable dependencies under structural equations. Extensive evaluations of recommendation performance highlight NCGCF's ability to produce precise recommendations.

5.2 Counterfactual Policy Optimization

5.2.1 Overview

Research Question. Reinforcement learning (RL)-based approaches have attracted a lot of attention in recommender systems, in which an agent (recommender) is guided by a recommendation algorithm (policy) to drive user interaction with the environment [112, 126]. The core idea of RL methods is to train RS as an intelligent agent that learns an optimal recommendation policy to maximize each user's long-term satisfaction with its system [31]. To train such an optimal RL agent, it is natural to perform online learning on interactions between recommender systems and users. However, such online learning is infeasible in real RS since it might degrade user satisfaction and deteriorate the revenue of the platform [30, 63]. Fortunately, off-policy learning emerges as a favorable opportunity for policy optimization, which uses historical user feedback instead of constructing expensive online interactive environments [94, 146, 233]. Therefore, this

research aims to investigate the off-policy learning methods for recommender systems so that dynamic user interactions can be modeled to improve recommendation performance without using expensive and risky online learning.

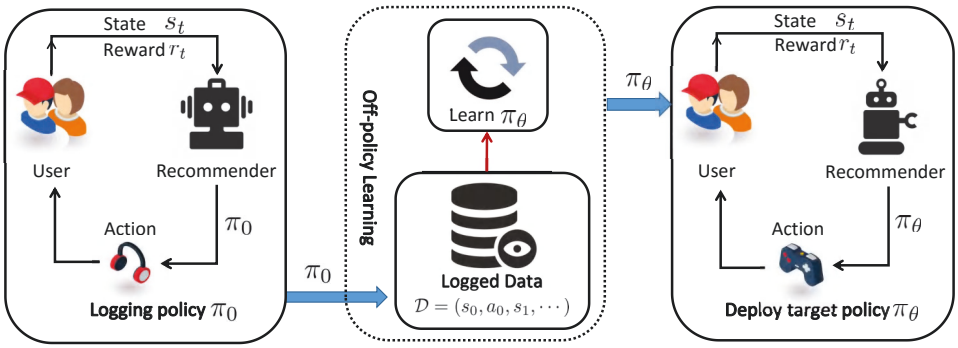


Figure 5.4: Off-policy learning in recommendation.

Research Objective. As shown in Figure 5.4, the goal of off-policy learning is to maximize each user's long-term satisfaction with the system, given the logged data generated by a logging policy. Achieving this off-policy goal has to address the question of "*how much reward would be received if a new target policy had been deployed instead of the original logging policy*?". This counterfactual question is not easy to address since the target policy is different from the historical logging policy in the off-policy setting [188]. To this effect, most off-policy learning for recommendation relies on inverse propensity score (IPS) estimator correction to get an unbiased empirical risk minimization objective [32, 133]. A major disadvantage of these methods is that IPS is likely to be over-fitted as some actions have zero probability of being taken in recommender systems [95]. For example, compared with a large action space (e.g., items) in a recommender system, actions taken by users are limited in a deficiency action space due to the ubiquity of biases [25] (e.g., exposure bias or conformity bias). Thus, a large number of actions would not be selected at all in recommender systems, leading to the "poor gets poorer" phenomenon [48], i.e.., a video will not be nominated in the target policy simply because it was never nominated in the behavior logging policy. Therefore, the research objective of this research is to develop a learning method that alleviates the "poor gets poorer" issue brought by the IPS-based estimators.

Motivations. This research is motivated by the following intuition: real-world context information could be useful in dealing with deficient log data and empowering off-policy learning in the recommendation. An example is shown in Figure 5.5, we have a user u_1 exposed to (i_1, i_2) but not to (i_3, i_4). As a result, it is impossible to learn cumulative

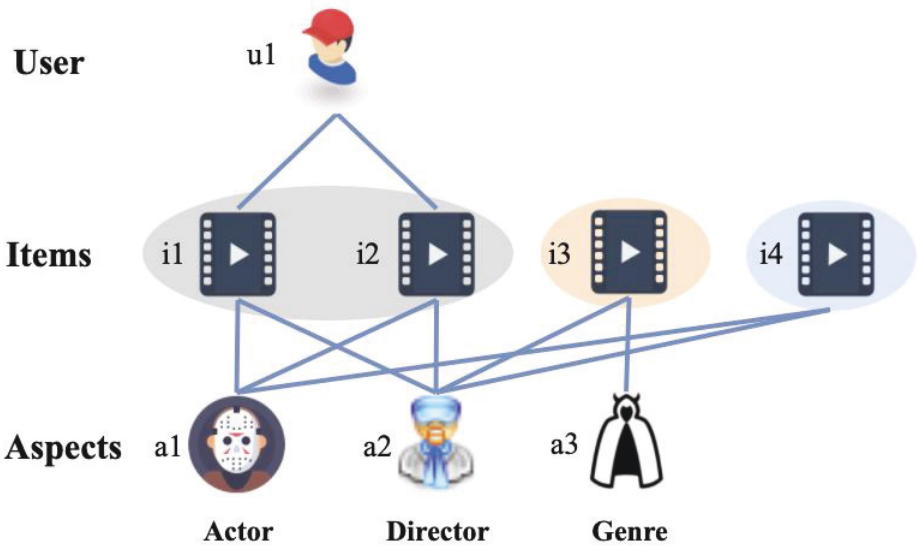


Figure 5.5: A toy example of inferring rewards from HIN.

rewards (e.g., users' feedback during a period of time) for a policy that selects actions (e.g., i_3 and i_4) not contained in the logged data. That means information about the reward for the action of interacting with i_3 and i_4 can never be chosen by the deterministic logging policy. Fortunately, we can infer such feedback information with the assistance of contextual information. Suppose u_1 offers positive feedback to i_1 and i_2. Since both i_1 and i_2 have the same actor a_1 and the same director a_2, we may infer that the combination of actor a_1 and director a_2 is an important factor of u_1's interest. The movie i_4 with the same actor a_1 and director a_2 should be highly likely to be preferred by u_1. By contrast, u_1 probably has less interest in i_3 with a different actor. That means i_3 and i_4 could offer high-quality negative feedback and positive feedback of i_4. As such, exploiting the contextual information can alleviate the "poor gets poorer" phenomenon in off-policy learning for the recommendation.

The Proposed Approach. This research proposes to correct off-policy biases with the assistance of a Heterogeneous Information Network, i.e., HIN augmented off-policy learning (HINpolicy). In particular, we design a co-attentive mechanism to mutually derive the interaction-specific context information to produce the high-quality target policy of the recommendation. Meanwhile, counterfactual risk minimization is designed to explore the target policy so as to optimize the behavior policy that can maximize users' long-term satisfaction.

Contributions. The proposed method offers the following contributions:

- We are the first to leverage contextual information in HIN to provide high-quality target policy learning for correcting the bias in off-policy recommendations.
- We develop a new end-to-end framework HINpolicy, which achieves counterfactual risk minimization in an explicit manner under the co-attention mechanism.
- Empirically, we generate an online environment using simulators to carry out experiments on two benchmark datasets. Extensive results show that our methods outperform the state-of-the-art methods.

5.2.2 Off-policy Learning for Recommendation

Unlike classical reinforcement learning, off-policy learning does not have real-time interactions with recommender systems due to learning and infrastructure constraints. Instead, in the off-policy learning setting for the recommendation, we have access to a logged dataset of trajectories $\mathscr{D}$. The generation of $\mathscr{D}$ can be formulated with a Markov Decision Process (MDP) as denoted in Figure 5.4, where

- $\mathscr{S}$: a continuous state space describing the user states, e.g., user's contextual information involved during interactions;
- $\mathscr{A}$: a discrete action space containing items available for recommendation;
- $\mathscr{P}$: the state transition probability;
- $\mathscr{R}$: $r(s,a) \in \mathscr{R}$ is the immediate reward produced by taking the action a to the user state s;
- γ: a discount factor $\gamma \in [0,1]$ used for future immediate rewards;

Particularly, $\mathscr{D} = \{\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma\}$ has been collected under stochastic logging policy $\pi_0(a|s)$ that describes a probability distribution over items $\mathscr{A}$ (i.e., action), conditioned on user states $\mathscr{S}$. Meanwhile, the recommender system receives feedback reward $r(a,s)$ (i.e., clicks or watch time) for this particular state-action pair. As a result, training a recommender system seeks a policy π_θ that maximizes the expected cumulative rewards $R(\pi_\theta)$ over potentially infinite time horizon T, is defined as,

$$R(\pi_\theta) = \mathbb{E}_{s_0 \sim \rho(s), a_t \sim \pi_\theta(a|s_t), s_{t+1} \sim P(s|s_t,a_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{5.14}$$

where $\rho(s)$ is the initial distribution of user states, $P(s \mid s_t, a_t) \in \mathscr{P}$ is the state transition probability.

5.2.3 Methodology

5.2.3.1 The HINpolicy Framework

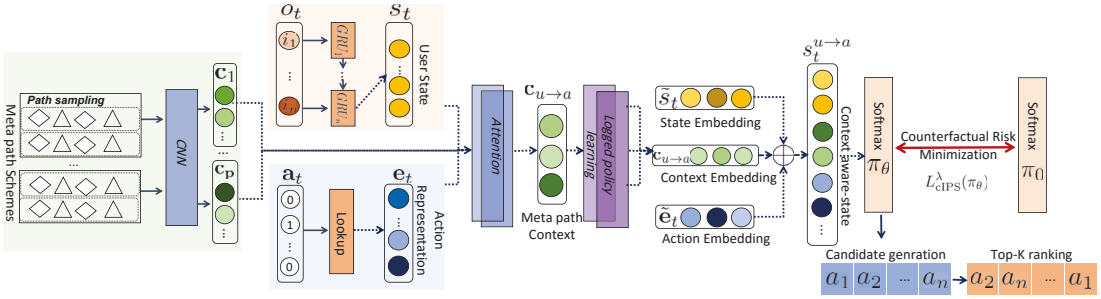


Figure 5.6: Our model framework of HINpolicy. Our HINpolicy includes rich contextual information retained in meta-path schemes for policy learning, thus capturing the potential influence of user/item attributes.

The overall framework of HINpolicy is presented in Figure 5.6, which consists of two important components: HIN-augmented policy learning and counterfactual risk minimization. In HIN-augmented policy learning, we aim at leveraging complex relations in meta-paths to learn the meta-path context for the involved users and actions, then use the designed co-attention mechanism to derive the context-aware state that guides a better policy learning for the recommendation. We further take advantage of counterfactual risk minimization for the unbiased approximation of policy evaluation. Bias is ubiquitous in the off-policy learning setting [63], since the logged feedback data generated by a historical behaviour policy π_0 of the recommender system is different from the target policy π_θ trained. To correct the bias of distribution mismatch between behavior policy and target policy, counterfactual risk minimization (CRM) [188] uses IPS estimator to re-weight the logged data according to ratios of slate probabilities under the target and logging policy. The final corrected recommendation policy π_θ is then used to produce candidate actions that wait to be re-ranked by the Top-K ranking model.

HIN-augmented policy learning. While off-policy learning methods [133] achieve great success towards policy optimization in recommendations, the benefits of contextual information are not fully explored to improve policy learning. A heterogeneous information network (HIN), whose nodes are of different types and links among nodes represent different relations, reveals high-order dependency in recommendation environments (e.g., users' behaviors, recommendation policies, and action aspects). Towards this, we take HIN as the prior knowledge of the policy learning to exploit its rich relations for exploring more suitable positive feedback that is missing in the logged data. We propose

HIN-augmented policy learning that learns context-aware state by updating user state representation, meta-path context and action representation in a mutual enhancement way, so as to guide a better target policy learning. In particular,

- **State Representation.** To derive the impact of meta-path based context towards the preference shift of users on taking actions, we first categorize the user state s_t with key information about the user preference. We design the state representation module, which extracts the user's preferences through its historical interactions with items. Generally, the state representation s_t at time t in an online recommendation scenario is learned from the user's interactions (e,g., clicked) at timestep t. Formally, for state s_t, we have a set of actions $o_t = \{i_1, i_2, ..., i_n\}$ interacted by the user. Considering $\{o_t\}$ have sequential patterns, we resort Recurrent Neural Networks (RNN) to learn an embedding vector $\mathrm{o}_t \in \mathbb{R}^d$ from $\{o_t\}$. To aggregate user's historical embedding o_t, we conducted experiments with a large volume of popular RNN cells, including Bidirectional Recurrent Neural Networks (BRNN) [172], Gated Recurrent Units (GRU) [35], and Long Short-Term Memory family (LSTM) [65] with varying gates. Finally, the RNN with a gated recurrent unit (GRU) stands out among these cells due to its stability and computational efficiency. Hence, we learn the representation of the user state s_t by a GRU cell:

$$
\begin{aligned}
\mathrm{z}_t &= \sigma_g(\mathrm{W}_1 \mathrm{o}_t + \mathrm{U}_1 s_{t-1} + \mathrm{b}_1) \\
\mathrm{r}_t &= \sigma_g(\mathrm{W}_2 \mathrm{o}_t + \mathrm{U}_2 s_{t-1} + \mathrm{b}_2) \\
\hat{s}_t &= \sigma_h(\mathrm{W}_3 \mathrm{o}_t + \mathrm{U}_3(\mathrm{r}_t \circ s_{t-1}) + \mathrm{b}_3) \\
s_t &= (1 - \mathrm{z}_t) \circ s_{t-1} + \mathrm{z}_t \circ \hat{s}_t
\end{aligned}
\tag{5.15}
$$

  where z_t and r_t denote the update gate and reset gate vector generated by GRU, $\circ$ is the element-wise product operator, W_i, U_i are weight matrix and b_i are the bias vectors. Particularly, the hidden state s_t is generated by a GRU with inputs of a previous hidden state s_{t-1} and a new candidate hidden state $\hat{s}_t$. Finally, s_t serves as the representation of the current user state.

- **Attentive Meta-path Context Representation.** The attentive meta-path context representation module produces interaction-specific context that captures diverse semantics of meta-paths on user-action interactions. Our attentive meta-path context representation module calculates attention weights over meta-paths conditioned on state-action pairs, thus can capture the influence of each meta-path on user interest drift. In the attentive meta-path context representation module,

we first resort to *Meta-path Based Random Walks* [41] for generating path instances $\rho = \{v_1, \cdots, v_l\}$ under a specific meta-path scheme p through an effective meta-path instance sampling. To further capture both the semantics and structural correlations between different types of nodes, we adopt Convolution Neural Network (CNN) [129] parameterized by Θ to transform ρ of lengths l into meta-path embedding as,

$$\mathrm{c_p} = \text{max-pooling}\left(\left\{CNN\left(\{\mathrm{X}_i^{\rho}\};\Theta\right)\right\}_{i=1}^{L}\right) \tag{5.16}$$

where $\{\mathrm{X}_i^{\rho}\}$ denote the set of embeddings for L path instances from meta-path p, where each X_i^{ρ} denotes the embedding matrix of a path instance ρ. Assume L path instances can be generated under a meta-path p, we apply max pooling operation [144] to aggregate them into one embedding $\mathrm{c_p}$.

We then learn the interaction-specific meta-path context representation. Having obtained the meta-path embeddings $\mathrm{c_p}$ of meta-path p, we pair the meta-path embedding $\mathrm{c_p}$ with the current user state s_t and a dispensing action a_t. The dispensing action a_t can be represented as a one-hot representation a_t overall potential actions in $\mathscr{A}$, however, such a one-hot encoding manner may result in high computation complexity. Thus, we implement a simple embedding lookup layer to transform the one-hot representation a_t of action a_t into low-dimensional dense vectors:

$$\mathrm{e}_t = \mathrm{Q}^{\top} \cdot \mathrm{a}_t \tag{5.17}$$

where $\mathrm{Q}^{\top} \in \mathbb{R}^{|\mathscr{A}| \times d}$ is the parameter matrix which stores the latent factors of actions and d is the embedding dimension, $\mathrm{e}_t \in \mathbb{R}^d$ is the dense embedding of action a_t. Given user state s_t , context embedding $\mathrm{c_p}$ and action embedding e_t, we implement a two-layer attention mechanism as,

$$\boldsymbol{\alpha}^{(1)}_{s_t,a_t,\mathrm{p}} = f\left(\mathrm{W}^{(1)}_u s_t + \mathrm{W}^{(1)}_a \mathrm{e}_t + \mathrm{W}^{(1)}_{\mathrm{p}} \mathrm{c_p} + \mathrm{b}^{(1)}\right) \tag{5.18}$$

$$\boldsymbol{\alpha}^{(2)}_{s_t,a_t,\mathrm{p}} = f\left(\mathrm{W}^{(2)^{\top}} \boldsymbol{\alpha}^{(1)}_{s_t,a_t,\mathrm{p}} + \mathrm{b}^{(2)}\right) \tag{5.19}$$

where $\{\mathrm{W}^{(1)}\}$ and $\mathrm{b}^{(1)}$ denote the weight matrix and the bias vector for the first layer, and the $\mathrm{W}^{(2)}$ and $b^{(2)}$ denote the weight vector and the bias for the second layer, $f(\cdot)$ is set to the ReLU function. As we care about the user-action interaction,

we select all meta-path schemes whose starting node type is *user* while the ending node type is *action* (i.e., item), and denote those meta-path schemes into meta-path schemes set $\mathscr{M}_{u\to a}$. We then normalize the attentive scores $\boldsymbol{\alpha}^{(2)}_{s_t,a_t,\mathrm{p}}$ in Eq. (5.19) over all meta-path schemes in $\mathscr{M}_{u\to a}$ using a softmax function, and derive the final interaction-specific meta-path context representation $\mathrm{c}_{u\to a}\in\mathbb{R}^d$ as a weighted sum:

$$\mathrm{c}_{u\to a}=\sum_{\mathrm{p}\in\mathscr{M}_{u\to a}}\frac{\exp\left(\boldsymbol{\alpha}^{(2)}_{s_t,a_t,\mathrm{p}}\right)}{\sum_{\mathrm{p}'\in\mathscr{M}_{u\to a}}\exp\left(\boldsymbol{\alpha}^{(2)}_{s_t,a_t,\mathrm{p}'}\right)}\cdot\mathrm{c}_{\mathrm{p}} \tag{5.20}$$

- **Target Policy Learning.** For each time $t\in T$, the interaction-specific meta-path context representation can provide important semantics to regulate the user state and the involved actions. Therefore, user state s_t and action representation e_t should be adjusted accordingly based on context representation $\mathrm{c}_{u\to a}$ for the later off-policy learning. Specifically, we first compute the attention vectors of meta-path context on users state and actions in <*user state - meta-path context - action*> pairs, then use these attention vectors to refine the user state/action representations in origin space. Formally, giving user state s_t in Eq. (5.15) and the action representation e_t in Eq. (5.17), and the meta-path based context embedding $\mathrm{c}_{u\to a}$ connecting them, we use a single-layer network to compute the attention vectors $\boldsymbol{\beta}^u$ and $\boldsymbol{\beta}^a$ for user state s_t and action a_t as,

$$\begin{aligned}\boldsymbol{\beta}^u_t&=Relu(\mathrm{W}_u s_t+\mathrm{W}_{u\to a}\mathrm{c}_{u\to a}+\mathrm{b}_u)\\ \boldsymbol{\beta}^a_t&=Relu(\mathrm{W}_a \mathrm{e}_t+\mathrm{W}_{u\to a}\mathrm{c}_{u\to a}+\mathrm{b}_a)\end{aligned} \tag{5.21}$$

where W_u and b_u denote the weight matrix and bias vector for user state attention; W_a and b_a denote the weight matrix and bias vector for action attention. Then, the final representations of user states and actions are computed by using an element-wise product with the attention vectors:

$$\begin{aligned}\tilde{s}_t&=\boldsymbol{\beta}^u_t\odot s_t\\ \tilde{\mathrm{e}}_t&=\boldsymbol{\beta}^a_t\odot \mathrm{e}_t\end{aligned} \tag{5.22}$$

We now transform the refined representations of user state $\tilde{s}_t$ and action $\tilde{\mathrm{e}}_t$, along with interaction-specific context representations $\mathrm{c}_{u\to a}$ into the HIN-enhanced state $s^{u\to a}_t\in\mathbb{R}^d$, to parametrize the policy π_θ. Specifically, the three embedding vectors are combined into a unified representation at the current interaction $t\in T$ as,

$$s^{u\to a}_t=\tilde{s}_t\oplus\mathrm{c}_{u\to a}\oplus\tilde{\mathrm{e}}_t \tag{5.23}$$

where $\oplus$ denotes the vector concatenation operation. The $s_t^{u \to a}$ serves as the final representation of the HIN-enhanced user interests taken at time t. The policy $\pi_\theta(a_t|s_t^{u \to a})$, which is the probability of recommending the action a_t given the possible action space $\mathscr{A}_t$, is modeled as a softmax function:

$$\pi_\theta(a_t \mid s_t^{u \to a}) = \frac{\exp\left(\mathrm{e}_{t+1}^\top s_t^{u \to a}\right)}{\sum_{a_t \in \mathscr{A}_t} \exp\left(\mathrm{e}_t{}^\top s_t^{u \to a}\right)} \tag{5.24}$$

where $\mathrm{e}_{t+1} \in \mathbb{R}^d$ is also derived by the embedding lookup operation as denoted in Eq. (5.17). In fact, e_{t+1} is the embedding of action a_{t+1} which dispensed in the next time step $t+1$.

Counterfactual Risk Minimization for Recommendation. The goal of off-policy learning is to maximize each user's long term satisfaction with the system, given the historical logged data generated by historical logging policy π_0. Remember that the target policy π_θ, which is what we care about most, serves to optimize RS to maximize the objective cumulative rewards in (5.14). To achieve this goal, we have to address the counterfactual question that how much reward would be received if a new target policy π_θ in Eq. (5.24) had been deployed instead of the original logging policy π_0. This counterfactual question is not easy to address, since the target policy π_θ is different from the historical logging policy π_0 in the off-policy setting. Here we apply *Counterfactual Risk Minimization* (CRM) [188] to correct the discrepancy between the target policy π_θ and logging policy π_0, thus to answer the counterfactual question.

It is well-known that the inverse propensity scoring (IPS) estimator is a common practice to correct the discrepancy between π_θ and π_0 [17]. However, the IPS estimator suffers from the "propensity overfitting" issue due to the uncertainty on rare actions [95, 133]; when directly optimizing IPS within a learning algorithm, the results tend to have a large variance. To reduce the variance, we resort to a clipped estimator that caps the propensity ratios (i.e., importance weight) to a maximum value [17]. The core idea is to regulate large weights necessarily associated with actions that are different to the logging policy. The clipped estimator (cIPS) can be represented as

$$L_{\mathrm{cIPS}}(\pi_\theta) = \frac{1}{T} \sum_{t=1}^{T} r_t \min\left\{ \frac{\pi_\theta(a_t \mid s_t^{u \to a})}{\pi_0(a_t \mid s_t^{u \to a})}, c \right\} \tag{5.25}$$

where c is a constant that serves as the regulator for constraining the importance weight $\frac{\pi_\theta(a_t|s_t)}{\pi_0(a_t|s_t)}$ to at most c, smaller value of constant c reduces variance in the gradient estimate, but introduces larger bias. We thus follow Joachims et.al [97] to prevent the additional

bias by adding an empirical variance penalty term λ as,

$$L^{\lambda}_{\text{cIPS}}(\pi_\theta) = \frac{1}{T}\sum_{t=1}^{T}(r_t - \lambda_t)\min\left\{\frac{\pi_\theta\left(a_t \mid s_t^{u\to a}\right)}{\pi_0\left(a_t \mid s_t^{u\to a}\right)}, c\right\} \tag{5.26}$$

where λ_i regulates the corresponding reward r_i at each interaction By plugging Eq. (5.26) into objective function Eq. (5.14), we have:

$$\begin{aligned} R(\pi_\theta) &= \mathbb{E}_{\pi_\theta}\left[\gamma^t L^{\lambda}_{\text{cIPS}}(\pi_\theta)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t\left(r(s_t^{u\to a}, a_t) - \lambda_t\right)\min\left\{\frac{\pi_\theta\left(a_t \mid s_t^{u\to a}\right)}{\pi_0\left(a_t \mid s_t^{u\to a}\right)}, c\right\}\right] \end{aligned} \tag{5.27}$$

5.2.3.2 Top-K Recommendation

We focus on the widely-adopted Top-K recommendation task and utilize the two-stage policy gradient strategy [133] as our learning method. The two-stage setup with candidate generation followed by ranking has been widely adopted in industry [15, 36, 45], which is capable of recommending highly personalized items from a huge item space in real-time. In the training phrase, the trained target policy π_θ in Eq. (5.27) is fed into candidate generation model to form the probability over the possible candidate sets $\mathscr{A}_t \in \mathscr{A}$ conditioning on the current state s_t, denoted by $p_\theta(\mathscr{A}_t \mid s_t)$. The possible candidate sets $\mathscr{A}_t$ can be the combination of any items $i \in \mathscr{I}$. The ranking model delivers the final recommendation results through optimizing together with the candidate generation model, which is drawn from a probability over all action a_t conditioned on the current state s_t and a candidate set $\mathscr{A}_t$, denoted by $q_\theta(a_t \mid s_t, \mathscr{A}_t)$. Assuming that the policy takes a function form π_θ parameterized by $\theta \in \mathbb{R}^d$, the policy gradient of the cumulative reward function Eq. (5.27) w.r.t. θ can be expressed as the following REINFORCE gradient thanks to the log-trick:

$$\begin{aligned} \nabla_\theta R(\pi_\theta) &= \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T}\gamma^t\left(r\left(s_t^{u\to a}, a_t\right) - \lambda_t\right)\nabla_\theta \log \pi_\theta\left(a_t \mid s_t^{u\to a}\right) \\ &\qquad \left\{\frac{\pi_\theta\left(a_t \mid s_t^{u\to a}\right)}{\pi_0\left(a_t \mid s_t^{u\to a}\right)}, c\right\}\Bigg] \\ &= \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T}\gamma^t\left(r\left(s_t^{u\to a}, a_t\right) - \lambda_t\right) \\ &\qquad \left\{\frac{\sum_{\mathscr{A}_t} q_\theta\left(a_t \mid s_t^{u\to a}, \mathscr{A}_t\right)\nabla_\theta p_\theta\left(\mathscr{A}_t \mid s_t^{u\to a}\right)}{\pi_0\left(a_t \mid s_t^{u\to a}\right)}, c\right\}\Bigg] \end{aligned} \tag{5.28}$$

Here, we consider REINFORCE [227], a typical policy gradient method, as our optimization algorithm is to search the optimal policy in recommendation.

5.2.3.3 Computational Complexity Analysis

HINPolicy consists of two key components: 1) HIN-Augmented Policy Learning: which leverages meta-paths and co-attention for state representation learning, 2) Counterfactual Risk Minimization (CRM): which uses importance sampling to correct distribution bias. The HIN-augmented policy learning has two steps - state representation and attentive meta-path context representation - in which the complexity comes from RNN and Meta-path-based Random Walks, respectively. The complexity of CRM-based optimization depends mainly on the complexity of IPS re-weighting, in which the IPS estimator adjusts the M historical interactions and has the complexity of $O(M)$. Analyzing the above, meta-path and state representation learning dominate the computational cost, particularly for large-scale HINs with extensive meta-paths. In addition, the co-attention mechanism scales quadratically with the size of the embedding d and the heads of attention C.

5.2.4 Experiments

To thoroughly evaluate the proposed off-policy method for the recommendation, we conduct extensive experiments to answer the following research questions:

- **RQ1.** How does our HINpolicy perform compared with state-of-the-art off-policy recommendation methods?
- **RQ2.** How does HIN information affect our method and different sparsity levels of user feedback?
- **RQ3.** How do hyper-parameters in our method impact the recommendation performance?

5.2.4.1 Setup

Logging Policy from Logged Data. We adopt two widely used public datasets from different domains, namely `MovieLens` and `Douban-book`. For both datasets, we binarize the feedback data (i.e., ratings) by interpreting ratings of 4 or higher as positive feedback (i.e., $r = 1$), otherwise negative (i.e., $r = 0$). The detailed statistics of all datasets are given in Table 5.5. To facilitate the utility of real-world recommendation datasets in off-policy learning, we start with designing simulation experiments based on an online simulator [260, 265] to recover the missing reward r in partially-observed recommendation

datasets. The logged feedback samples are then acquired by running a logging policy π_0 on the recovered datasets. We adopt the wildly used logging policy, i.e., the uniform-based logging policy.

Table 5.5: Statistics of datasets.

Dataset	#Users	#Items	#Total Feedback	#Feedback Per Customer	#Feedback Per Item	Density
MovieLens-100K	943	1682	100000	106.0445	59.4530 %	6.30%
MovieLens-1M	6040	3952	1000209	165.5975	253.0893 %	3.95%
Douban Book	13024	22347	792062	60.8156	35.4438 %	0.27%

Table 5.6: Meta-path statistics of datasets.

Dataset	Node	Meta-path Schemes
MovieLens-100K	#User(U): 943 #Movie(M): 1,682 #Genre(R): 18 #Gender(G): 2 #Age(A): 7 #Occupation(O): 21	UMGM,UMAM, UMRM,UMOM, UMUM
MovieLens-1M	#User(U): 6,040 #Movie(M): 3,952 #Genre(R): 18 #Gender(G): 2 #Age(A): 7 #Occupation(O): 21	UMGM,UMAM, UMRM,UMOM, UMUM
Douban Book	#User(U): 13,024 #Book(B): 22,347 #Group(G): 2,936 #Author(A): 10,805 #Publisher(P): 1,815 #Year(Y): 64 #Location(L): 38	UBGB,UBAB,UBPB, UBYB,UBLB,UBUB

Contextual Information. We consider two publicly available recommendation datasets, i.e., MovieLens and Douban-book; the selected meta-paths are presented in Table 5.6. These two datasets contain multiple attributes for both users and items, thus can provide rich contextual information for off-policy learning. We only select short meta-paths of at most four steps, since long meta-paths are likely to contain noise. The three used datasets, i.e., MovieLens-100K, MovieLens-1M and Douban-book, are produced with diminishing density of 6.30%, 3.95% and 0.27%, respectively, for testing HINpolicy's capability in dealing with the data sparsity.

Baselines. Existing off-policy learning approaches can be roughly sorted into (1) value-based approaches and (2) policy-based approaches. The former generates a target policy based on modeling the actual reward that is received by a certain action, while the latter directly models the actions that should be taken in order to maximise the total cumulative reward a policy will collect. Note that our method can be categorized as a policy-based approach. We perform our method and three representative methods from these two categories on Top-K recommendation task. We evaluate all baselines on the same logged user feedback samples as in our HINpolicy.

- *Bandit-MLE* [87](1): is a value-based approach that estimates the likely reward (i.e., value) a certain action would yield through Maximum Likelihood Estimation (MLE), then generates target policy by selecting actions that have the maximum value. It applies the IPS estimator to adjust for the difference in the distribution of logging policy and target policy to eliminate bias.

- *POEM* [188] (2): Optimizer for Exponential Models (POEM) is the earliest policy-based approach that uses IPS-based counterfactual risk minimisation for off-policy bias correction.

- *BanditNet* [97] (2): is a recent notable extension of off-policy learning with logged user feedback using counterfactual risk minimisation, it optimises an additional Lagrangian form of SNIPS estimator and extends the off-policy learning to deep neural networks.

We evaluate all baseline methods using Precision@K and Normalized Discounted Cumulative Gain (NDCG)@K with $K = [1, 5, 10, 20]$.

Parameter Settings. We implement baseline models and our proposed HINpolicy on a Linux server with NVIDIA RTX 3090Ti GPU. For a fair comparison, all logged user feedback samples used for training models are generated through our online simulator together with a logging policy. The logged ratings in `Movielens` and `Douban Book` for training the simulator are split as train/test/validate set with a proportion of 60%/20%/20% of original datasets. Without special mention, the final logged user feedback samples are given by applying *uniform* logging policy on the full-information data generated through the trained simulator. As for the policy training, we optimize the two-stage policy gradient with AdaGrad, the same gradient descent method is also applied in all the baseline models. A grid search is conducted to choose the optimal parameter combinations in all models considered, with batch size and learning rate searched in $\{32, 64, 128, 512, 1024\}$

and $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, respectively. The maximum epoch N_{epoch} for all methods is set as 2000, while an early stopping strategy is performed (i.e., if the loss stops to increase, then terminate the model training). Unique settings of our HINpolicy are the dimension of HIN embedding vector d, which is searched in $\{8, 16, 32, 64, 128, 256\}$; the number of random walkers for sampling meta-path node sequences are set to default as $n_{walk} = 1000$ with walk length $l_{walk} = 100$; the weight capping constant with $c = 10$. For evaluation, we adopt the two-stage policy gradient method as in [133], different from conventional Learn-to-Rank problems which directly give ranked lists based on sorting the predicted values (e.g., probability) of user-item pairs, we measure the quality of the recommendations given by the ranking model conditioned on the candidate set provided by the candidate generation model. That is, we firstly select top-n items predicted by the candidate generation model as a candidate set, then feed this candidate set to the ranking model to re-rank these candidates. The candidate set length is set as $n = 20$ for all datasets, while we test the final performance on the trained ranking model with ranking length $K = [1, 5, 10, 20]$.

5.2.4.2 Performance Comparison (RQ1)

Table 5.7 reports the experimental results by an average of 5 repeated experiment runs of evaluation. Note that both our method and all baselines are performed under the uniform-based logging policy. The uniform-based logging policy samples every action at random with an equal probability, which is an idealised (i.e., unbiased) setting that a recommender desire. Analyzing Table 5.7, we have the following observations:

- Our proposed HINpolicy consistently yields the best performance among all datasets on both evaluation metrics. By averaging the performance under all K, our method achieves significant improvements over the best baseline. Particularly, our method improves Precision@K by 25.40%, 5.85%, 38.18% and NDCG@K by 27.47%, 6.38%, 41.78% on `MovieLens-100K`, `MovieLens-1M` and `Douban Book`, respectively. This indicates that HINpolicy indeed improves the Top-K recommendation thanks to the modeling of rich context information.

- Across the datasets, all baseline models achieve downgraded performance on `MovieLens-100K` compared with those on `MovieLens-1M`. We consider this is because the size of the former is much smaller than the latter, while the small dataset size easily renders sub-optimal learning of the recommendation policy. However, our HINpolicy can still achieve satisfactory results and adapts well with limited

Table 5.7: Performance comparison.

Datasets	Metrics	Bandit-MLE	BanditNet	POEM	HINpolicy	Improv.%
MovieLens-100k	Precision@1	0.705	0.671	0.709	**0.894**	26.09%
	Precision@5	0.673	0.663	0.692	**0.862**	24.57%
	Precision@10	0.652	0.645	0.666	**0.832**	24.92%
	Precision@20	0.649	0.622	0.642	**0.818**	26.04%
	NDCG@1	0.692	0.683	0.684	**0.801**	15.75%
	NDCG@5	0.678	0.667	0.661	**0.872**	28.61%
	NDCG@10	0.649	0.621	0.639	**0.848**	30.66%
	NDCG@20	0.627	0.605	0.631	**0.851**	34.87%
MovieLens-1M	Precision@1	0.789	0.727	0.781	**0.883**	11.91%
	Precision@5	0.791	0.811	0.857	**0.931**	8.63%
	Precision@10	0.777	0.867	0.901	**0.925**	2.66%
	Precision@20	0.755	0.888	0.917	**0.919**	0.22%
	NDCG@1	0.741	0.736	0.759	**0.852**	12.25%
	NDCG@5	0.822	0.748	0.821	**0.898**	9.25%
	NDCG@10	0.871	0.822	0.828	**0.896**	2.87%
	NDCG@20	0.908	0.851	0.931	**0.942**	1.18%
Douban Book	Precision@1	0.659	0.625	0.611	**0.943**	43.10%
	Precision@5	0.649	0.616	0.593	**0.875**	34.82%
	Precision@10	0.626	0.602	0.568	**0.852**	36.10%
	Precision@20	0.599	0.572	0.545	**0.831**	38.73%
	NDCG@1	0.608	0.582	0.628	**0.817**	30.10%
	NDCG@5	0.588	0.567	0.612	**0.889**	45.26%
	NDCG@10	0.553	0.548	0.594	**0.868**	46.13%
	NDCG@20	0.549	0.528	0.572	**0.833**	45.63%

samples, since the HIN provides the auxiliary information to augment the off-policy learning. Meanwhile, the degrade can be also found for the performance of baselines on Douban Book and MovieLens-1M. This is because Douban-book is much sparser than MovieLens-1M, with 0.27% and 4.19% density, respectively. Likewise to MovieLens-1M, HINpolicy outperforms all baselines on Douban Book. This indicates that HINpolicy handles the sparsity well with the support of rich side knowledge.

- Across the baseline models, policy-based POEM outperforms value-based Bandit-MLE in most cases, since it learns policy directly through calculating the end-to-end cumulative rewards without incorporating another decision-making stage like value-based approaches. Moreover, considering the IPS estimator used in POEM, although BanditNet contains an advanced SNIPS estimator, it still cannot outperform POEM. We infer that regulating the inverse weights with SNIPS is not well suited in the recommendation task. This is mainly because SNIPS introduces multiplicative control variate to the IPS estimator that heavily penalises the target

policy, limiting its exploration ability of policy learning in the recommendation. On the contrary, we consider leveraging the IPS-based clipped estimator with penalty term, which can reduce bias and variance effectively.

5.2.4.3 Study of HIN (RQ2)

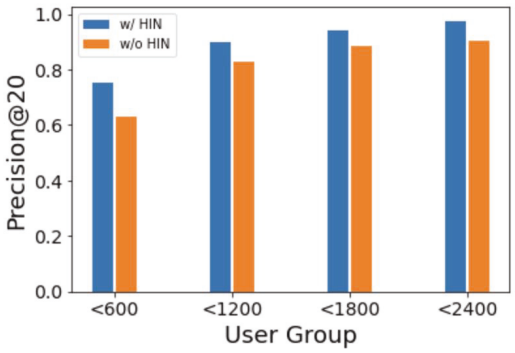
To thoroughly investigate HINpolicy's contributions on alleviating the sparsity issue in HIN information, we conduct how the HIN assist in achieving better recommendation in presence of various sparsity of logged data. To characterize the sparsity levels, we divide users into four groups based on their total amounts of interactions with items. For example, in `MovieLens-1M`, user group 1 represents users who have less than 600 ratings for movies; likewise user group 2, 3 and 4 have less than 1200, 1800 and 2400 ratings, respectively. Hence, user group 1 represents the sparsest data level, while user group 4 represents the densest data level. Table 5.8 shows the overall performance of our HINpolicy with or without HIN, based on 5 runs of repeated training on `MovieLens-1M` and `Douban Book`. With the trained HINpolicy-w/ HIN and HINpolicy-w/o HIN model, we give the test results of the two models on the four user groups and show the comparisons in Figure 5.7. We have the following observations: (1) Apparently, HINpolicy augmented with HIN outperforms the counterpart without HIN, evidences can be found in both Table 5.8 and Figure 5.7. Moreover, facing different user groups as denoted in Figure 5.7, the test recommendation performance of HINpolicy-w/ HIN consistently outperforms HINpolicy-w/o HIN. These promising discoveries confirm HIN has significant effects on policy learning, which can benefit to achieve satisfying recommendations; (2) HIN information has a critical effect on sparse dataset `Douban Book`, especially on the sparsest user group (i.e., ratings <500), with the precision of 0.642% and 0.501% and the NDCG of 0.667% and 0.521% for HINpolicy-w/ HIN and HINpolicy-w/o HIN, respectively. Compared to `MovieLens-1M` having a higher density, our improvement is not as significant as in the `Douban Book` having a lower density. Hence, we conclude that HINpolicy has a more significant effect on the sparse dataset. In a nutshell, incorporating HIN can significantly improve recommendation performance in off-policy setting, while advantages of HIN are more apparent when the dataset is extremely sparse.

5.2.4.4 Case Study (RQ3)

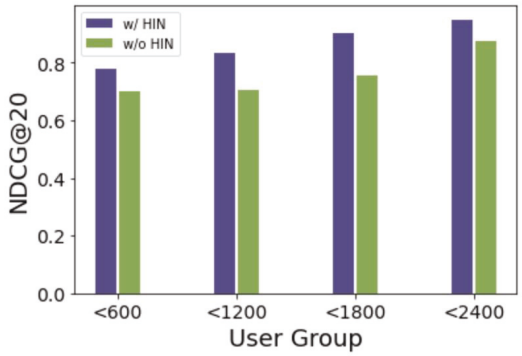
A case study on HINpolicy is also conducted to investigate the impact of parameters on the model performance. Specifically, our method has two key parameters, i.e., embedding size d that controls the latent factor numbers of user state, action and context repre-

Table 5.8: Our performance with (w/) or without (w/o) HIN.

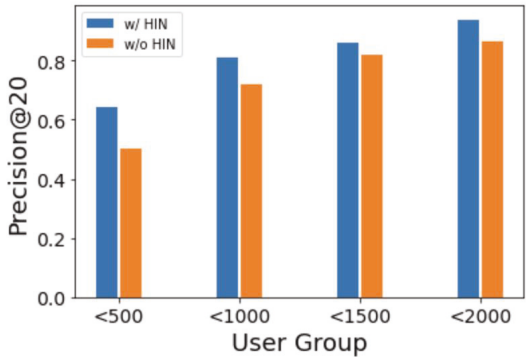
Dataset	Metrics	HINpolicy-w/ HIN	HINpolicy-w/o HIN
MovieLens-1M	Precision@20	0.918	0.730
	NDCG@20	0.942	0.752
Douban Book	Precision@20	0.835	0.708
	NDCG@20	0.831	0.720



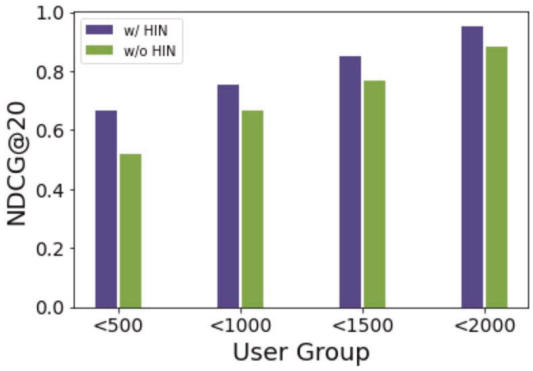
(a) The impact of HIN for Precision on MovieLens-1M.



(b) The impact of HIN for NDCG on MovieLens-1M.



(c) The impact of HIN for Precision on Douban Book.

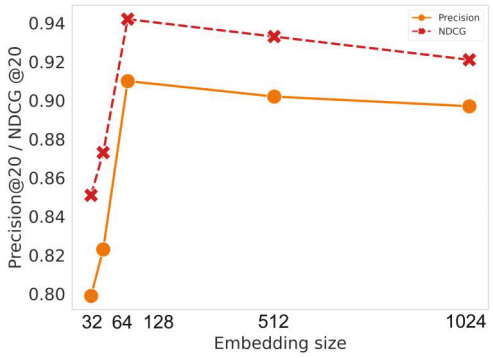


(d) The impact of HIN for NDCG on Douban Book.

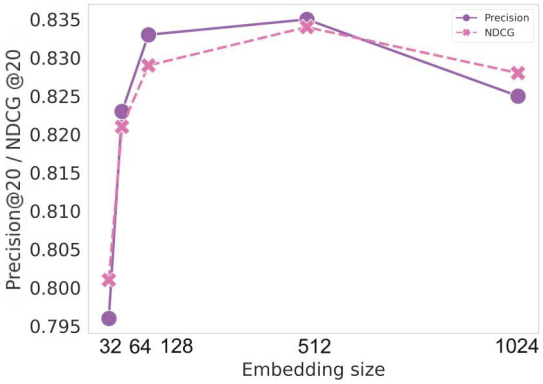
Figure 5.7: Effectiveness analysis on HIN: different user groups control the interaction numbers.

sentations; candidate length n that controls the length of candidate generation set that waits to be re-ranked by the ranking. For all parameters listed above, we vary the value of one parameter while keeping the others unchanged. Figure 5.8 shows the parameter sensitivities of d and n for Precision@20 and NDCG@20 on both MovieLens-1M and Douban Book datasets. From both sub-figures, it is easy to see that relative trends of model performance variations are consistent across different datasets.

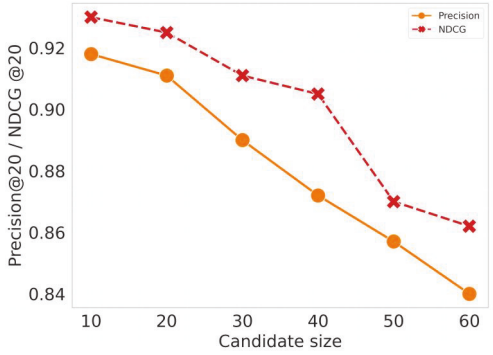
- For the embedding dimension d, in Figure 5.8 (a) (b), the recommendation results



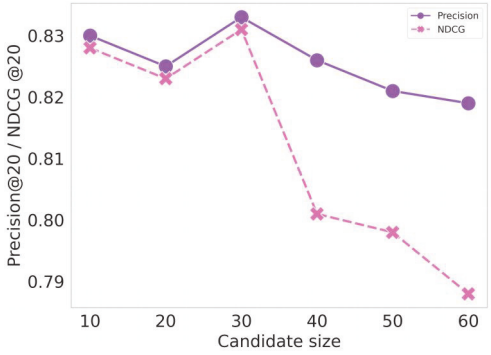
(a) The impact of embedding size d on `MovieLens-1M`.



(b) The impact of embedding size d on `Douban Book`.



(c) The impact of candidate size n on `MovieLens-1M`.



(d) The impact of candidate size n on `Douban Book`.

Figure 5.8: Parameter sensitiveness of embedding size d and candidate length n.

on `MovieLens-1M` and `Douban Book` achieve the peak when $d = 128$, then tend to be stable afterwards. We consider the witness of the increasing trend from $d = 32$ to $d = 128$ is reasonable: as d controls the latent vectors of state/action/context embeddings, exiguous (say $d < 128$) latent factors can not retain sufficient information, thus can not serve well the later policy learning. The stable performance of our HINpolicy after d is set to 128 demonstrates that our model is robust towards varying embedding dimensions.

- For candidate generation length n, in Figure 5.8 (c) (d), we find that the model performance on both datasets decrease when n increases from 10 to 40, while the impact of n is more severe on NDCG@20 compared with it on Precision@20. The candidate generation length n in the two-stage setting we adopt can be interpreted as the number of candidate actions that wait to be re-ranked by the ranking model.

Thus, when n increases, it is harder for the ranking model to give correct results, resulting in the shown decrease trends. We consider this as the "curse" of the two-stage setup with candidate generation followed by ranking, however, we value the two-stage setup for its ability to recommend relevant items from a huge space in real-time, which is suitable for training massive logged user feedback data. We leave the exploration of a more robust ranking model for our future work.

5.2.5 Summary

The work researches the bias in logged user feedback data and proposes the first principled approach to policy to achieve unbiased off-policy learning for the recommendation. The proposed HINpolicy is capable of generating high-quality recommendation policy by virtue of the informative knowledge of the given HIN. In addition, counterfactual risk minimization adaptively corrects the rewards and produces an unbiased recommendation policy that maximizes users' long-term satisfaction. The HINpolicy is evaluated on three real-world recommendation datasets, with extensive experiments and in-depth analyses demonstrating its' robustness and effectiveness.

CHAPTER 6

ASCERTAIN RECOMMENDATION EXPLAINABILITY AND FAIRNESS USING COUNTERFACTUAL

This chapter presents research works that offer recommendation models for the capacity of explainability and fairness. Using counterfactual learning, the CERec model (Counterfactual Explainable Recommendation) generates item attribute-level counterfactual explanations to explain which item attributes attract users' interests; the CFairER model (Counterfactual Explanation for Fairness) generates attribute-level counterfactual explanations to explain which attributes cause item exposure unfairness.

6.1 Counterfactual Explanation

6.1.1 Overview

Research Question. Modern recommendation models become sophisticated and opaque by modeling complex user/item contexts, e.g., social relations and item profiles. Hence, there is a pressing need for faithful explanations to interpret user preferences while enabling model transparency. Explainable recommender systems (XRS) aim to provide personalized recommendations complemented with explanations that answer why particular items are recommended [256]. It is fairly well-accepted that high-quality explanations help improve users' satisfaction and recommendation persuasiveness [217, 234]. Explanations also facilitate system designers in tracking the decision-making of recommendation models for model debugging [49]. Thus, this research aims to investigate how

to generate high-quality explanations that explain users' behaviors well so that users' satisfaction can be improved.

Research Objective. Counterfactual explanation [189] has emerged as a favorable opportunity to solve the above research question. Counterfactual explanation interprets the recommendation mechanism via exploring how minimal alterations on items or users affect the recommendation decisions. Typically, counterfactual explanation is defined as a minimal set of influential factors that, if applied, flip the recommendation decision. To build counterfactual explanations, we have to fundamentally address: "*what the recommendation result would be if a minimal set of factors (e.g., user behaviors/item features) had been different*?" [197]. Thus, the research objective of this work is to answer this counterfactual question through feasible solutions. With counterfactual explanations, users can understand how minimal changes affect recommendations and conduct counterfactual thinking under the "what-if" scenario [153].

Motivations. Through analyzing existing works on counterfactual explainable recommendations in Section 2.3.3, one can find those existing works exhibit the following limitations: they either focus on user action [34, 61, 193] or item aspect explanations [189]. In contrast, item attribute-based counterfactual explanations are largely unexplored. In this research, we find that *item attribute-based counterfactual explanations are more desired in the recommendation scenario*, as they could benefit both users' trust and recommendation performance. On the one hand, item attribute-based counterfactual explanations are usually more intuitive and persuasive in seeking users' trust. This is because users prefer to know detailed information, e.g., which item attribute caused the film "Avatar" not to be recommended anymore? Is it the director of "Avatar"? Besides, it is essential to search for a minimal change in item attributes that would alter the recommendation. We take Figure 6.1 as an example. u_1 gave negative feedback on i_2 with attributes $\{p_1, p_2, p_3, p_4\}$. We could infer attributes $\{p_1, p_2, p_3, p_4\}$ reveal negative preferences of u_1. However, item i_1 with attributes $\{p_1, p_2\}$ was liked by u_1, which somehow reflect positive user preferences on $\{p_1, p_2\}$. Thus, merely using attributes $\{p_1, p_2, p_3, p_4\}$ as explanations for u_1's recommendations would be controversial and misunderstand the user's preference. In fact, the slight changes between i_1's attributes and i_2's attributes, i.e., $\{p_3, p_4\}$, could be the true determinant factors for explaining u_1's dislike. On the other hand, counterfactual explanations could boost recommendation performance since they offer high-quality negative signals of user preferences. Particularly, if we know the slightly changes $\{p_3, p_4\}$ explain u_1's dislike, we could infer that item i_3 with $\{p_3, p_4\}$ would be disliked by u_1 as well. As a result, the counterfactual explanation

helps generate more precise recommendations by filtering out negative items.

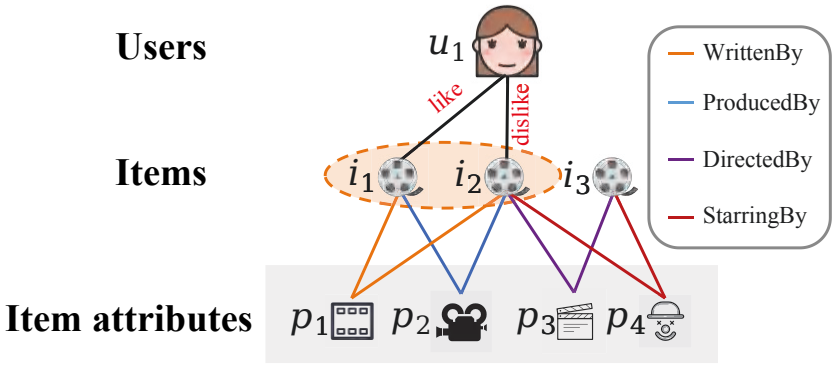


Figure 6.1: Toy example of inferring item attribute-based counterfactual explanations from knowledge graphs.

The Proposed Approach. This work leverages knowledge graphs (KGs) that represent relations among real-world entities of users, items, and attributes to infer attribute-based counterfactual explanations meanwhile boosting recommendation performance. We propose a new Counterfactual Explainable Recommendation (CERec) that crafts the counterfactual optimization problem as a reinforcement learning (RL) task. The RL agent optimizes an explanation policy upon uniformly searching candidate counterfactuals. To reduce the search space, we propose an adaptive path sampler over a KG with a two-step attention mechanism and select item attributes as the counterfactual explanation candidates. With explanation candidates, the RL agent optimizes the explanation policy to find optimal attribute-based counterfactual explanations. We also deploy the explanation policy to a recommendation model to enhance the recommendation.

Contributions. The contributions of this research work are:

- To the best of our knowledge, we are the first to leverage the rich attributes in knowledge graphs to provide attribute-based counterfactual explanations for recommendations.
- We propose an RL-based framework to find the optimal counterfactual explanations, driven by an adaptive path sampler and a counterfactual reward function.
- We use counterfactual explanations to augment the recommendation model for boosting the recommendation and explainability.
- Extensive results on explainability and recommendation performance demonstrate the effectiveness of our method.

6.1.2 Counterfactual Explanation for Recommendation

Following the definitions of counterfactual explanation in Definition 3.4.1 and item attribute-based counterfactual explanation in Definition 3.5.4, the task of this research work can be formulated as the following.

Task formulation. Given user interaction records and a knowledge graph, we can build a collaborative knowledge graph (CKG) $\mathcal{G}$ that unifies user-item interactions and item knowledge as a relational graph. With $\mathcal{G}$, we aim to construct a counterfactual explanation model that exploits rich relations among $\mathcal{G}$ to produce counterfactual items for original recommend items. We formulate our counterfactual explanation model as:

$$j \sim \pi_E(u, i, f_R, \mathcal{G}|\Theta_E) \tag{6.1}$$

where $\pi_E(\cdot)$ is the counterfactual explanation model parameterized with Θ_E, f_R is the Top-K recommendation model that produces the recommendations. The counterfactual explanation model generates empirical distribution over items among $\mathcal{G}$ to yield counterfactual item j for the recommended item i, which is expected to meet the counterfactual goal with a minimal set of item attributes that reverse the recommendation.

We also aim to use counterfactual items produced by π_E to improve the recommendation model f_R. Conceptually, a counterfactual item offers high-quality negative signals to the recommendation, since it owns attributes that make the positive item not match the user's preference anymore. Inspired by pairwise ranking [145], we pair one positive user-item interaction with one counterfactual item to aggregate counterfactual signals into the recommender f_R. Formally,

$$\min_{\Theta_R} -\ln \sigma(f_R(u, i|\Theta_R) - f_R(u, j|\Theta_R)), \forall j \sim \pi_E(\Theta_E) \tag{6.2}$$

where $f_R(\cdot)$ is the recommendation model parameterized with Θ_R and $\sigma(\cdot)$ is the sigmoid function. (u, i) is the positive interaction that satisfies $\exists r \in \mathcal{R}', s.t.(u, r, i) \in \mathcal{G}$. This formulation encourages positive items to receive higher prediction scores from the target user than counterfactual items. As such, the recommendation model not only produces recommendation results, but is also co-trained with the counterfactual explanation model by interactively aggregating counterfactual signals.

6.1.3 Methodology

6.1.3.1 Model Framework

We now introduce our counterfactual explainable recommendation (CERec) framework that generates attribute-based counterfactual explanations while providing improved

Table 6.1: Key notations and descriptions.

Notation	Description
$\mathcal{U}$, u	user set in recommendation and a user.
$\mathcal{I}$, i	item set in recommendation and an item.
$\mathcal{Y}$, y_{ui}	user-item interaction set and an interaction.
$\mathcal{G}$	collaborative knowledge graph.
f_R	a recommendation model.
π_E	an explanation policy.
$\boldsymbol{Q}_u$	a recommendation list for a user u.
Δ_{ui}	a minimal item attributes set for (u,i).
U_u, V_i	latent factors for user u and item i.
h_u	the embedding of user entity u.
e_t, h_{e_t}	an item entity and its embedding.
e'_t, $\mathrm{h}_{e'_t}$	an item attribute entity and its embedding.
$\mathcal{N}_e$	neighbor set of entity e.
α_1	attention score for the first sampling step.
α_2	attention score for the second sampling step.
a_t, s_t, $r(s_t,a_t)$	the action, state, reward at step t.
Θ_R	recommendation model parameters.
Θ_E	counterfactual explanation model parameters.

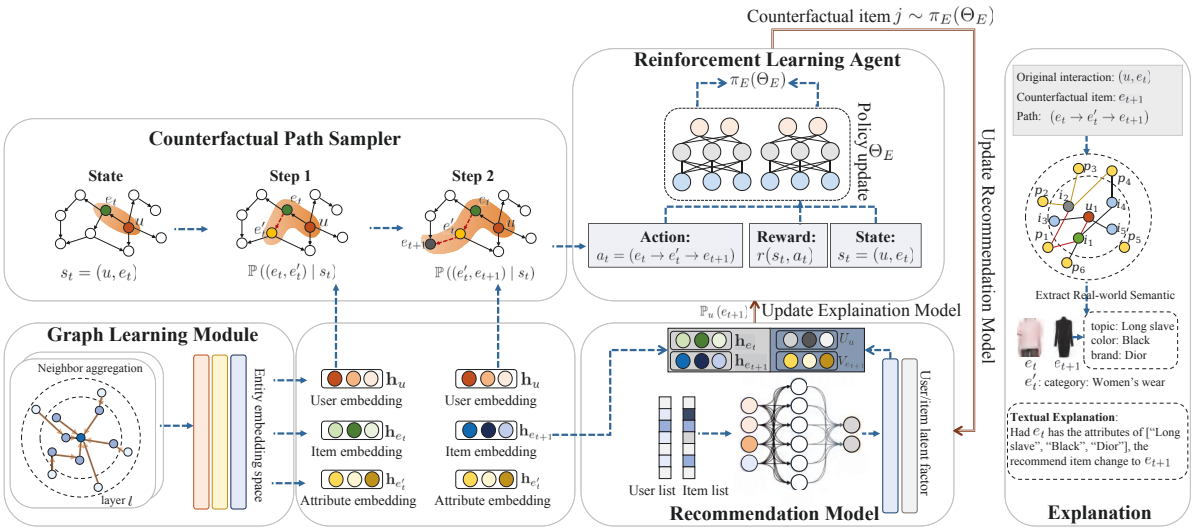


Figure 6.2: CERec framework.

recommendations. The primary notations used throughout this paper are listed in Table 6.1. Our CERec is operated under a CKG [204] that unifies user-item interactions and a knowledge graph. CERec consists of three modules - one recommendation model, one graph learning module and the proposed counterfactual explanation model. The recommendation model generates ranking scores and is co-trained with the counterfactual explanation model by interactively aggregating the produced counterfactual items. The graph learning module embeds users, items, and attribute entities among a given CKG as embedding vectors. The counterfactual explanation model conducts effective path sampling based on entity embeddings and ranking scores to discover high-quality counterfactual items. Two main parts are performed in our counterfactual explanation model: 1) counterfactual path sampler: uses entity embeddings to sample paths as actions for reinforcement learning agent; 2) reinforcement learning agent: learns explanation policy by optimizing the cumulative counterfactual rewards of deployed actions from the sampler. We depict the framework of CERec in Figure 6.2. We introduce the recommendation model and the graph learning module below. Our counterfactual explanation model is detailed in the next section.

Recommendation Model. We now present the recommendation model f_R that uses user and item latent factors for Top-K recommendation. Here, we employ the pairwise learning-to-rank model CliMF [178]. The CliMF initializes the IDs of users and items as latent factors, and updates latent factors by directly optimizing the Mean Reciprocal Rank (MRR) to predict ranking scores of items for users. It is worth noting that f_R can be any model as long as it takes users' and items' embeddings as part of the input and produces ranking results, which makes our counterfactual explanation framework applicable to a broad scope of models, e.g., neural networks [206]. Specifically, we first map users and items into latent factors with the recommendation model,

$$f_R(u,i) = U_u^\top V_i \tag{6.3}$$

where U_u and V_i denote d-dimensional latent factors for user u and item i, respectively. We use the pairwise Mean Reciprocal Rank loss [178] to define our objective function to optimize recommendation model parameters Θ_R as below.

$$\mathscr{L}_R = \min_{\Theta_R} \sum_{(u,i)\in\mathscr{Y}} [\ln\sigma(f_R(u,i)) + \sum_{j=1} \ln(1-\sigma(f_R(u,j) - f_R(u,i)))] \tag{6.4}$$

where $\mathscr{Y} \in \mathbb{R}^{M\times N}$ is the historical user-item interaction matrix, $\sigma(\cdot)$ is the sigmoid function and $j \sim \pi_E(\Theta_E)$ is the counterfactual item generated from our counterfactual explanation model. By optimizing Eq. (6.4), we can get the ranking score for each item i

from user u. Formally,

$$\mathbb{P}_u(i) = \frac{\exp\left(U_u^\top V_i\right)}{\sum_{k=1}^{K} \exp\left(U_u^\top V_k\right)} \tag{6.5}$$

where K is the length of user u's recommendation list $\boldsymbol{Q}_u$, $\mathbb{P}_u(i)$ indicates the ranking score of an item i in u's recommendation list.

Graph Learning Module. The graph learning module (GLM) learns users, items and attributes representations (i.e., embeddings) from the given collaborative knowledge graph $\mathcal{G}$. The learned embeddings are deployed into our counterfactual explanation model to: (1) calculate the importance scores of user, item and attribute entities to form sampling paths as actions for counterfactual path sampler; (2) calculate the similarities among item entities to define the counterfactual rewards of deployed actions for the reinforcement learning agent.

Inspired by recent advances in Graph Neural Networks (GNNs) [91, 206] for graph data representation, we employ GraphSAGE [74] in our GLM to learn representations for users, items and attributes entities. For an entity $e \in \mathcal{G}$, the GraphSAGE aggregates the information propagated from its neighbors $\mathcal{N}_e$ to learn e's representation. As a user entity would connect with entities whose type is the item, i.e., $\mathcal{N}_e \in \mathcal{I}$, the learned user embeddings capture the influence of historical user-item interactions. Analogously, item entities connect with item attribute entities such that the learned item embeddings absorb context information from item attributes. In particular, we firstly initialize entity representations at the 0-th layer of GraphSAGE with Multi-OneHot [254] by mapping entity IDs into embeddings, where the embedding for an entity e is denoted by h_e^0. Then, at the l-th graph convolutional layer, an entity e receives the information propagated from its neighbors to update its representation,

$$\mathrm{h}_e^{(l)} = \delta\left(\mathrm{W}^{(l)}\left(\mathrm{h}_e^{(l-1)} \| \mathrm{h}_{\mathcal{N}_e}^{(l-1)}\right)\right) \tag{6.6}$$

where $\mathrm{h}_e^{(l)} \in \mathbb{R}^{d_l}$ is the embedding of an entity e at layer l and d_l is the embedding size; $\mathrm{W}^{(l)} \in \mathbb{R}^{d_l \times 2d_{l-1}}$ is the weight matrix, $||$ is the concatenation operation and $\delta(\cdot)$ is a nonlinear activation function as LeakyReLU [237]. $\mathrm{h}_{\mathcal{N}_e}^{(l-1)}$ is the information from e's neighbors $\mathcal{N}_e$ and is given by:

$$\mathrm{h}_{\mathcal{N}_e}^{(l-1)} = \sum_{e' \in \mathcal{N}_e} \frac{1}{\sqrt{|\mathcal{N}_e||\mathcal{N}_{e'}|}} \mathrm{h}_{e'}{}^{(l-1)} \tag{6.7}$$

where $\mathcal{N}_e = \left\{e' \mid \left(e, e'\right) \in \mathcal{G}\right\}$ denotes e's connected entities.

Having obtained the representations $\mathrm{h}_e^{(l)}$ at each graph convolutional layer $l \in \{1, \cdots, L\}$, we adopt layer aggregation mechanism [238] to concatenate embeddings at all layers into a single vector, as follows:

$$\mathrm{h}_e = \mathrm{h}_e^{(1)} + \cdots + \mathrm{h}_e^{(L)} \tag{6.8}$$

where $\mathrm{h}_e^{(i)}$ is the embedding for an entity e at the i-th layer. By performing layer aggregation, we can capture higher-order propagation of entity pairs across different graph convolutional layers. After stacking L layers, we obtain the final representation for each entity among the CKG. Note that in the following, we use h_u to denote the embedding of user entity u, while h_{e_t} is the item embedding for item entity e_t and $\mathrm{h}_{e'_t}$ is the embedding for item attribute entity e'_t.

Counterfactual Explanation Model. Our counterfactual explanation model contains two main parts: the counterfactual path sampler and the reinforcement learning agent. The counterfactual path sampler performs two-step attention on users, items, and attribute embeddings from GLM to search for a high-quality path as action a_t for each state s_t. Then, action a_t and state s_t are fed into our reinforcement learning agent to learn the explanation policy. Based on ranking scores produced by the recommendation model and item embeddings, the reinforcement learning agent learns the reward $r(s_t, a_t)$ at state s_t and updates the explanation policy $\pi_E(\Theta_E)$ accordingly. Finally, with the learned explanation policy π_E and path histories, our CERec generates attribute-based counterfactual explanations for recommendations. We detail our counterfactual explanation model in the next section.

6.1.3.2 Reinforced Learning for Counterfactual Explanation Model

We now introduce our counterfactual explanation model assisted by GLM and recommendation model to generate explanation policy π_E over reinforcement depth T. Our counterfactual explanation model contains two main parts: *counterfactual path sampler* (CPS) performs attention mechanisms on CKG entities to sample paths as actions for reinforcement learning agent; *reinforcement learning agent* learns the explanation policy π_E by optimizing cumulative counterfactual rewards of the sampled actions from CPS. We introduce each part in our counterfactual explanation model as follows.

Counterfactual Path Sampler. The counterfactual path sampler (CPS) conducts path exploration over CKG to sample paths as actions for the latter explanation policy learning. The basic idea is to condition on the target user, start from the recommended item, learn to navigate to its item attribute, and then yield the potential counterfactual

item along the sampling paths. In practice, to conduct such higher-order path sampling, large-scale CKGs are required since they encode rich relations, i.e., more than one-hop connectivity. As a result, counterfactual items are sampled from higher-hop neighbors of target user-item interactions to form the candidate action space. However, learning counterfactual explanations from the whole candidate action space is infeasible since the space would cover potentially enormous paths. Thus, our CPS is designed to reduce the candidate action space by filtering out irrelevant paths and selecting important paths for later policy learning. Here, we employ attention mechanisms to calculate the importance of the visited entity condition on the source entity to generate sampling paths. We now introduce our CPS that samples paths as actions a_t at each state s_t.

In particular, at each state s_t, our CPS produces a path as an action by a probability of $\mathbb{P}(a_t|s_t)$. Formally, $a_t \sim \mathbb{P}(a_t|s_t) = (e_t \rightarrow e'_t \rightarrow e_{t+1})$ is the path that roots at an item e_t towards another item proposal e_{t+1}, where $(e_t, e'_t), (e'_t, e_{t+1}) \in \mathcal{G}$ and $e_t, e_{t+1} \in I$ are connected via the item attribute e'_t. The action $a_t = \left(e_t \rightarrow e'_t \rightarrow e_{t+1}\right)$ is generated by the two-step path sampling: 1) choose an outgoing edge from e_t to the internal item attribute entity e'_t; 2) determine the third entity e_{t+1} conditioned on e'_t. We separately model the confidence of each exploration step into two attention mechanisms, i.e., $\mathbb{P}\left(\left(e_t, e'_t\right) \mid s_t\right)$ and $\mathbb{P}\left(\left(e'_t, e_{t+1}\right) \mid s_t\right)$.

The first attention mechanism $\mathbb{P}\left(\left(e_t, e'_t\right) \mid s_t\right)$ specifies the importance of item attributes for e_t, which are sensitive to state $s_t = (u, e_t)$, i.e., user u and item e_t. Formally, for each outgoing edge from e_t to its item attribute $e'_t \in \mathcal{N}_{e_t}$, we first obtain the embeddings of item e_t, attribute e'_t and user u from Eq (6.8), denoted by h_{e_t}, $\mathrm{h}_{e'_t}$ and h_u. Then, the importance score of item attribute e'_t is:

$$\alpha_1\left(e_t, e'_t\right) = \mathrm{h}_u^\top \delta\left(\mathrm{h}_{e_t} \odot \mathrm{h}_{e'_t}\right) \tag{6.9}$$

where α_1 is the attention score at the first attention mechanism. $\odot$ is the element-wise product and $\delta(\cdot)$ is LeakyReLU [237]. Thereafter, we normalize the scores of all neighbors of e_t as:

$$\mathbb{P}\left(\left(e_t, e'_t\right) \mid s_t\right) = \frac{\exp\left(\alpha_1\left(e_t, e'_t\right)\right)}{\sum_{e''_t \in \mathcal{N}_{e_t}} \exp\left(\alpha_1\left(e_t, e''_t\right)\right)} \tag{6.10}$$

where $\mathcal{N}_{e_t}$ is the neighbor item attributes set for e_t.

Having selected item attribute e'_t, we employ another attention mechanism $\mathbb{P}\left(\left(e'_t, e_{t+1}\right) \mid s_t\right)$ to decide yield which item from its neighbors $\mathcal{N}_{e'_t}$ as the counterfactual item proposal. We firstly calculate the attention score of $e_{t+1} \in \mathcal{N}_{e'_t}$ based on attribute embedding of e'_t,

item embedding of e_{t+1} and user embedding of u, as:

$$\alpha_2\left(e'_t, e_{t+1}\right) = \mathrm{h}_u^\top \delta\left(\mathrm{h}_{e'_t} \odot \mathrm{h}_{e_{t+1}}\right) \tag{6.11}$$

where α_2 is the attention score at the second attention mechanism. h_u, $\mathrm{h}_{e'_t}$ and $\mathrm{h}_{e_{t+1}}$ are embeddings of user u, attribute e'_t and item e_{t+1}. Then, we normalize attention scores for all item neighbors in $\mathcal{N}_{e'_t}$ to generate the selection probability of item e_{t+1}. Since we care for generating valid counterfactual items as proposals, we filter out irrelevant items that do not meet the counterfactual goal, i.e., those being recommended for u in the recommendation list $\boldsymbol{Q}_u$. Formally,

$$\mathbb{P}\left(\left(e'_t, e_{t+1}\right) \mid s_t\right) = \begin{cases} \frac{\exp(\alpha_2(e'_t, e_{t+1}))}{\sum_{e''_{t+1} \in \mathcal{N}_{e'_t}} \exp(\alpha_2(e'_t, e''_{t+1}))}, & e_{t+1} \notin \boldsymbol{Q}_u \\ 0, & e_{t+1} \in \boldsymbol{Q}_u \end{cases} \tag{6.12}$$

Finally, the two attention mechanisms are aggregated into the CPS $\mathbb{P}(a_t \mid s_t)$ to yield the path $a_t = \left(e_t \to e'_t \to e_{t+1}\right)$ as an action for each state s_t:

$$\mathbb{P}(a_t \mid s_t) = \mathbb{P}\left(\left(e_t, e'_t\right) \mid s_t\right) \cdot \mathbb{P}\left(\left(e'_t, e_{t+1}\right) \mid s_t\right) \tag{6.13}$$

where $\mathbb{P}\left(\left(e_t, e'_t\right) \mid s_t\right)$ is the probability of stepping from e_t to e'_t and is derived from Eq. (6.10); $\mathbb{P}\left(\left(e'_t, e_{t+1}\right) \mid s_t\right)$ derived from Eq. (6.12) is the probability of selecting e_{t+1} as the counterfactual item proposal. With $\mathbb{P}(a_t \mid s_t)$, we can generate the action a_t for each state s_t for explanation policy learning.

Reinforcement Learning Agent. We define the counterfactual explanation model as the path-based reinforcement learning (RL) agent to discover counterfactual items. Each action a_t is a path toward a candidate counterfactual item. The counterfactual reward $r(s_t, a_t)$ measures whether the action a_t returns a valid counterfactual item for the current state s_t.

- **Agent.** Formally, the counterfactual explanation model is formulated as a Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$, where $s_t \in \mathcal{S}$ is the state absorbing the current user and the visited entity, $a_t \in \mathcal{A}$ is the action deposited to the current state. $\mathcal{P}$ is the transition of states, and $\mathcal{R}$ is the reward function. In the policy learning, the explanation policy $\pi_E(a_t|s_t)$ selects an action $a_t \in \mathcal{A}$ to take conditioning on the current state $s_t \in \mathcal{S}$, and the counterfactual explanation model receives counterfactual reward $r(s_t, a_t) \in \mathcal{R}$ for this particular state-action pair. The final explanation policy is learned to maximize the expected cumulative counterfactual rewards. We introduce these key elements for RL as follows.

- State $\mathscr{S}$: is a continuous state space describing a target user and the currently visited item entity among the CKG. Formally, for a user u, at step t, state s_t is defined as $s_t = (u, e_t)$, where $u \in \mathscr{U}$ is a user and $e_t \in \mathscr{I}$ is the item entity the agent visits currently. The initial state s_0 is (u, i) and i is the positive item of u, i.e., $y_{ui} \in \mathscr{Y}$.
- Action $\mathscr{A}$: is a discrete space containing actions available for policy learning. The action $a_t \in \mathscr{A}$ is a path sampled from our CPS.
- State Transition $\mathscr{P}$: is the state transition containing transition probabilities of the current states to the next states. Given a_t at s_t, the transition to the next state s_{t+1} is $\mathbb{P}(s_{t+1} \mid s_t, a_t) \in \mathscr{P} = 1$, where $s_{t+1} = (u, e_{t+1})$, $s_t = (u, e_t)$ and $a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1})$.
- Counterfactual Reward $\mathscr{R}$: $r(s_t, a_t) \in \mathscr{R}$ is the counterfactual reward measures whether the visited item e_{t+1} is a valid counterfactual item by deploying action $a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1})$ at state s_t, which is defined based on the two criteria: 1) *Rationality* [197]: e_{t+1} should receive high confidence of being removed from the current user u's recommendation list compared with the original item e_t; 2) *Similarity* [47]: as a counterfactual explanation requires the minimal change of item attributes between counterfactual item and original item, e_{t+1} should be as similar as possible with the original item e_t. The formal definition of the reward $r(s_t, a_t)$ is given by:

$$r(s_t, a_t) = \begin{cases} 1 + \cos(\mathrm{h}_{e_t}, \mathrm{h}_{e_{t+1}}), \text{ if } \mathbb{P}_u(e_t) - \mathbb{P}_u(e_{t+1}) \geq \epsilon \\ \cos(\mathrm{h}_{e_t}, \mathrm{h}_{e_{t+1}}), \text{ otherwise} \end{cases} \tag{6.14}$$

  where ϵ is the recommendation threshold determines *Rationality*. ϵ is defined as the margin between ranking scores of the original item e_t and the K-th item (i.e., $\boldsymbol{Q}_u^K$) in u's recommendation list $\boldsymbol{Q}_u$, i.e., $\epsilon = \mathbb{P}_u(e_t) - \mathbb{P}_u(\boldsymbol{Q}_u^K)$. $\cos(\mathrm{h}_{e_t}, \mathrm{h}_{e_{t+1}})$ is the cosine similarity between item embeddings h_{e_t} and $\mathrm{h}_{e_{t+1}}$ and is used to measure *Similarity*, i.e., $\cos(\mathrm{h}_{e_t}, \mathrm{h}_{e_{t+1}}) = \frac{\langle \mathrm{h}_{e_t}, \mathrm{h}_{e_{t+1}} \rangle}{\|\mathrm{h}_{e_t}\| \|\mathrm{h}_{e_{t+1}}\|}$. Note that h_{e_t} and $\mathrm{h}_{e_{t+1}}$ are obtained by Eq. (6.8).

- **Objective Function.** Using the trajectories $\{\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}\}$ from the agent, our counterfactual explanation model seeks a counterfactual explanation policy π_E that maximizes the cumulative rewards $R(\pi_E)$ over reinforcement depth T:

$$R(\pi_E) = \mathbb{E}_{s_0 \sim \mathscr{S}, a_t \sim \mathbb{P}(a_t \mid s_t), s_{t+1} \sim \mathbb{P}(s_{t+1} \mid s_t, a_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{6.15}$$

where T is the terminal step determines reinforcement depth, γ^t is a decay factor at current step $t \in T$. π_E is the explanation policy that produces counterfactual items for users' recommended items. The final counterfactual explanations are formed by distilling the attributes of counterfactual items and their real-world semantics.

Algorithm 1: Recommendation Model and Explanation Policy Optimization

Input : Users $\mathcal{U}$, items $\mathcal{I}$, user-item interactions $\mathcal{Y}$, embedding size d
Output : Optimized recommendation model parameters Θ_R and counterfactual explanation model parameters Θ_E

Initialize Θ_R with uniform sampling;
Initialize Θ_E with random weights;
for *epoch* $\leftarrow 1$ *to* N **do**
  Initialize accumulate reward $R = 0$, reward $r(s_t, a_t) = 0$, state $s_0 \in \mathcal{R}^d$ with random weights, action a_0 uniformly sampled from $\mathcal{I}$, decay factor $\gamma^0 = 1$;
  Fix Θ_R ; `// counterfactual explanation model optimization`
  Receive initial observation state s_1;
  for $t \leftarrow 1$ *to* T **do**
    Sample action a_t (CPS $\mathbb{P}(a_t|s_t)$);
    Execute action a_t and observe the reward $r(s_t, a_t)$;
    Accumulate reward $R \leftarrow R + \gamma^t r(s_t, a_t)$;
    Transit to next state s_{t+1};
  end
  Update Θ_E using REINFORCE gradient:
  $\nabla_{\Theta_E} R(\pi_E) \simeq \frac{1}{T} \sum_{t=0}^{T} \left[\gamma^t r(s_t, a_t) \nabla_{\Theta_E} \log \mathbb{P}(a_t|s_t)\right]$;
  Fix Θ_E ; `// recommendation model optimization`
  for *iteration* $\leftarrow 1$ *to* M **do**
    Sample a counterfactual item $j \sim \pi_E(\Theta_E)$ for user $u \in \mathcal{U}$;
    Compute $\mathcal{L}_R$;
    Update Θ_R using SGD on loss $\mathcal{L}_R$;
  end
end
return Optimized Θ_R and Θ_E;

6.1.3.3 Model Optimization

We adopt iteration optimization [96] to optimize the recommendation model and the counterfactual explanation model.

Recommendation Model Optimization. We first initialize the recommendation model by pairing one positive interaction $y_{ui} \in \mathcal{Y}$ with one unobserved item $v \in \mathcal{I}$ sampled

from uniform sampling [162]. Then, the recommendation model is optimized by training together with the counterfactual explanation model. At each iteration, the counterfactual explanation model outputs a counterfactual item $j \sim \pi_E(\Theta_E)$, j is then fed into the recommendation model to update user and item latent factors U and V in Eq. (6.3). Finally, the loss $\mathscr{L}_R$ in Eq. (6.4) w.r.t. model parameters Θ_R is optimized by using stochastic gradient descent (SGD).

Explanation Policy Optimization. Our counterfactual explanation model involves discrete sampling steps, i.e., the counterfactual path sampler, which blocks gradients when performing differentiation. Conventional policy gradient methods such as stochastic gradient descent fail to calculate such hybrid gradients. Thus, we solve the optimization problem through REINFORCE with baseline [186] for explanation policy optimization. Having obtained the policy optimization function from Eq. (6.15), and the sampling function from Eq. (6.13), the optimization goal is to combine the two functions and optimize them together with a REINFORCE gradient w.r.t. model parameters Θ_E:

$$
\begin{aligned}
\mathscr{L}_E &= \nabla_{\Theta_E} R(\pi_E) \\
&= \nabla_{\Theta_E} \mathbb{E}_{s_0 \sim \mathscr{S}, a_t \sim \mathbb{P}(a_t|s_t), s_{t+1} \sim \mathbb{P}(s_{t+1}|s_t,a_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \\
&\simeq \frac{1}{T} \sum_{t=0}^{T} \left[\gamma^t r(s_t, a_t) \nabla_{\Theta_E} \log \mathbb{P}(a_t \mid s_t) \right]
\end{aligned}
\tag{6.16}
$$

where Θ_E are learnable parameters for our counterfactual explanation model.

6.1.3.4 Computational Complexity Analysis

Our recommendation model performs matrix factorization with a complexity of $O(|\mathscr{O}|)$. For the graph learning module, establishing node representations has a complexity of $O(\sum_{l=1}^{L}(|\mathscr{G}| + |\mathscr{O}^+|)d_l d_{l-1})$, where L is the number of layers, $|\mathscr{G}|$ is the size of the graph, $|\mathscr{O}^+|$ is the size of the positive observation set, and d_l and d_{l-1} are the dimensions of the current and previous layers, respectively. For the counterfactual explanation model, the complexity is mainly determined by the attention score calculation in the counterfactual path sampler. The complexity of the attention score calculation is $O(2T|\mathscr{O}^+||\tilde{\mathscr{N}_e}|d^2)$, where T is the reinforcement depth, $|\mathscr{O}^+|$ is the size of the positive observation set, $|\tilde{\mathscr{N}_e}|$ is the size of the sampled counterfactual neighbors, and d is the dimension of the embeddings. In total, the time complexity is given by: $O(|\mathscr{O}| + \sum_{l=1}^{L}(|\mathscr{G}| + |\mathscr{O}^+|)d_l d_{l-1} + 2T|\mathscr{O}^+|n_2 d^2)$.

6.1.4 Experiments

We thoroughly evaluate the proposed CERec for counterfactual explainable recommendations on three publicly available datasets. We evaluate CERec in terms of recommendation performance and explainability performance to answer the following research questions:

- **RQ1.** Could CERec with a counterfactual explanation model improve the recommendation performance compared with state-of-the-art recommendation models?
- **RQ2.** Are the counterfactual explanations generated by CERec appropriate to explain users' preferences?
- **RQ3.** How do different components (i.e., graph learning module, counterfactual path sampler, reinforcement learning agent) affect CERec's performance?
- **RQ4.** How do different parameters (i.e., reinforcement depth T, training epoch) impact CERec's performance?
- **RQ5.** How is the computation cost of CERec?

6.1.4.1 Setup

Dataset. We use three publicly available datasets: `Last-FM`, `Amazon-book` and `Yelp2018`. The statistics of these datasets are presented in Table 6.2, which depicts historical user-item interactions and the knowledge graphs. The `Amazon-book` [1] [137] dataset is a widely used book recommendation dataset, and the `Last-FM` [2] [20] is a music listening dataset. For `Amazon-book` and `Last-FM`, we first map their items into Freebase [13] entities. Then, the knowledge graphs for `Amazon-book` and `Last-FM` are built by extracting knowledge-aware facts for each item from the Freebase. `Yelp2018` [3] [204] is a business recommendation dataset. For `Yelp2018`, we extract its item knowledge from the business information network to construct the knowledge graph. Each dataset is processed by the following settings: for user-item interactions, we adopt a 10-core setting, i.e., retaining users and items with at least ten interactions. Each knowledge-aware fact (i.e., <user, item>, <item, attribute>) is represented as an edge among the collaborative knowledge graph (CKG). Moreover, to ensure CKG quality, we filter out infrequent entities (i.e., lower than 10 in both datasets) and retain the relations appearing in at least 50 triplets.

[1] http://jmcauley.ucsd.edu/data/amazon
[2] https://grouplens.org/datasets/hetrec-2011/
[3] https://www.yelp.com/dataset

Table 6.2: Statistics of datasets.

Dataset		Last-FM	Amazon-book	Yelp2018
User-Item Interaction	#Users	23,566	70,679	45,919
	#Items	48,123	24,915	45,538
	#Interactions	3,034,796	847,733	1,185,068
	#Density	0.268%	0.048%	0.057%
Knowledge Graph	#Entities	58,266	88,572	90,961
	#Relations	9	39	42
	#Triplets	464,567	2,557,746	1,853,704

Evaluation Metrics. To evaluate the recommendation performance, we use three popular Top-K recommendation metrics: Recall@K, Normalized Discounted Cumulative Gain (NDCG)@K and Hit Ratio (HR)@K. The K is set as 20 by default. The average results w.r.t. the metrics over all users are reported in Table 6.3 while a Wilcoxon signed-rank test is performed in Table 6.3 to evaluate the significance. We evaluate the quality of explanations in terms of *consistency*. The *consistency* measures to what extent the attributes in counterfactual explanations are appropriate to explain users' preferences. We adopt the explanation Precision, Recall and F_1 protocols following CountER [189]. In particular, we first build the ground-truth attribute sets of evaluations for Precision, Recall and F_1 protocols. As we aim to search for a minimal item attributes set that can flip the positive user-item interaction to the negative. The ground-truth set, therefore, absorbs item attributes that cause the user to dislike the item. Formally, the ground truth set is defined as $\mathcal{O}_{ui} = \{o^1_{ui}, \cdots, o^p_{ui}\}$, where $o^p_{ui} = 1$ if user u has negative preference on the p-th attribute of item i; otherwise, $o^p_{ui} = 0$. Our model produces the attributes set $\Delta_{ui} = \{a^1_{ui}, \cdots, a^r_{ui}\}$, which constitutes the counterfactual explanation for user-item pair (u, i). Then, for each user-item pair, the Precision, Recall and F_1 of Δ_{ui} with regard to $\mathcal{O}_{ui}$ are calculated by:

$$\begin{gathered} \text{Precision} = \frac{\sum_{p=1}^{r} o^p_{ui} \cdot I(a^p_{ui})}{\sum_{p=1}^{r} I(a^p_{ui})}, \ \text{Recall} = \frac{\sum_{p=1}^{r} o^p_{ui} \cdot I(a^p_{ui})}{\sum_{p=1}^{r} o^p_{ui}}, \\ F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{gathered} \tag{6.17}$$

where $I\left(a^p_{ui}\right)$ is an identity function such that $I(\cdot) = 1$ when $a^p_{ui} \neq 0$ and otherwise, $I(\cdot) = 0$.

Baselines. To evaluate the recommendation and explainability performance, we compare our CERec with five popular recommendation models as well as three attribute-aware baselines. Each baseline is aligned with different evaluation tasks either for (I) recommendation evaluation or (II) explainability evaluation. The baseline models are listed in the following:

- **NeuMF** [81] (I): extends Matrix Factorization (MF) to Deep Neural Network (DNN) for modeling user-item interactions.
- **MCrec** [88] (I): uses DNN to model context information from Heterogeneous Information Network (HIN) for recommendations.
- **KGpolicy** [218] (I): uses an RL agent to explore negative items over a KG. It trains a Bayesian Personalized Ranking model with sampled negatives for recommendations.
- **KGQR** [263] (I): models KG information with a Graph Convolution Network (GCN). It learns a recommendation policy with KG-enhanced state representations.
- **KGAT** [204] (I): introduces the CKG concept and learns node embeddings by aggregating CKG neighbors for recommendations.
- **NeuACF** [75] (II): learns attribute-based latent factors of users and items based on their similarities with user preferences. Explanations are the top 10 similar attributes with user preferences.
- **CaDSI** [208] (II): disentangles user embeddings as separated user intent chunks. It learns attribute-aware intent embeddings by assigning item context trained from a HIN to each user intent chunk. Explanations are the top 10 attributes of user intent.
- **RDExp** (II): We randomly select 10 attributes from the item attribute space for each user-item interaction and generate explanations based on the selected attributes.

Implementation Details. We train the recommendation model by the CliMF [4] with the train/test/validate sets, which are split from user-item interactions with a proportion of 60%/20%/20% of the original dataset. We optimize the CliMF using stochastic gradient descent (SGD). The same data splitting and gradient descent methods are also applied in all baselines. The three datasets for training our counterfactual explanation model are prepossessed into collaborative knowledge graphs, and the REINFORCE [186] policy gradient is calculated to update the parameters. The REINFORCE is also applied to KGpolicy and KGQR, and the same collaborative knowledge graphs are used in KGpolicy and KGAT. The embedding size for all baselines and our CERec is fixed as $d = 64$. Two

[4] https://github.com/salvacarrion/orange3-recommendation

graph convolutional layers with $\{32, 64\}$ output dimensions are performed for the graph learning in our model. All neural network-based (i.e., DNN, GCN) baselines also keep 2 layers. For MCRec, NeuACR and CaDSI, we use meta-paths with the path scheme of user-item-attribute-item to ensure the model compatibility, e.g., *user-book-author-book* for `Amazon-book` dataset. For counterfactual explanation, we fix the parameters of the trained counterfactual explanation model to produce one counterfactual item for each positive user-item interaction. The final explanation comprises item attributes that connect with the counterfactual item in the CKG but not with the positive user-item interaction. For explanation consistency evaluation, we construct ground-truth attribute sets using dynamic negative sampling (DNS) [255]. We first train an attribute-aware MF model by feeding positive user-item interactions and random negative item attributes. The DNS then uniformly draws one item attribute from the attribute space and feeds its latent factors into the MF model to predict the preference score. Item attributes with the highest scores are selected as negative samples to train the MF model recursively. Among multiple rounds of sampling, the DNS generates negative item attributes that match users' negative preferences. The hyper-parameters of all models are chosen by the grid search, including learning rate, L_2 norm regularization, discount factor γ, etc. The maximum epoch for all methods is set as 400, while an early stopping strategy is performed, i.e., if the loss stops increasing, then terminate the model training.

6.1.4.2 Counterfactual Enhanced Recommendation (RQ1)

In this section, we present the recommendation performance evaluation to answer the RQ1. At each iteration, the counterfactual explanation model in our CERec produces one counterfactual item for each positive user-item interaction to recursively train the recommendation model. Thus, we are interested in knowing whether incorporating counterfactual items could boost our recommendation performance. We present the recommendation performance of our CERec and baselines in Table 6.3, and here are our main findings.

- Our proposed CERec equipped with a counterfactual explanation model consistently outperforms all baselines across three datasets on both evaluation metrics. For example, CERec obtains 16.6%, 0.3% and 17.4% improvements for Recall@20, NDCG@20 and HR@20 respectively over the best baseline on `Last-FM` dataset. By designing a counterfactual explanation model, CERec is capable of exploring the high-order connectivity among the CKG to generate counterfactual items for rec-

Table 6.3: Recommendation evaluation.

Top-K Recommendation										
	Dataset	Last-FM			Amazon-book			Yelp2018		
Model	K	Recall@K	NDCG@K	HR@K	Recall@K	NDCG@K	HR@K	Recall@K	NDCG@K	HR@K
NeuMF	20	0.0651	0.0767	0.2921	0.0799	0.0611	0.1010	0.0279	0.0380	0.1008
	40	0.0807	0.0848	0.3409	0.1377	0.0769	0.2660	0.0357	0.0411	0.1241
	60	0.0913	0.0955	0.3788	0.1581	0.0888	0.3007	0.0389	0.0448	0.1552
	80	0.1005	0.1009	0.4267	0.1888	0.0974	0.3332	0.0411	0.0477	0.2008
MCRec	20	0.0770	0.0821	0.2811	0.0879	0.0666	0.1420	0.0327	0.0412	0.1234
	40	0.0825	0.0845	0.3124	0.1041	0.0712	0.2177	0.0338	0.0424	0.1406
	60	0.0919	0.0919	0.3888	0.1801	0.0844	0.2805	0.0429	0.0436	0.1721
	80	0.1112	0.0945	0.4282	0.2257	0.0957	0.3672	0.0457	0.0437	0.2205
KGpolicy	20	0.0761	0.0689	0.3197	0.1242	0.0684	0.2211	0.0557	0.0435	0.1528
	40	0.1018	0.0763	0.4091	0.1716	0.0805	0.2947	0.0649	0.0460	0.2588
	60	0.1179	0.0815	0.4639	0.2048	0.0879	0.3417	0.0832	0.0547	0.3300
	80	0.1302	0.0855	0.5006	0.2315	0.0935	0.3762	0.0883	0.0618	0.3841
KGQR	20	0.0821	0.0856	0.3162	0.1076	0.0787	0.2182	0.0417	0.0458	0.1621
	40	0.0932	0.0923	0.4229	0.1652	0.0891	0.2358	0.0456	0.0499	0.1899
	60	0.0999	0.0939	0.5228	0.1987	0.0912	0.3077	0.0512	0.0535	0.2172
	80	0.1132	0.0947	0.5323	0.2212	0.0942	0.3531	0.0557	0.0587	0.2566
KGAT	20	0.0870	0.0897	0.3292	0.0801	0.0791	0.2006	0.0444	0.0472	0.2341
	40	0.0962	0.0918	0.3762	0.1435	0.0812	0.2634	0.0469	0.0511	0.2666
	60	0.1083	0.0937	0.4515	0.1766	0.0922	0.3244	0.0501	0.0546	0.3015
	80	0.1232	0.0939	0.5464	0.2378	0.0985	0.3921	0.0544	0.0577	0.3655
CERec	20	**0.1015**	**0.0900**	**0.3867**	**0.1406**	**0.0803**	**0.2451**	**0.0650**	**0.0495**	**0.2515**
	40	**0.1304**	**0.0993**	**0.4801**	**0.1881**	**0.0926**	**0.3174**	**0.1025**	**0.0555**	**0.3549**
	60	**0.1478**	**0.1052**	**0.5295**	**0.2203**	**0.0999**	**0.3624**	**0.1317**	**0.0639**	**0.4218**
	80	**0.1603**	**0.1094**	**0.5628**	**0.2462**	**0.1054**	**0.3948**	**0.1572**	**0.0706**	**0.4739**
Improv.	20	+ 16.6%	+ 0.3%	+ 17.4%	+ 13.2%	+1.5%	+ 10.8%	+ 16.7%	+ 4.8%	+ 7.4%
	40	+ 28.0%	+ 7.5%	+ 13.5%	+ 9.6%	+ 3.9%	+ 6.5%	+ 57.9%	+ 8.6%	+ 33.1%
	60	+ 25.3%	+ 10.1%	+ 1.2%	+ 7.5%	+ 8.3%	+ 6.0%	+ 58.2%	+ 16.8%	+ 27.8%
	80	+ 23.1%	+ 8.4%	+ 3.0%	+ 3.5%	+ 7.0%	+ 0.6%	+ 78.0%	+ 14.2%	+ 23.3%

ommendation model training. This verifies the effectiveness of our counterfactual explanation model in producing high-quality counterfactual items that can boost the recommendation. Counterfactual items provide high-quality negative signals of user preference. By pairing one counterfactual item with one positive user-item interaction for training, our recommendation model learns to distinguish between positive items and negative items to generate more precise recommendations.

- By jointly analyzing the results across the three datasets, our CERec achieves the most significant improvements over baselines on the Yelp2018 dataset, e.g., 78.0%, 14.2%, and 23.3% improvements for Recall@80, NDCG@80, and HR@80, respectively. The Yelp2018 dataset records a large number of inactive users that have few interactions with items. It is well acknowledged that inferring user pref-

erence for inactive users is challenging and limits the performance of recommender systems [215]. However, our CERec can still achieve favorable recommendation performance on the `Yelp2018`. We attribute the improvements of our CERec to the augmenting of counterfactual items that could infer negative user preferences for inactive users and thus boost the recommendation.

- Reinforcement learning (RL) endows recommendation models with improved recommendation performance. In particular, in Table 6.3, the recommendation performance of RL-based baselines (i.e., KGpolicy and KGQR) beats non-RL-based baselines (i.e., NeuMF, MCRec and KGAT) in most cases. For instance, KGpolicy achieves 0.1018 Recall@40 while NeuMF only has Recall@40 of 0.0807. Unlike static baselines without using RL, RL-based baselines could handle dynamic user-item interactions with the recommendation environments and thus better capture users' preference shifts. This observation suggests that incorporating reinforcement learning helps improve recommendation performance by considering the dynamic nature of user preferences. Our CERec also harnesses the power of reinforcement learning to adapt to dynamic user preferences when inferring counterfactual items. Those counterfactual items identified by our CERec are proper to explain users' ever-changing preferences. Ultimately, CERec enhances the recommendation model by augmenting these high-quality counterfactual items, resulting in superior recommendation performance compared to all other baselines.

- Among the knowledge-aware models, our CERec consistently outperforms MCRec, KGpolicy, KGQR, and KGAT. This is mostly because these knowledge-aware baselines might not fully utilize the item knowledge by lacking the counterfactual reasoning ability as in our CERec. All knowledge-aware baselines employ various attention mechanisms to capture users' preferences on item knowledge w.r.t. different item attributes. However, they fail to consider the underlying mechanism that really triggered users' interactions. On the contrary, our CERec learns to discover the item attributes that cause the change of users' interactions, thus resulting in a promoted recommendation performance.

6.1.4.3 Counterfactual Explanation Quality (RQ2)

To answer RQ2, we evaluate the explainability of our framework by reporting the quantitative results of explanation evaluation metrics. Then, we discuss the interpretability of our generated counterfactual explanations by distilling their real-world semantics.

Table 6.4: Explainability evaluation w.r.t the consistency.

Explanation Consistency									
Dataset / Model	Last-FM			Amazon-book			Yelp2018		
	Precision%	Recall%	F_1%	Precision%	Recall%	F_1%	Precision%	Recall%	F_1%
RDExp	0.0214 ± 0.01	0.0191 ± 0.00	0.0198 ± 0.00	0.0196 ± 0.00	0.0090 ± 0.00	0.0117 ± 0.00	0.4595 ± 0.01	0.0987 ± 0.01	0.1320 ± 0.03
NeuACF	3.7295 ± 0.90	0.5843 ± 0.04	0.6036 ± 1.32	4.7336 ± 1.21	0.6171 ± 0.06	1.0226 ± 0.08	8.5603 ± 1.51	8.2619 ± 1.47	7.0589 ± 1.38
CaDSI	12.2143 ± 0.96	2.4371 ± 0.89	3.2667 ± 1.98	13.2632 ± 1.98	1.0897 ± 0.09	2.8721 ± 0.06	17.7322 ± 2.07	19.0801 ± 2.38	18.3096 ± 2.60
CERec	**21.8394** ± 1.21	**3.8957** ± 1.62	**6.5234** ± 0.89	**23.8969** ± 2.84	**3.1020** ± 0.92	**5.1411** ± 0.83	**45.5013** ± 3.11	**41.5931** ± 3.74	**37.2509** ± 3.09

Quantitative Results. We report the Precision, Recall and F_1 (*cf.* Eq. (6.17)) of explanations generated by our CERec and baseline models in Table 6.4 to test the explanation consistency. The three evaluation metrics reflect what extent the item attributes in explanations are appropriate to explain users' preferences. By analyzing Table 6.4, we have several observations. Firstly, the RDExp method performs very poorly on both Precision, Recall, and F_1. The RDExp generates explanations by randomly sampling item attributes from the knowledge graph provided. The poor performance of RDExp shows that randomly choosing item attributes as explanations can barely reveal the reasons for recommendations. This demonstrates that high-quality explanations require the appropriate choice of item attributes. Secondly, the counterfactual explanations generated by our CERec consistently show superiority against all baselines in finding item attributes that are consistent with users' preferences. This indicates that our CERec achieves superior explainability and can generate more relevant attribute-based explanations that truly match user preference. Unlike NeuACF and CaDSI, which generate explanations by selecting a fixed number of item attributes indicated by their importance scores, our CERec searches a minimal set of item attributes that would flip the recommendation result. We hence conclude that inferring item attributes with minimal changes is more appropriate to generate explanations, since they reflect simple but essential attributes that directly cause user preference changes. This further sheds light on the importance of conducting counterfactual explanations for recommendations.

Interpretability of Counterfactual Explanations. We present a case study on the interpretability of counterfactual paths that root at a positive user-item interaction and end at its counterfactual item. Each freebase entity among paths are mapped into real-world entities. We show the real-world cases in Figure 6.3 and give our observations in the following.

The first example (Case 1) comes from the Last-FM dataset, where user u_{17724} positively interacted with "Perfect 10" in his listening record. Our CERec picks the "Loving you is killing me" as the counterfactual item, which shares three overlapping item attributes with the "Perfect 10". Conventional explainable recommendation models would

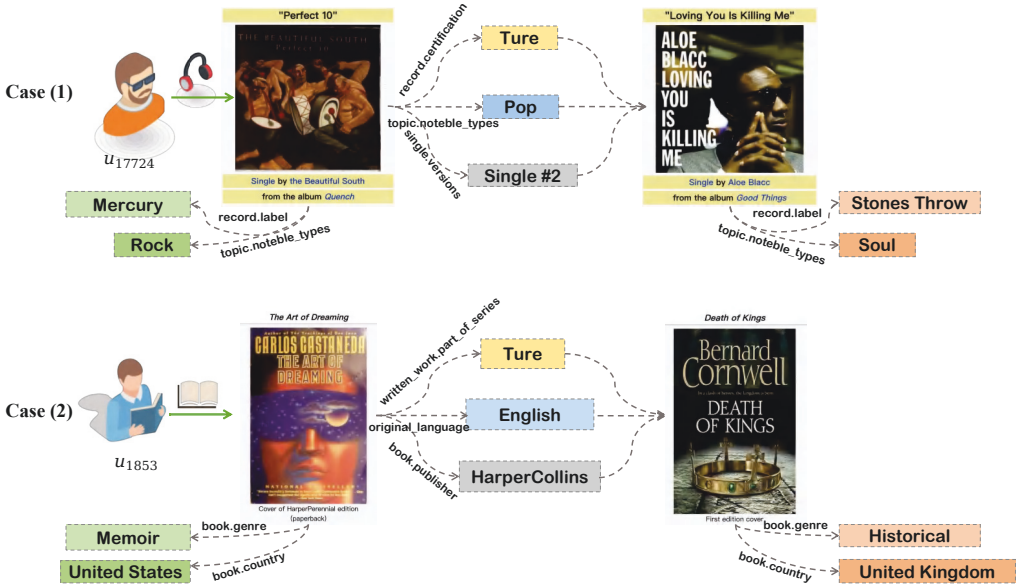


Figure 6.3: Real cases of counterfactual reasoning paths.

pick these three common item attributes to generate explanations as "you like Perfect 10 is because you like pop music that was certificated for its single#2 version". However, this explanation cannot answer why the "Loving you is killing me" with the same item attributes is actually being removed from the user's recommendation list. By contrast, our CERec captures the counterfactual item attributes (i.e., *Stones Throw* and *Soul*) and generates the counterfactual explanation as "if Perfect 10 has the label as *Stones Throw* and the type as *Soul*, then it will not be recommended anymore." This counterfactual explanation provides us with deeper insights into users' real preferences. For example, analyzing the attribute differences, we find that although "Perfect 10" shares the same type *Pop* as "Loving you is killing me", it is actually the *Rock* music published by *Mercury*. On the contrary, the "Loving you is killing me" that holds the opposite music type *Soul* and different polisher would potentially be disliked by the same user. Analogously, in Case 2 from the `Amazon-book`, our CERec picks the "Death of Kings" as the counterfactual item for the positive interaction between u_{1853} and "The Art of Dreaming", and uses item attributes *Historical* and *United Kingdom* as the counterfactual explanation to reflect u_{1853}'s preferences on book genre and country. These counterfactual explanations are more rational than conventional explanations, since they discard tedious associations of item attributes and expand explanations to counterfactual attributes.

6.1.4.4 Ablation Study on CERec (RQ3)

We have three contributing components in our CERec (*cf.* Figure 6.2): graph learning module (GLM), counterfactual path sampler (CPS) and reinforcement learning agent (RLA). We explore how replacing key designs in GLM, CPS and RLA impact CERec's performance. We present evaluation results for `Amazon-book` and `Yelp2018` on both Top-K recommendation and explanation consistency evaluation tasks. Since the results on `Last-FM` and `Amazon-book` are similar, we include only the results of `Amazon-book` in this study.

Table 6.5: Ablation study on graph learning module: impact of graph learning methods.

Variants	*Top-K Recommendation*		*Explanation Consistency*	
	Recall@20	NDCG@20	Precision%	Recall%
	`Amazon-book`			
CERec-GCN	0.1392 (-1.0%)	0.0796 (-0.8%)	21.7364 (-9.0%)	2.8790 (-7.1%)
CERec-GNN	0.1323 (-5.9%)	0.0742 (-7.5%)	21.6832 (-9.2%)	2.7793 (-10.3%)
CERec-SGC	0.1358 (-3.9%)	0.0779 (-2.8%)	19.0863 (-19.9%)	2.5837 (-16.7%)
CERec-LGC	0.1301 (-7.6%)	0.0752 (-6.8%)	19.4328 (-18.6%)	2.5602 (-17.4%)
CERec-GAT	0.1394 (-0.8%)	0.0782 (-2.1%)	21.8298 (-8.4%)	2.9074 (-6.6%)
CERec	**0.1406**	**0.0803**	**23.8969**	**3.1020**
	`Yelp2018`			
CERec-GCN	0.0613 (-5.6%)	0.0401 (-19.1%)	40.0807 (-11.8%)	36.6742 (-11.8%)
CERec-GNN	0.0644 (-0.9%)	0.0426 (-13.9%)	40.9344 (-10.0%)	37.0901 (-10.8%)
CERec-SGC	0.0564 (-13.2%)	0.0366 (-26.2%)	39.0927 (-14.0%)	35.6321 (-14.3%)
CERec-LGC	0.0563 (-13.3%)	0.0363 (-26.6%)	38.8902 (-14.5%)	35.0966 (-15.6%)
CERec-GAT	0.0628 (-3.3%)	0.0439 (-11.3%)	41.0843 (-9.6%)	38.0871 (-8.4%)
CERec	**0.0650**	**0.0495**	**45.5013**	**41.5931**

Impact of Graph Learning Methods. Our graph learning module (GLM) employs GraphSAGE [74] to quantify complex graph topology and node features of a given collaborative knowledge graph as graph embeddings. To investigate how different graph learning methods affect CERec's performance, we replace GraphSAGE in our GLM with other graph learning methods, including Graph Convolutional Network (GCN) [103], Graph Neural Network (GNN) [140], Simple Graph Convolutional Network (SGC) [229], Light Graph Convolution Network (LGC) [80] and Graph Attentional Network (GAT) [196]. Those graph learning methods are both state-of-the-art benchmarks in graph representation tasks [235]. Table 6.5 shows the experimental results, wherein CERec-GCN indicates the CERec trained with Graph Convolutional Network (GCN); similar notations for other variants. Following the parameter settings, for fair comparisons, we keep the

same embedding size (i.e., $d = 64$) for all variants and the original CERec in Table 6.5. Besides, two graph layers with $\{32, 64\}$ output dimensions are performed for all variants and CERec. Upon analyzing Table 6.5, we have several observations.

- Our CERec with a GraphSAGE-based GLM produces the best recommendation and explanation performance compared with CERec with other graph learning methods. This verifies the effectiveness of using GraphSAGE in our CERec. GraphSAGE exhibits inductive learning, in contrast to GCN and GNN, which employ transductive learning. The inductive learning nature of GraphSAGE helps our CERec to generalize well to unseen nodes, as inductive learning uses partial graph Laplacian to allow better inferences of unseen nodes compared with transductive learning. Besides, compared with SGC and LGC, GraphSAGE captures higher-order neighbor dependencies and layer dependencies to model complex graph structures. In return, CERec with a GraphSAGE-based GLM could harness complex dependencies in graphs to enhance the quality of explanation learning. Because of these two advantages, GraphSAGE generates superior graph embeddings, which in turn improves counterfactual explanation learning of CERec.
- Graph learning methods largely impact model explainability, as evidenced by the severely degraded explanation consistency of both variants in Table 6.5. This suggests that explanations heavily rely on the quality of embeddings learned by the graph learning methods. The reason is that graph embeddings provide essential semantics for explanation learning, while inaccurate or false semantics can lead to varying or contradictory explanations for recommendations. For example, on `Last-FM` dataset, we observe that our CERec with a GraphSAGE-based GLM explains user u_{17724}'s preference by the music type "Soul", which aligns with the actual user preference recorded in the dataset. In contrast, CERec with a GCN-based GLM uses the music singer "Aloe Blacc" as the explanation, which is contradictory with u_{17724}'s preference. We thus conclude that ensuring the reliability of graph embeddings through appropriate graph learning methods is crucial for maintaining high explanation consistency. GraphSAGE, with its ability to capture accurate semantics, shows to be a suitable choice for enhancing the learning of counterfactual explanations.

Impact of Attention Scores. The attention mechanism is pivotal in the counterfactual path sampler (CPS) to guide the search for counterfactual paths within the two-step path sampling. Hence, we investigate the effectiveness of the attention mechanism, i.e., how

Table 6.6: Ablation study on counterfactual path sampler: impact of attention scores.

Variants	*Top-K Recommendation*		*Explanation Consistency*	
	Recall@20	NDCG@20	Precision%	Recall%
	`Amazon-book`			
Unif(α_1)	0.1390 (-1.1%)	0.0790 (-1.6%)	21.8847 (-8.4%)	2.7903 (-10.0%)
Unif(α_2)	0.1193 (-15.1%)	0.0674 (-16.0%)	19.0109 (-20.4%)	2.4380 (-21.4%)
CERec	**0.1406**	**0.0803**	**23.8969**	**3.1020**
	`Yelp2018`			
Unif(α_1)	0.0602 (-7.9%)	0.0443 (-11.7%)	41.3869 (-9.9%)	39.0843 (-6.4%)
Unif(α_2)	0.0520 (-25.0%)	0.0334 (-48.3%)	35.6721 (-27.5%)	33.5833(-23.8%)
CERec	**0.0650**	**0.0495**	**45.5013**	**41.5931**

attention scores facilitate searching for informative counterfactual paths. In particular, two attention scores α_1 and α_2 are calculated respectively through the attention mechanism on source and target node embeddings. We thus do the ablation study by fixing one attention score as uniform probability while keeping the other score unchanged. We present the experimental results in Table 6.6, wherein the notation Unif(α_1) signifies that α_1 is sampled using a uniform probability $U(0,1)$; the same applies for Unif(α_2). Note that we do not consider setting both $\alpha_1 \sim U(0,1)$ and $\alpha_2 \sim U(0,1)$, as this would lead to the zero gradients of our CPS. Analyzing Table 6.6, we observe that both Unif(α_1) and Unif(α_2) downgrade the recommendation and explainability performance of CERec. We thus conclude that both attention scores are important for our CERec to find appropriate counterfactual paths to better serve the explanation learning. By leveraging these attention scores, our CERec can effectively guide the search for informative counterfactual paths toward candidate counterfactual items, ultimately improving the quality of counterfactual explanations. Moreover, Unif(α_2) exhibits relatively larger decrease percentages compared with Unif(α_1). For instance, Unif(α_1) downgrades CERec's performance by a Recall@20 of 7.9% while Unif(α_2) downgrades 25.0% Recall@20 on `Yelp2018`. This is reasonable as Unif(α_2) aligns uniformly distributed attention scores at the second sampling step, which ignores the influence of α_1 calculated during the first sampling step. This observation further highlights the superiority of applying the attention mechanism at each step of the path sampling.

Impact of Reward Functions. The reward function is the core of the reinforcement learning agent as it guides policy explorations. To verify the influence of reward functions, we consider two variants of our proposed counterfactual reward in Eq. (6.14). In particular, Eq. (6.14) incorporates two criteria: *Rationality* [197] and *Similarity* [47];

Table 6.7: Ablation study on reinforcement learning agent: impact of reward functions.

Variants	*Top-K Recommendation*		*Explanation Consistency*	
	Recall@20	NDCG@20	Precision%	Recall%
	`Amazon-book`			
R-reward	0.1417 (-0.7%)	0.0802 (-0.0%)	15.8732 (-33.5%)	1.8533 (-40.1%)
S-reward	0.1225 (-14.7%)	0.0733 (-9.5%)	10.6273 (-55.5%)	0.9231 (-66.3%)
CERec	**0.1406**	**0.0803**	**23.8969**	**3.1020**
	`Yelp2018`			
R-reward	0.0632 (-2.8%)	0.0419 (-18.1%)	31.0579 (-31.6%)	29.7834 (-28.3%)
S-reward	0.0556 (-16.8%)	0.0367 (-34.9%)	20.0731 (-55.8%)	25.6565 (-38.1%)
CERec	**0.0650**	**0.0495**	**45.5013**	**41.5931**

we employ only *Rationality* to build the variant R-reward, while using *Similarity* to construct the variant S-reward. Formally, R-reward is given by,

$$r(s_t, a_t) = \begin{cases} 1, \text{ if } \mathbb{P}_u(e_t) - \mathbb{P}_u(e_{t+1}) \geq \epsilon \\ 0, \text{ otherwise} \end{cases} \tag{6.18}$$

where $r(s_t, a_t)$ is the reward calculated for action a_t at state s_t. S-reward is defined as,

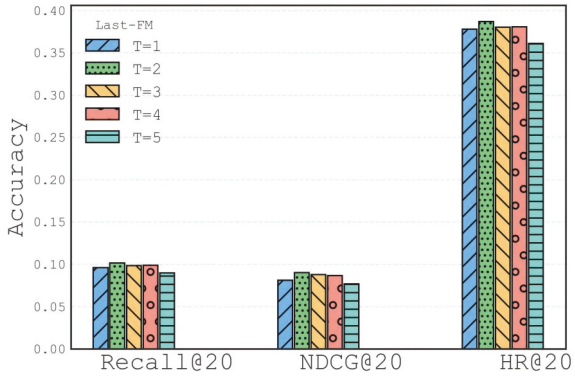
$$r(s_t, a_t) = \cos(\mathrm{h}_{e_t}, \mathrm{h}_{e_{t+1}}) \tag{6.19}$$

The results are shown in Table 6.7. We observe that using R-reward and S-reward degrades CERec's performance, especially for the explanation consistency performance. This suggests that both R-reward and S-reward misguide policy learning to learn irrelevant counterfactual explanations that are not consistent with users' preferences. This is reasonable as excluding either *Rationality* or *Similarity* would cause the learning process to deviate from the desired objectives of counterfactual explanations, resulting in inferior performance. In contrast, our proposed counterfactual reward considers both *Rationality* and *Similarity*, aligning with the definition of counterfactual explanation to guide accurate policy learning. Moreover, using S-reward leads to relatively more severe model degradation compared with using R-reward. S-reward only encourages the candidate counterfactual node to be similar to the target node, which, however, does not verify the *Rationality*, i.e., whether the candidate node crosses the decision boundary. We thus conclude that ensuring *Rationality* is essential for learning counterfactual explanations.

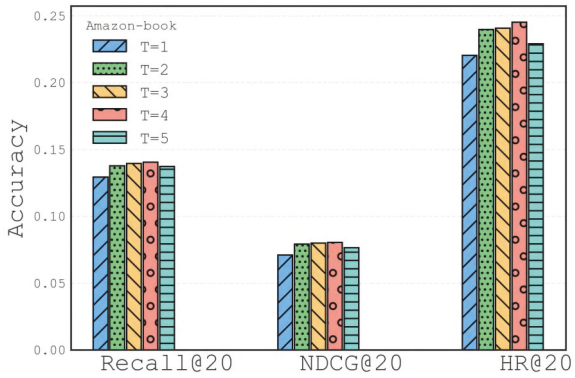
6.1.4.5 Parameter Analysis and Computation Costs (RQ4, RQ5)

We first study how the reinforcement depth T in Eq. (6.15) affects the recommendation performance. We then investigate how the training epoch impacts the stability of our

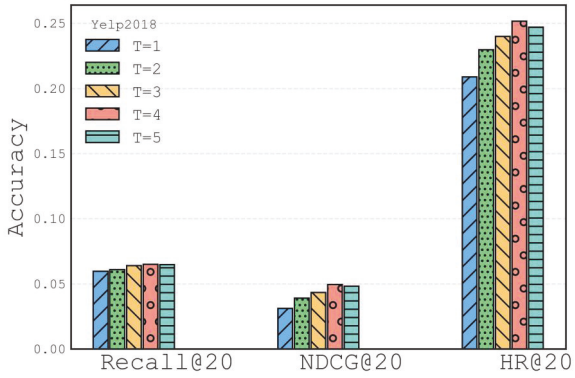
recommendation model. We also give the computation costs of reinforcement learning-based baselines and our method.



(a) The impact of T on Last-FM.



(b) The impact of T on Amazon-book.



(c) The impact of T on Yelp2018.

Figure 6.4: Impact of reinforcement depth T on three datasets.

Impact of Reinforcement Depth. The reinforcement depth T in Eq. (6.15) determines the search space, with $T = 1$ denoting that at most 1-hop neighbors of starting entities are visited as proposals for policy learning. We search the number of T in $\{1, 2, 3, 4, 5\}$

and report the recommendation performance of our model on both datasets in Figure 6.4 (a) (b) (c). We have the following observations. Firstly, our CERec with $T = 4$ yields the best performance on the Amazon-book and Yelp2018 dataset, while $T = 2$ gives the best results on the Last-FM. The Amazon-book and Yelp2018 are presented with 0.048% and 0.057% density and are much sparser compared with the Last-FM with 0.26% density. That means more inactive users with few item interactions are recorded in the Amazon-book and Yelp2018. To achieve the optimal recommendation performance, the Amazon-book and Yelp2018 require larger reinforcement depth (i.e., $T = 4$) compared with the Last-FM (i.e., $T = 2$). This is because diverse counterfactual items are retrieved by increasing the reinforcement depth to help the recommendation model achieve optimal performance on the two datasets. This finding indicates that augmenting diverse counterfactual items could provide precise recommendations for inactive users to enhance the recommendation. Counterfactual items offer high-quality negative signals for user preferences, and thus could help filter out negative items when providing recommendations for inactive users. Secondly, increasing the reinforcement depth enhances the recommendation performance before reaching the peaks on both datasets. We attribute such consistent improvements to the improved diversity of counterfactual items. This is because higher-hop item neighbors naturally cover more items beyond those unexposed but are actually counterfactual ones than lower-hop neighbors. Thirdly, after peaks, increasing reinforcement depth leads to downgraded performance. This is because performing too many counterfactual item explorations introduces less relevant items that may bias the recommendation results.

Reward Gradient. To evaluate the stability and robustness of our model, we plot the cumulative rewards while training our model on the three datasets in Figure 6.5. Here are our observations. First, our counterfactual explanation model gains stable cumulative rewards on both datasets along with training, i.e., the cumulative rewards increase as the epoch increases and finally reach stable states without suffering any drastic fluctuations. This indicates that our counterfactual explanation model enjoys stable training without losing reward gradients (e.g., gradients varnishing). This further verifies the rationality and robustness of our proposed model. Second, the rates of reward convergences are different across different datasets. For example, the cumulative rewards on Amazon-book and Yelp2018 start to increase drastically at the beginning and converge at around epoch 40, while the counterpart on Last-FM first converges slowly and then becomes stable at around epoch 120. This indicates that our model on Amazon-book and Yelp2018 can quickly reach stable states using a small number of iterations; even Amazon-book and

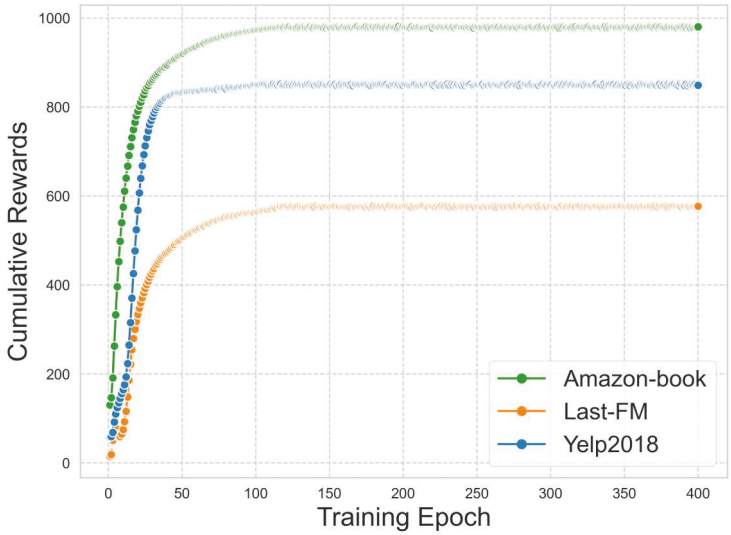


Figure 6.5: Learning curves of cumulative rewards w.r.t. training epoch.

`Yelp2018` are much more sparse than `Last-FM`. This is because we use counterfactual items during training, which contain negative user preference signals to assist the decision-making for inactive users. Besides, we explore higher-order connectivity among the CKG as side information for training, thus enhancing model robustness facing sparse datasets.

Computation Costs. We evaluate the running time of reinforcement learning-based baselines, including KGpolicy, KGQR, and KGAT, on the largest `Amazon-book` dataset. For the average case, the corresponding results are 232s, 379s, and 279s per epoch, respectively. Our CERec algorithm runs in 284s per epoch on average. Thus, CERec has a comparable cost to other RL-based baselines. Besides, our CERec runs in 273s per epoch in the best case and 291s per epoch in the worst case, which are also comparable costs to other RL-based baselines. We also observed that the training time of CERec increases by 42.78s per epoch on `Amazon-book` after using the counterfactual explanation model. Overall, as our counterfactual explanation model delivers explanations with superior increased explainability, e.g., 23.8969 F_1 score on `Amazon-book` (*cf.* Table 6.4), we consider the increased cost by 42.78s as a reasonable trade-off.

6.1.5 Summary

This research work proposes a reinforcement learning-based counterfactual explainable recommendation framework over a CKG. The proposed CERec model can generate attribute-based counterfactual explanations while providing precise recommendations. We design a counterfactual explanation model as a reinforcement learning agent to discover high-quality counterfactual explanations. The counterfactual explanation model takes paths sampled from our counterfactual path sampler as actions to optimize an explanation policy. By maximizing the counterfactual rewards of the deployed actions, the explanation policy is learned to generate high-quality counterfactual items. In addition, we reduce the vast action space by utilizing attention mechanisms in our path sampler to yield effective paths from the CKG. Finally, the learned explanation policy generates attribute-based counterfactual explanations for recommendations. We deploy the explanation policy to a recommendation model to enhance the recommendation. Extensive explainability and recommendation evaluations on three large-scale datasets demonstrate CERec's abilities to improve the recommendation and provide counterfactual explanations consistent with user preferences.

6.2 Counterfactual Fairness Explanation

6.2.1 Overview

Research Question. A fundamental problem for fairness-aware recommendations is understanding why a model is unfair, i.e., "*what causes unfair recommendation results*?". Fairness explanation, as an important aspect of fairness research, would enhance the design of fairness-aware recommendation approaches by tracking unfair factors for model curation. This research work focuses on the item exposure unfairness problem in recommendations, and targets generating high-quality explanations that uncover unfair factors to promote the fair allocation of user-preferred but less exposed items.

Research Objective. This research aims to generate counterfactual explanations for fairness rather than conventional explanations. Counterfactual explanations learn unfair factors as the "minimal" change of the input features that guarantee the reduction of unfairness [55, 197]. They provide "what-if" explanations to determine the most vital and essential (i.e., minimal) factors that change model fairness [197]. Unlike conventional explainable methods with high complexity, counterfactual explanations have the advantage of being minimal w.r.t. the generated explanations and are faithful to model

fairness changes. Unfortunately, there is only one counterfactual explainable method for recommendation fairness. Ge et al. [56] explain which item feature changes the item exposure fairness of the feature-based recommendation models. They perturb feature scores within pre-defined user-aspect and item-aspect matrices and feed the perturbed matrices into a recommendation model. Those perturbed features that change the fairness disparity are considered fairness explanations. This counterfactual explainable method suffers from the following limitations: 1) It conducts optimization on feature matrices, which require vector-wise perturbations. However, when dealing with high-dimensional features, vector-wise perturbations may not significantly change perturbed matrices as feature matrices would become large-scale. In such cases, the effectiveness of perturbing feature matrices to identify sensitive features can be limited. 2) It is primarily designed for feature-based models and is not well-suited for handling discrete attributes in attribute-aware recommendation models. Including context-based filtering [174, 176] and knowledge-aware systems [88, 204], modern recommendation models are largely attribute-aware and rely on explicit attributes to generate recommendations. Those attributes are user and item demographics, which are generated through data tagging on discrete attributes, e.g., genre, language, and location. However, the feature-level optimization in [56] can only deal with continuous values and cannot be directly applied to discrete attributes. For example, assigning a continuous value, such as *gender*=0.19, to the discrete *gender* attribute is impractical in constructing explanations and provides no valuable clue to improve the explanation. Consequently, feature-level optimizations have limited capability in handling discrete attributes that are frequently encountered in recommendation scenarios. To tackle the limitations of the existing work, this work focuses on exploring attribute-level counterfactual explanations for attribute-aware recommendation models. The research objective is to answer: "*what the fairness would be if a minimal set of attributes had been different*?".

Motivations. Driven by recent successes in Heterogeneous Information Networks (HINs)-enhanced recommendations [88, 174, 176, 206, 208, 216], this work proposes to leverage real-world attributes from a HIN for counterfactual reasoning when dealing with discrete attributes. In contrast to value-based features, real-world attributes residing in HINs are represented as discrete nodes, with edges denoting their connections. By utilizing attributes from HINs, we can overcome the limitation of feature-level optimizations to directly measure whether the removal of specific attributes changes the model's fairness. We use a toy example in Figure 6.6 to illustrate how HIN assists in learning attribute-level counterfactual explanations. Given a HIN carrying attributes

of recommended items i_1, i_2 and users u_1, u_2, we want to know why i_1 and i_2 cause discrimination in recommendation results. The counterfactual explanation performs "what-if" reasoning by removing user and item attributes from the HIN and checking the fairness disparity of recommendation results. Both E_1 and E_2 are valid candidate explanations since they reduce (i.e., ↓) fairness disparities of recommendations (i.e., i_3, i_4) from 0.90 to 0.19. To determine which attributes are the primary reason for unfairness, the counterfactual explanation will use the minimal attribute changes, i.e., E_1, instead of utilizing attribute combinations in E_2. Thus, we could infer E_1 is the most vital reason for model unfairness. Besides, since a counterfactual explanation E_1 is minimal, it only reveals the essential attributes (i.e., "Female") that effectively explain unfairness while discarding the less accurate explanations.

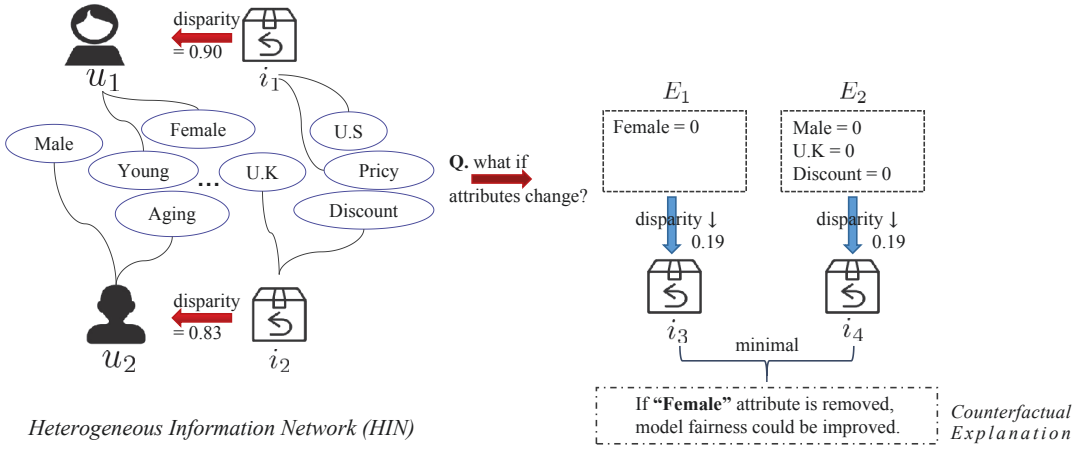


Figure 6.6: Toy example of inferring the attribute-level counterfactual explanation for fairness with a HIN.

The Proposed Approach. This work proposes the CFairER model (Counterfactual Explanation for Fairness) to find optimal attribute-level counterfactual explanations from a given HIN. Particularly, we focus on generating explanations for item exposure unfairness to promote the fair allocation of user-preferred but less exposed items. Note that the proposed approach maintains generalizability and can be integrated with different recommendation models to mitigate item exposure unfairness. Specifically, we use an off-policy reinforcement learning agent in CFairER to optimize a fairness explanation policy by uniformly exploring candidate counterfactuals from a given HIN. We devise attentive action pruning over the HIN to reduce the search space for reinforcement learning. We propose a counterfactual reward to ensure the rationality of the learned fairness explanations. Finally, our CFairER optimizes the explanation policy using an

unbiased counterfactual risk minimization objective, resulting in accurate attribute-level counterfactual explanations for fairness.

Contributions. The contributions of this work are:

- We make the first attempt to leverage rich attributes in a Heterogeneous Information Network to offer attribute-level counterfactual explanations for recommendation fairness.
- We propose an off-policy learning framework to identify optimal counterfactual explanations, which is guided by attentive action pruning to reduce the search space.
- We devise a counterfactual risk minimization for off-policy correction to achieve unbiased policy optimization.
- Comprehensive experiments show the superiority of our method in generating trustworthy explanations for fairness while preserving satisfactory recommendation performance.

6.2.2 Fairness Definitions and Metrics

We focus on the item exposure (un)fairness in recommendations. We first split items in historical user-item interactions into head-tailed group G_0 the long-tailed group G_1 [5]. Note that items from the head-tailed group refer to the top popular items that dominate historical user-item interactions. Following previous works [54, 56], we use demographic parity (DP) and exact-K (EK) defined on item subgroups to measure whether a recommendation result is fair. In particular, DP requires that each item has the same likelihood of being classified into G_0 and G_1. EK regulates the item exposure across each subgroup to remain statistically indistinguishable from a given maximum α. By evaluating the deviation of recommendation results from the two fairness criteria, we can calculate the fairness disparity, i.e., to what extent the recommendation model is unfair. Formally, giving a recommendation result $H_{u,K}$, the fairness disparity $\Delta(H_{u,K})$ of $H_{u,K}$ is:

$$
\begin{aligned}
&\Delta(H_{u,K}) = (1-\lambda)|\Psi_{DP}| + \lambda|\Psi_{EK}|, \\
&\Psi_{DP} = |G_1| \cdot \text{ Exposure } \left(G_0 \mid H_{u,K}\right) - |G_0| \cdot \text{ Exposure } \left(G_1 \mid H_{u,K}\right), \\
&\Psi_{EK} = \alpha \cdot \text{ Exposure } \left(G_0 \mid H_{u,K}\right) - \text{ Exposure } \left(G_1 \mid H_{u,K}\right)
\end{aligned} \tag{6.20}
$$

[5] Following [56], we consider the top 20% items with the most frequent interactions with users as G_0, while the remaining 80% belongs to G_1.

where K is the recommendation list length of $H_{u,K}$, $\Delta(\cdot)$ is the fairness disparity metric that quantifies model fairness status. λ is the trade-off parameter between DP and EK, where λ is chosen from $\{0,1\}$ and $\lambda = 0$ representing DP is considered as the fairness measurement while $\lambda = 1$ denoting EK is used as the fairness measurement. The definition of $\Delta(\cdot)$ enables us to choose the optimal disparity definition for a variety of recommendation tasks, e.g., music and movie recommendations. Exposure$\left(G_j \mid H_{u,K}\right)$ is the item exposure number of $H_{u,K}$ within G_j w.r.t. $j \in \{0,1\}$.

6.2.3 Counterfactual Explanation for Fairness

This work aims to generate attribute-level counterfactual explanations for item exposure fairness. In particular, we aim to find the "minimal" changes in attributes that reduce the fairness disparity (*cf.* Eq. (6.20)) of item exposure. Formally, given historical user-item interaction $\boldsymbol{Y}$, and user attribute set $\mathcal{V}_U$ and item attribute set $\mathcal{V}_I$ extracted from an external Heterogeneous Information Network (HIN) $\mathcal{G}$. Suppose there exists a recommendation model that produces the recommendation result $H_{u,K}$ for user u. Given all user-item pairs (u,v) in $H_{u,K}$, our goal is to find a minimal attributes set $\mathcal{V}^* \subseteq \{\{e_u, e_v\} \mid (u,e_u),(v,e_v) \in \mathcal{E}', e_u \in \mathcal{V}_U, e_v \in \mathcal{V}_I\}$. Each attribute in $\mathcal{V}^*$ is an attribute entity from HIN $\mathcal{G}$, e.g., user's gender and item's genre. With a minimal set of $\mathcal{V}^*$, the counterfactual reasoning pursues to answer: what the fairness disparity would be, if $\mathcal{V}^*$ is applied to the recommendation model. $\mathcal{V}^*$ is recognized as a valid *counterfactual explanation for fairness*, if after applied $\mathcal{V}^*$, the fairness disparity of the intervened recommendation result $\Delta(H^{cf}_{u,K})$ reduced compared with original $\Delta(H_{u,K})$. In addition, $\mathcal{V}^*$ is *minimal* such that there is no smaller set $\mathcal{V}^{*'} \in \mathcal{G}$ satisfying $|\mathcal{V}^{*'}| < |\mathcal{V}^*|$ when $\mathcal{V}^{*'}$ is also valid.

6.2.4 Methodology

6.2.4.1 The CFairER Framework

We now introduce our Counterfactual Explanation for Fairness (CFairER). As shown in Figure 6.7, CFairER devises three major components: 1) graph representation module embeds users, items, and attributes among HIN as embedding vectors; 2) recommendation model learns user and item latent factors to produce recommendation results and 3) our proposed counterfactual fairness explanation (CFE) model assisted by the graph representation module and the recommendation model to conduct counterfactual reasoning. This section discusses how the CFE model collaborates with the other two components,

Algorithm 2: CFairER: Counterfactual Fairness Explanation for Recommendation

Input : HIN $\mathcal{G}$; user set $\mathcal{U}$, item set $\mathcal{I}$, user attribute set $\mathcal{V}_U$, item attribute set $\mathcal{V}_I$; observed user-item interaction matrix $\boldsymbol{Y}$; model parameters Θ;

Output : Counterfactual fairness explanation and fair recommendation results

Initialize Θ with uniform sampling;

Step 1: Graph Representation Learning

Initialize embedding vectors for users, items, and attributes;

for *each node* $v \in \mathcal{V}$ **do**

  Compute user embedding $\mathbf{h}_u$, item embedding $\mathbf{h}_v$, user attribute embedding $\mathbf{e}_u$ and item attribute embedding $\mathbf{e}_v$ using message passing among HIN;

end

Step 2: Recommendation Model Learning

for *each user* $u \in \mathcal{U}$ **do**

  for *each item* $i \in \mathcal{I}$ **do**

    Compute latent user and item factors: $\boldsymbol{U}_u$ and $\boldsymbol{V}_v$;

    Compute predicted interaction score: $\hat{y}_{ui} = f(\boldsymbol{U}_u, \boldsymbol{V}_v; \Theta)$;

    Compute Top-K recommendation lists $H_{u,K}$;

  end

end

Step 3: Counterfactual Fairness Explanation (CFE)

Initialize explanation policy π_E;

for *each user-item pair* (u, i) **do**

  // State Representation Learning

  Compute state s_t using $\mathbf{h}_u, \mathbf{h}_i, \mathbf{e}_u, \mathbf{e}_i$;

  // Attentive Action Selection

  Select action a_t using explanation policy $\pi_E(s_t) \rightarrow a_t$;

  // Counterfactual Intervention and Fairness Evaluation

  Generate intervened recommendation outcome $H^{cf}_{u,K}$ by fusing attribute embeddings with user and item latent factors;

  Compute fairness disparity change: $\Delta_{\text{fairness}} = H^{cf}_{u,K} - H_{u,K}$;

  // Reward Calculation and Policy Optimization

  Compute reward $r(s_t, a_t)$ based on fairness disparity change;

  Optimize explanation policy π_E using $\nabla_\Theta \widehat{R}(\pi_E)$.

end

Return Optimized fairness-aware explanation policy π_E.

and then introduces the graph representation module and the recommendation model. We will elaborate on our proposed CFE model in the next section.

Counterfactual Fairness Explanation Model. As shown in Figure 6.7, our CFE model uses off-policy learning, in which an explanation policy π_E is optimized to produce attribute-level counterfactual explanations for fairness. At each state s_t, π_E produces

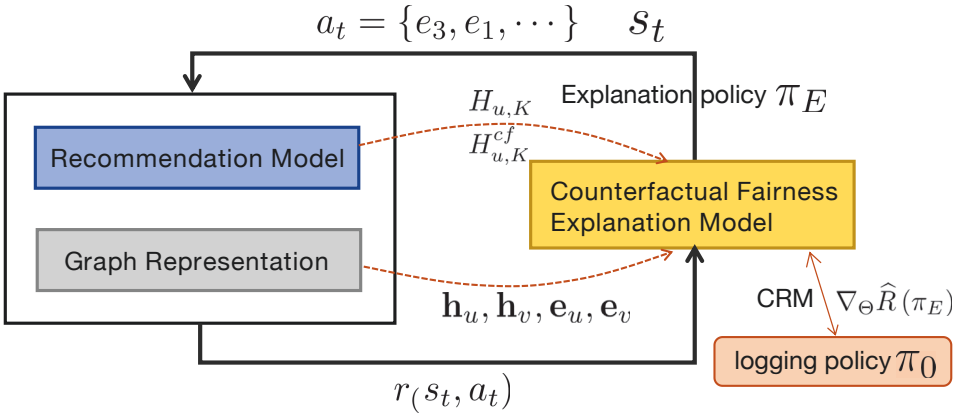


Figure 6.7: The proposed CFairER framework.

actions a_t absorbing user and item attributes as potential counterfactual explanations. These actions are committed to the recommendation model and graph representation module to produce the reward $r(s_t,a_t)$ for optimizing π_E. Specifically, the graph representation module provides dense vectors h_u, h_v, e_u and e_v as user, item, user attribute and item attribute embeddings, respectively. Those embeddings are used in the state representation learning (i.e., learn s_t) and attentive action pruning (i.e., select a_t) in our CFE model. Moreover, the attribute embeddings are fused with user or item latent factors learned by the recommendation model to explore the model fairness change. In particular, the fused embeddings of users and items are used to predict the intervened recommendation result $H^{cf}_{u,K}$. By comparing the fairness disparity (*cf.* Eq. (6.20)) difference between $H^{cf}_{u,K}$ and the original recommendation $H_{u,K}$, we determine the reward $r(s_t,a_t)$ to optimize π_E, accordingly. The reward $r(s_t,a_t)$ measures whether the current attribute (i.e., action) is a feasible fairness explanation responsible for the fairness change. Finally, π_E is optimized with a counterfactual risk minimization (CRM) objective $\nabla_\Theta \widehat{R}(\pi_E)$ to balance the distribution discrepancy from the logging policy π_0.

Graph Representation Module. Our graph representation module conducts heterogeneous graph representation learning to produce dense vectors of users, items, and attributes among the HIN. Compared with homogeneous graph learning such as GraphSage [74], our graph representation injects both node and edge heterogeneity to preserve the complex structure of the HIN. In particular, we include two weight matrices to specify varying weights of different node and edge types. In the following, we present the graph learning for user embedding h_u. The embeddings of h_v, e_u, and e_v can be obtained analogously by replacing nodes and node types during computations. Specifically, we first use Multi-OneHot [254] to initialize node embeddings at the 0-th layer, in which

u's embedding is denoted by h_u^0. Then, at each layer l, user embedding h_u^l is given by aggregating node u's neighbor information w.r.t. different node and edge types:

$$\mathrm{h}_u^l = \sigma\left(\mathrm{concat}\left[\mathrm{W}_{\phi(u)}^l \mathrm{D}_p\left[\mathrm{h}_u^{l-1}\right], \frac{\mathrm{W}_{\psi(e)}^l}{\left|\mathcal{N}_{\psi(e)}(u)\right|} \sum_{u' \in \mathcal{N}_{\psi(e)}(u)} \mathrm{D}_p\left[\mathrm{h}_{u'}^{l-1}\right]\right] + b^l\right) \tag{6.21}$$

where $\sigma(\cdot)$ is LeakyReLU [237] activation function and concat$(\cdot)$ is the concatenation operator. $\mathrm{D}_p[\cdot]$ is a random dropout with probability p applied to its argument vector. h_u^{l-1} is u's embedding at layer $l-1$. $\mathcal{N}_{\psi(e)}(u) = \left\{u' \mid \left(u, e, u'\right) \in \mathcal{G}\right\}$ is a set of nodes connected with user node u through edge type $\psi(e)$. The additionally dotted two weight matrices, i.e., node-type matrix $\mathrm{W}_{\phi(u)}^l$ and edge-type matrix $\mathrm{W}_{\psi(e)}^l$, are defined based on the importance of each type $\phi(u)$ and $\psi(e)$. b^l is an optional bias.

With Eq (6.21), we obtain u's embedding h_u^l at each layer $l \in \{1, \cdots, L\}$. We then adopt layer-aggregation [238] to concatenate u's embeddings at all layers into a single vector, i.e., $\mathrm{h}_u = \mathrm{h}_u^{(1)} + \cdots + \mathrm{h}_u^{(L)}$. Finally, we have user node u's embedding h_u through aggregation. The item embedding h_v, user attribute embedding e_u and item attribute embedding e_v can be calculated analogously.

Recommendation Model. The recommendation model f_R is initialized using user-item interaction matrix $\boldsymbol{Y}$ to produce the Top-K recommendation result $H_{u,K}$ for all users. Here, we employ a linear and simple matrix factorization (MF) [162] as the recommendation model f_R. Particularly, MF initializes IDs of users and items as latent factors and uses the inner product of user and item latent factors as the predictive function:

$$f_R(u, v) = \boldsymbol{U}_u^\top \boldsymbol{V}_v \tag{6.22}$$

where $\boldsymbol{U}_u$ and $\boldsymbol{V}_v$ denote d-dimensional latent factors for user u and item v, respectively. We use the cross-entropy [259] loss to define the objective of the recommendation model:

$$\mathscr{L}_R = -\sum_{u, v, y_{uv} \in \boldsymbol{Y}} y_{uv} \log f_R(u, v) + (1 - y_{uv}) \log(1 - f_R(u, v)) \tag{6.23}$$

After optimizing the loss function $\mathscr{L}_R$, we can use the trained user and item latent factors (i.e., $\boldsymbol{U}$, $\boldsymbol{V}$) to produce the original Top-K recommendation lists $H_{u,K}$ for all users $u \in \mathcal{U}$.

6.2.4.2 Reinforcement Learning for Counterfactual Fairness Explanation

We cast our CFE model in an off-policy learning environment, which is formulated as Markov Decision Process (MDP). The MDP is provided with a static logged dataset

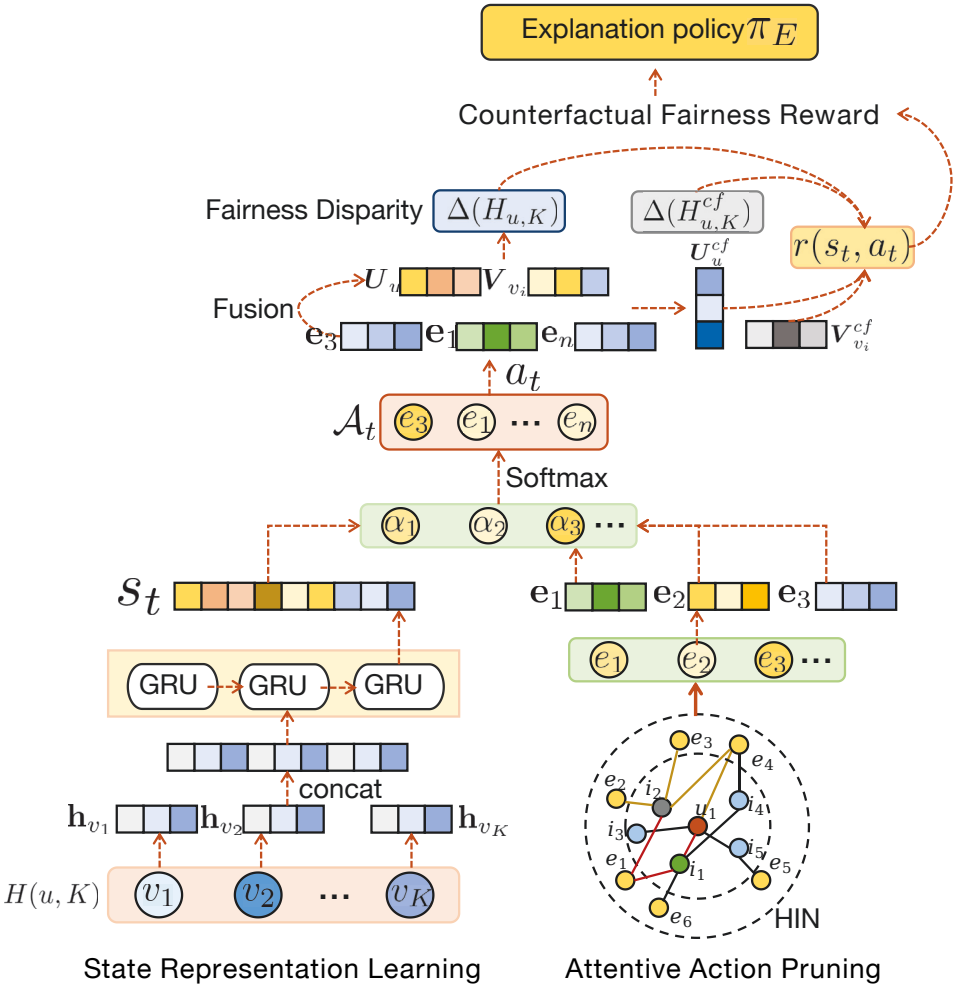


Figure 6.8: Counterfactual Fairness Explanation (CFE) Model.

generated by a logging policy π_0 [6]. The logging policy π_0 collects trajectories by uniformly sampling actions from the user and item attribute space. We use the off-policy learning to optimize an explanation (i.e., target) policy π_E by approximating the counterfactual rewards of state-action pairs from all timestamps, wherein the logging policy π_0 is employed for exploration while the target policy π_E is utilized for decision-making. In the off-policy setting, the explanation policy π_E does not require following the original pace of the logging policy π_0. As a result, π_E is able to explore the counterfactual region, i.e., those actions that haven't been taken by the previous agent using π_0. Formally, at each timestamp $t \in \{1, \cdots, T\}$ of MDP, the explanation policy $\pi_E(a_t|s_t)$ selects an action (i.e., a candidate attribute) $a_t \in \mathcal{A}_t$ conditioning on the user state $s_t \in \mathcal{S}$, and receives

[6]We adopt the uniform-based logging policy as π_0. It samples attributes as actions from the attribute space with the probability of $\pi_0(a_t \mid s_t) = \frac{1}{|\mathcal{V}_U + \mathcal{V}_I|}$.

counterfactual reward $r(s_t, a_t) \in \mathscr{R}$ for this particular state-action pair. Then the current state transits to the next state s_{t+1} with transition probability of $\mathbb{P}(s_{t+1} \mid s_t, a_t) \in \mathscr{P}$. The whole MDP has the key elements:

- $\mathscr{S}$ is a finite set of states $\{s_t \mid t \in [1, \cdots, T]\}$. Each state s_t is transformed into dense vectors (i.e., embeddings) by our *state representation learning*.

- $\mathscr{A}_t$ is a finite set of actions (i.e., attributes) available at s_t. $\mathscr{A}_t$ is select from attributes $\mathscr{V}_t \in \mathscr{G}$ by our *attentive action pruning* to reduce the search space.

- $\mathscr{P} : \mathscr{S} \times \mathscr{A} \rightarrow \mathscr{S}$ is the state transition, which absorbs transition probabilities of the current states to the next states. Given action a_t at state s_t, the transition to the next state s_{t+1} is determined as $\mathbb{P}(s_{t+1} \mid s_t, a_t) \in \mathscr{P} = 1$.

- $\mathscr{R} : \mathscr{S} \rightarrow \mathscr{R}$ is the counterfactual reward measures whether a deployed action (i.e., an attribute) is a valid counterfactual explanation for fairness. $\mathscr{R}$ is the counterfactual reward used to guide the explanation policy learning.

We now introduce the implementation of each key component.

State Representation Learning. The state $\mathscr{S}$ describes target users and their recommendation lists from the recommendation model. Formally, at step t, the state s_t for a user u is defined as $s_t = (u, H(u,K))$, where $u \in \mathscr{U}$ is a target user and $H(u,K)$ is the recommendation produced by f_R. The initial state s_0 is (u, v) and v is the interacted item of u, i.e., $y_{uv} \in \boldsymbol{Y} = 1$. Our state representation learning maps user state $s_t = (u, H(u,K))$ into dense vectors for latter explanation policy learning. Specifically, given s_t that absorbs current user u and its recommendation $H(u,K) = \{v_1, v_2, ..., v_K\}$. We first acquire the embedding h_{v_k} of each item $v_k \in H(u,K)$ from our graph representation module. The state s_t then receives the concatenated item embeddings (i.e., concat $\left[\mathrm{h}_{v_k} | \forall v_k \in H(u,K)\right]$) to update its representation. Considering states within $\mathscr{S}$ have sequential patterns [215], we resort to Recurrent Neural Networks (RNN) with a gated recurrent unit (GRU) [39] to capture the sequential state trajectory. We firstly initialize the state representation s_0 with an initial distribution $s_0 \sim \rho_0$ [7]. Then we learn state representation s_t through the

[7] In our experiment, we used a fixed initial state distribution, where $s_0 = 0 \in \mathbb{R}^d$.

recurrent cell:

$$
\begin{aligned}
\mathrm{u}_t &= \sigma_g\left(\mathrm{W}_1 \operatorname{concat}\left[\mathrm{h}_{v_k} | \forall v_k \in H(u,K)\right] + \mathrm{U}_1 s_{t-1} + b_1\right) \\
\mathrm{r}_t &= \sigma_g\left(\mathrm{W}_2 \operatorname{concat}\left[\mathrm{h}_{v_k} | \forall v_k \in H(u,K)\right] + \mathrm{U}_2 s_{t-1} + b_2\right) \\
\hat{s}_t &= \sigma_h\left(\mathrm{W}_3 \operatorname{concat}\left[\mathrm{h}_{v_k} | \forall v_k \in H(u,K)\right] + \mathrm{U}_3 (\mathrm{r}_t \cdot s_{t-1}) + b_3\right) \\
s_t &= (1 - \mathrm{u}_t) \cdot s_{t-1} + \mathrm{u}_t \odot \hat{s}_t
\end{aligned}
\tag{6.24}
$$

where u_t and r_t denote the update gate and reset gate vector generated by GRU and $\odot$ is the element-wise product operator. W_i, U_i are weight matrices and b_i is the bias vector. Finally, s_t serves as the state representation at time step t.

Attentive Action Pruning. Our attentive action pruning is designed to reduce the action search space by specifying the varying importance of actions for each state. As a result, the sample efficiency can be largely increased by filtering out irrelevant actions to promote an efficient action search. Our method defines actions as candidate attributes selected from a given HIN that potentially impact the model fairness. In particular, given state $s_t = (u, H(u,K))$, we can distill a set of attributes $\mathcal{V}_t$ of the current user u and items $v \in H(u,K)$ from the HIN. Intuitively, we can directly use $\mathcal{V}_t$ as candidate actions for state s_t. However, the user and item attribute amount of the HIN would be huge, resulting in a large search space that terribly degrades the learning efficiency [214]. Thus, we propose an attentive action pruning based on attention mechanism [195] to select important candidate actions for each state. Formally, given the embedding e_i for an attribute $i \in \mathcal{V}_t$ from Eq. (6.21), and the state representation s_t from Eq. (6.24), the attention score α_i of attribute i is:

$$
\alpha_i = \mathrm{ReLU}(\mathrm{W}_s s_t + \mathrm{W}_h \mathrm{e}_i + b) \tag{6.25}
$$

where W_s and W_h are two weight matrices and b is the bias vector. We then normalize attentive scores of all attributes in $\mathcal{V}_t$ and select attributes with n-top attention scores into $\mathscr{A}_t$:

$$
\mathscr{A}_t = \left\{ i \mid i \in \text{Top-n}\left[\frac{\exp(\alpha_i)}{\sum_{i' \in \mathcal{V}_t} \exp(\alpha_{i'})}\right] \text{ and } i \in \mathcal{V}_t \right\} \tag{6.26}
$$

where n is the candidate size, and n is chosen based on the parameter analysis made in the ablation study (*cf.* Section 6.2.5.4). To the end, our candidate set $\mathscr{A}_t$ is of high sample efficiency since it filters out irrelevant attributes while dynamically adapting to the user state shift.

Counterfactual Reward Definition. The counterfactual reward $r(s_t, a_t) \in \mathscr{R}$ measures whether a deployed action $a_t \in \mathscr{A}_t$ is a valid counterfactual explanation for fairness

at the current state s_t. Inspired by [213], the reward is defined based on two criteria: 1) *Rationality* [197]: deploying action (i.e., attribute) a_t should cause the reduction of fairness disparity regarding the item exposure fairness. The fairness disparity change is measured by the fairness disparity difference between the recommendation result before (i.e., $\Delta(H_{u,K})$) and after (i.e., $\Delta(H^{cf}_{u,K})$) fusing the action a_t to the recommendation model f_R, i.e., $\Delta(H_{u,K}) - \Delta(H^{cf}_{u,K})$. 2) *Proximity* [47]: a counterfactual explanation is a minimal set of attributes that changes the fairness disparity.

For the *Rationality*, we fuse the embedding of a_t with user or item latent factors from the recommendation model to learn updated user and item latent vectors, so as to get the $\Delta(H^{cf}_{u,K})$. Specifically, for a state $s_t = (u, H(u,K))$, the embedding e_t of action a_t is fused to user latent factor $\boldsymbol{U}_u$ for user u and item latent factors $\boldsymbol{V}_{v_i}$ for all items $v_i \in H(u,K)$ by a element-wise product fusion. As a result, we can get the updated $\boldsymbol{U}^{cf}_u$ and $\boldsymbol{V}^{cf}_v$:

$$\begin{aligned} \boldsymbol{U}^{cf}_u &\leftarrow \boldsymbol{U}_u \odot \{\mathrm{e}_t \mid \forall t \in [1, \cdots, T]\}, \text{if } a_t \in \mathcal{V}_U \\ \boldsymbol{V}^{cf}_{v_i} &\leftarrow \boldsymbol{V}_{v_i} \odot \{\mathrm{e}_t \mid \forall t \in [1, \cdots, T]\}, \text{if } a_t \in \mathcal{V}_I \end{aligned} \tag{6.27}$$

where $\odot$ represents the element-wise product (a.k.a. Hadamard product). T is the total training iteration. At the initial state of $t = 0$, user and item latent factors $\boldsymbol{U}_u$ and $\boldsymbol{V}_v$ are learned form Eq (6.22). Through Eq. (6.27), the updated user and item latent vectors are then used to generate the intervened recommendation result $H^{cf}_{u,K}$.

For the *Proximity*, we compute whether a_t returns a minimal set of attributes that changes the recommendation model fairness. This is equal to regulating user and item latent factors before (i.e., $\boldsymbol{U}_u$, $\boldsymbol{V}_v$) and after (i.e., $\boldsymbol{U}^{cf}_u$, $\boldsymbol{V}^{cf}_v$) fusing a_t be as similar as possible.

Based on the two criteria, the counterfactual reward can be defined as the following:

$$r(s_t, a_t) = \begin{cases} 1 + \mathrm{dist}(\boldsymbol{U}_u, \boldsymbol{U}^{cf}_u) + \mathrm{dist}(\boldsymbol{V}_v, \boldsymbol{V}^{cf}_v), \text{ if } \Delta(H_{u,K}) - \Delta(H^{cf}_{u,K}) \geq \epsilon \\ \mathrm{dist}(\boldsymbol{U}_u, \boldsymbol{U}^{cf}_u) + \mathrm{dist}(\boldsymbol{V}_v, \boldsymbol{V}^{cf}_v), \text{ otherwise} \end{cases} \tag{6.28}$$

where $\mathrm{dist}(\cdot)$ is the distance metric defined as cosine similarity, i.e., $\mathrm{dist}(a,b) = \frac{\langle a,b \rangle}{\|a\|\|b\|}$. $\Delta(\cdot)$ is the fairness disparity evaluation metric defined in Eq.(6.20). ϵ is the disparity change threshold that controls the model flexibility.

6.2.4.3 Unbiased Policy Optimization

Using state $s_t \in \mathcal{S}$ from Eq. (6.24), candidate action $a_t \in \mathcal{A}_t$ from Eq. (6.26), and counterfactual reward $r(s_t, a_t)$ defined in Eq. (6.28) for each timestamp t, we aim to learn an

explanation policy π_E by maximizing the expected cumulative reward $R(\pi_E)$ over total iteration T. Formally,

$$\max_{\pi_E} \quad R(\pi_E) = \mathbb{E}_{\tau \sim \pi_E} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{6.29}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ are trajectories and $\tau \sim \pi_E$ means the distribution over trajectories is induced by the explanation policy π_E. The explanation policy $\pi_E : \mathcal{S} \mapsto \mathcal{P}(\mathcal{A}_t)$ is a map from states $\mathcal{S}$ to a probability distribution over actions $\mathcal{A}_t$, where $\mathcal{A}_t$ are selected from the attentive action pruning. $\mathcal{P}$ are state transitions and $\mathbb{P}(s_{t+1} \mid s_t, a_t) \in \mathcal{P} = 1$.

Intuitively, we can directly use $R(\pi_E)$ in Eq. (6.29) to guide the optimization of π_E. However, this would lead to a biased policy optimization [214] due to the existence of policy distribution discrepancy between π_E and the logging policy π_0. In particular, in the off-policy learning setting, the logging policy π_0 focuses on explorations, in which a uniform-based π_0 is adopted to explore attributes from the attribute space uniformly at random. In contrast, the target explanation policy π_E aims to exploit the learned knowledge and take actions that maximize the cumulative rewards given by Eq. (6.29). As a result, the data collected under π_0 may not align with the explanation policy's distribution. Directly using the logged data collected by π_0 for learning can lead to biased estimates, which has been showed in previous works [133, 168].

Bias alleviation with Counterfactual Risk Minimization. Fortunately, Counterfactual Risk Minimization (CRM) [188] could alleviate the bias by correcting the distribution discrepancy between π_E and π_0. The core idea of CRM is to use inverse propensity scoring (IPS) to balance the distribution discrepancy. Specifically, IPS reweights the collected data using the importance weight, which is the ratio of the target policy's distribution to the logging policy's distribution for each timestamp. By reweighting the data with importance weights, the distribution discrepancy can be corrected, enabling unbiased learning. Formally, IPS calculates the importance weight β^t for each timestamp t:

$$\beta^t = \frac{\pi_E(a_t \mid s_t)}{\pi_0(a_t \mid s_t)} \tag{6.30}$$

where β^t is the importance weight for reweighing at timestamp t. Applying β^t to Eq (6.29), we can alleviate the policy distribution bias by maximizing the CRM-based expected cumulative reward $\widehat{R}(\pi_E)$:

$$\max_{\pi_E} \quad \widehat{R}(\pi_E) = \mathbb{E}_{\tau \sim \pi_E} \left[\sum_{t=0}^{T} \gamma^t \beta^t r(s_t, a_t) \right] = \mathbb{E}_{\tau \sim \pi_E} \left[\sum_{t=0}^{T} \gamma^t \frac{\pi_E(a_t \mid s_t)}{\pi_0(a_t \mid s_t)} r(s_t, a_t) \right] \tag{6.31}$$

Finally, the policy gradient of the explanation policy learning w.r.t. model parameter Θ is achieved by the REINFORCE [227]:

$$\nabla_\Theta \widehat{R}(\pi_E) = \frac{1}{T}\sum_{t=0}^{T} \gamma^t \frac{\pi_E(a_t \mid s_t)}{\pi_0(a_t \mid s_t)} r(s_t, a_t) \nabla_\Theta \log \pi_E(a_t \mid s_t) \tag{6.32}$$

where T is the total training iteration. By optimizing the Eq. (6.32), the learned explanation policy π_E generates minimal sets of attributes responsible for item exposure fairness changes, so as to find the plausible fairness-related factors leading to unfair recommendations.

6.2.4.4 Computational Complexity Analysis

The computational complexity of CFairER is primarily determined by the three core components: (1) Graph Representation Learning, (2) Recommendation Model Training, and (3) Counterfactual Fairness Explanation (CFE). Below, we analyze the complexity of each step. The graph representation module embeds users, items, and attributes in a heterogeneous information network (HIN). Let $|\mathcal{V}|$ be the number of nodes, $|\mathcal{E}|$ be the number of edges, and d be the embedding dimension. Common methods such as graph neural networks (GNNs) require message passing over multiple layers. Assuming an L-layer GNN, the complexity is $\mathcal{O}(L|\mathcal{E}|d)$. The recommendation model learns user and item latent factors for prediction. The typical matrix factorization (MF) method involves dot product computations between user and item embeddings. Given $|\mathcal{U}|$ users, $|\mathcal{I}|$ items, and embedding dimension d, the complexity per iteration is $\mathcal{O}(|\mathcal{U}|d + |\mathcal{I}|d)$. The CFE model constructs a state representation s_t for each user-item pair. Using dense embeddings from the graph representation module, this step requires $\mathcal{O}(|\mathcal{U}| + |\mathcal{I}|)d$. Selecting attribute-based counterfactual explanations requires attention mechanisms. Let m be the number of attribute candidates per user-item pair, the complexity per step is $\mathcal{O}(md)$. The fairness-aware intervention modifies the embeddings and computes fairness disparity. Since fairness disparity is computed over K-top recommended items, the complexity is $\mathcal{O}(Kd)$. The reward function is optimized via gradient-based learning. Using CRM, the policy update involves computing gradients over the logging policy distribution π_0. Let T is the number of iterations, and b is the batch size, this step costs $\mathcal{O}(Tbd)$. Summing up all the major computational steps, the overall complexity per iteration is: $\mathcal{O}(L|\mathcal{E}|d) + \mathcal{O}(|\mathcal{U}|d + |\mathcal{I}|d) + \mathcal{O}(md) + \mathcal{O}(Kd) + \mathcal{O}(Tbd)$, which is simplifies to $\mathcal{O}(L|\mathcal{E}|d + Tbd + |\mathcal{U}|d + |\mathcal{I}|d)$. The overall complexity is linear in the number of users, items, and edges, making CFairER scalable to large recommendation datasets. However,

graph representation learning remains the bottleneck, which can be optimized using sampling-based GNN methods or graph sparsification techniques.

6.2.5 Experiments

We conduct extensive experiments to evaluate the proposed CFairER to explain item exposure fairness in recommendations. We aim to particularly answer the following research questions:

- **RQ1.** Whether CFairER produces attribute-level explanations that are faithful to explaining recommendation model fairness compared with existing approaches?
- **RQ2.** Whether explanations provided by CFairER achieve better fairness-accuracy trade-off than other methods?
- **RQ3.** Do different components (i.e., recommendation model, attentive action pruning, counterfactual risk minimization-based optimization) help CFairER to achieve better generalizability, sample efficiency and bias alleviation, respectively?
- **RQ4.** How do hyper-parameters impact CFairER?
- **RQ5.** What is the time complexity and computation cost of CFairER?

6.2.5.1 Setup

Datasets. We use logged user behavior data from three datasets `Yelp` [8], `Douban Movie` [9] and `LastFM` [10] for evaluations. Each dataset is considered an independent benchmark for different tasks, i.e., business, movie, and music recommendation tasks. The `Yelp` dataset records user ratings on local businesses and business compliment, category and city profiles. The `Douban Movie` is a movie recommendation dataset that contains user group information and movie actor, director and type details. The `LastFM` contains music listening records of users and artist tags. The details of both datasets are given in Table 6.8, which depicts statistics of user-item interactions, user-attribute and item-attribute relations. All datasets constitute complex user-item interactions and diverse attributes, thus providing rich contextual information for fairness explanation learning. Following previous works [114, 214, 216], we adopt a 10-core setting, i.e., retaining users

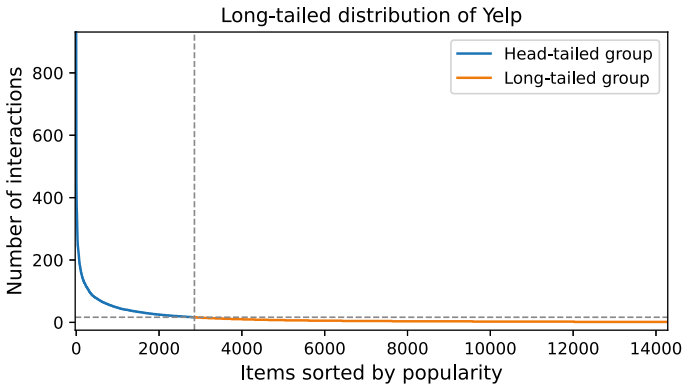
[8] https://www.yelp.com/dataset/
[9] https://movie.douban.com/
[10] https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding

Table 6.8: Statistics of datasets.

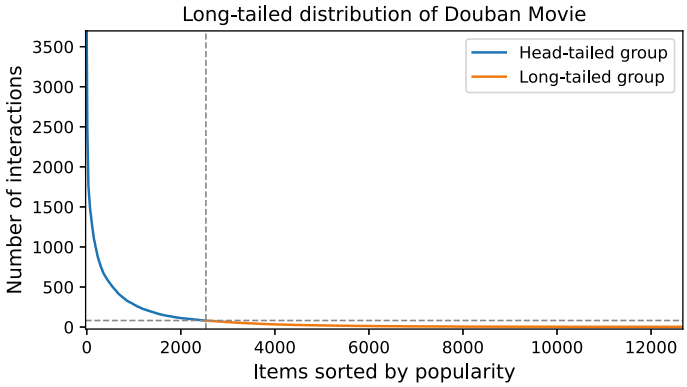
Dataset (Density)	Original		Preprocessed	
	Attribute: Number of Attribute	Relation: Number of Relation	Attribute: Number of Attribute	Relation: Number of Relation
Yelp (0.086%)	User (U): 16,239 Business (B): 14,284 Category (Ca): 511 City (Ci): 47 Compliment (Co): 11 Friend (F): 10,580	U-B: 198,397 B-U: 198,397 B-Ca: 40,009 B-Ci: 14,267 U-Co: 76,875 U-F: 158,590	User (U): 8,533 Business (B): 9,370 Category (Ca): 469 City (Ci): 46 Compliment (Co): 11 Friend (F): 9,609	U-B: 181,199 B-U: 181,199 B-Ca: 26,887 B-Ci: 9,363 U-Co: 41,149 U-F: 123,959
Douban Movie (0.63%)	User (U): 13,367 Movie (M): 12,677 Actor (A): 6,311 Director (D): 2,449 Type (T): 38 Group (G): 2,753 Friend (F): 2,294	U-M: 1,068,278 M-U: 1,068,278 M-A: 33,572 M-D: 11,276 M-T: 27,668 U-G: 570,047 U-F: 4,085	User (U): 10,075 Movie (M): 9,122 Actor (A): 5,960 Director (D): 2,281 Type (T): 38 Group (G): 2,753 Friend (F): 2,108	U-M: 1,059,305 M-U: 1,059,305 M-A: 26,247 M-D: 8,490 M-T: 20,420 U-G: 475,445 U-F: 3,651
LastFM (0.28%)	User (U): 1,892 Artist (A): 17,632 Tag (T): 9,718 Similar artist (S): 13,569 Friend (F): 1,749	U-A: 92,834 A-U: 92,834 A-T: 184,941 A-S: 153,399 U-F: 18,802	User (U): 1,879 Artist (A): 10,734 Tag (T): 6,392 Similar artist (S): 9,038 Friend (F): 1,747	U-A: 77,473 A-U: 77,473 A-T: 103,260 A-S: 107,987 U-F: 18,776

and items with at least ten interactions for both datasets to ensure the dataset quality. Meanwhile, we binarize the explicit rating data by interpreting ratings of 4 or higher as positive feedback, otherwise negative. Then, we sort the interacted items for each user based on the timestamp and split the chronological interaction list into train/test/valid sets with a proportion of 60%/20%/20%.

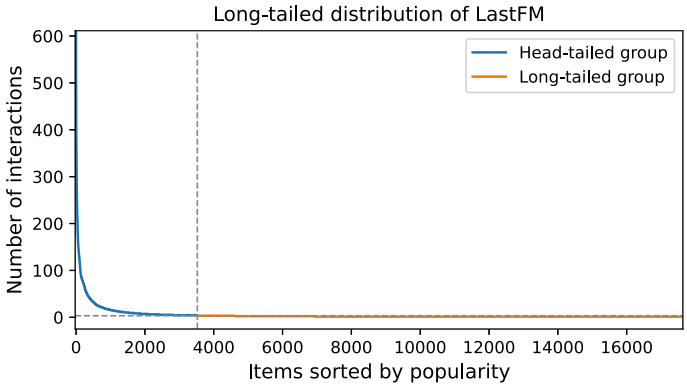
We also study the long-tail distribution of user-item interactions in the three datasets. We present the visualization results of the distribution in Figure 6.9. In particular, we first calculate the popularity of each item among user-item interactions, and sort the item index by the calculated popularity in descending order. Then, we plot the distribution of user-item interactions for both datasets: the x-axis represents the item index given by sorting the item popularity, and the y-axis represents the number of interactions (i.e., item popularity) in the dataset. We also calculate the split percentage of head-tailed items and long-tailed items and show the split as the vertical line in Figure 6.9. The head-tailed items are the top-ranked items based on popularity, positioned to the left (blue) of the vertical line; the long-tailed items are the remaining items positioned to the right (orange) of the vertical line. Upon analyzing Figure 6.9, we find that the user-item interactions in both datasets exhibit a skewed distribution. A small fraction of items are frequently interacted with by users, as evidenced by the head-tailed distribution in the blue plot area. Besides, we observe a long-tailed distribution in the orange plot area, where a majority of items receive a disproportionately small number of interactions. This



(a) User-item interaction distribution in `Yelp`.



(b) User-item interaction distribution in `Douban Movie`.



(c) User-item interaction distribution in `LastFM`.

Figure 6.9: Long-tailed distribution of item popularity in `Yelp`, `Douban Movie`, and `LastFM`.

suggests that while popular items capture a significant portion of user attention, a wide range of less popular items collectively contribute to a large fraction of interactions. In summary, the analysis in Figure 6.9 highlights the presence of a skewed distribution in the user-item interactions of both datasets, with a small fraction of popular items accounting for most of the user interactions in both datasets. The skewed distribution would result in serious item exposure unfairness issues in recommendations, such as the well-known filter-bubble problem [147] and Matthew effect [156].

Baselines. We adopt three heuristic approaches and two existing fairness-aware explainable recommendation methods as baselines. In particular, the three heuristic approaches **RDExp**, **PopUser** and **PopItem** generate explanations from user and item attributes that are chosen by random selection, top popular user attributes and top popular item attributes, respectively. The two fairness-aware explainable methods are **FairKGAT** [51] and **CEF** [56]. **FairKGAT** [51] mitigates the unfairness of explanation for a SOTA knowledge graph-enhanced recommender KGAT [204]. **CEF** [56] is the first work that explains fairness, in which aspect-based explanations are generated.

- **RDExp**: We randomly select attributes from the attribute space for each user-item interaction and generate explanations based on the selected attributes. Note that the selected attributes can be both user and item attributes. We fix the explanation length N as $N = 20$ for RDExp.

- **PopUser** and **PopItem**: We separately calculate the exposure number of attributes for each user-item interaction, then sort each attribute chronologically based on the exposure number. We devise a baseline **PopUser**, in which the top user attributes are selected as explanations. Analogously, we build **PopItem** that produces the top item attributes for the explanation. We fix the explanation length N as $N = 20$ for PopUser and PopItem.

- **FairKGAT**: uses FairKG4Rec [51] to mitigate the unfairness of explanations for a knowledge graph-enhanced recommender KGAT [204]. FairKG4Rec [51] is a generalized fairness-aware algorithm that controls the unfairness of explanation diversity in the recommendation model. KGAT [204] is a state-of-the-art knowledge graph-enhanced recommendation model that gives the best fairness performance in the original FairKG4Rec paper. We shortly denote the FairKG4Rec with KGAT as **FairKGAT**. The explanation length in FairKGAT is also fixed as $N = 20$.

- **CEF** [56]: is the first work that explains fairness in recommendation. It generates

feature-level explanations for item exposure unfairness by perturbing user and item features and searches for features that change the fairness disparity.

Note that to the best of our knowledge, **FairKGAT** [51] and **CEF** [56] are the only two existing methods designed for explainable fairness recommendation tasks.

Explanation Faithfulness Evaluation. We adopt the widely used erasure-based evaluation criterion [55] in Explainable AI to evaluate the explanation faithfulness. Note that the erasure-based evaluation is also used by CEF [56]. The erasure-based evaluation identifies the contributions of explanations by measuring model performance changes after these explanations are removed. As a result, one can tell whether the model actually relied on these particular explanations to make a prediction, i.e., faithful to the model. In our experiments, we use the erasure-based evaluation to test (I) the recommendation performance change and (II) the recommendation fairness change after a set of attributes from the generated explanation is removed. By doing so, we can identify whether our explanations are faithful to recommendation performance and fairness disparity. Following [55], we remove certain attributes from the generated explanations and evaluate the resulting recommendation performance. Therefore, in the starting evaluation point, we consider all attributes and add them to the user and item embeddings. We then remove certain attributes from the generated explanations to observe recommendation and fairness changes at later evaluation points. In particular, we first use historical user-item interactions to train a recommendation model through Eq. (6.23) to generate user and item embeddings. Then, we fuse all attribute embeddings from Eq. (6.21) with the trained user and item embeddings. The user and item embeddings after fusion are used to generate recommendation results at the starting evaluation point. Thereafter, we conduct counterfactual reasoning using our CFairER to generate attribute-level counterfactual explanations for model fairness. Those generated explanations are defined as the erasure set of attributes for each user/item. Finally, we exclude the erasure set from attribute space, and fuse the embeddings of attributes after erasure with the trained user and item embeddings to generate new recommendation results. Given the recommendation results at each evaluation point, we use two widely adopted recommendation evaluation metrics, namely, Normalized Discounted Cumulative Gain (NDCG)@K and Hit Ratio (HR)@K, to measure the recommendation accuracy. As this work focuses on explaining item exposure fairness in recommendations, we use two item-side evaluation metrics, i.e., Head-tailed Rate (HT)@K and Gini@K, for fairness evaluation. HT@K refers to the ratio of the head-tailed (i.e., popular) item number in each recommendation to the recommendation list length K. Those head-tailed items are

the top-ranked items selected based on the calculated item popularity for each dataset, and are derived from the statistical analysis on each dataset. Intuitively, the head-tailed items positioned to the left (blue) of the vertical lines in Figure 6.9 (a) (b) (c) for `Yelp`, `Douban Movie` and `LastFM`, respectively. Having obtained the defined head-tailed items and the recommendation list length K, HT@K is calculated by,

$$HT@K = \frac{N_{\text{head-tailed}}}{K} \tag{6.33}$$

where $N_{\text{head-tailed}}$ is the number of head-tailed items. Lager HT@K indicates that the model suffers from a more severe item exposure disparity by favoring items from the head-tailed (i.e., popular) group. Gini@K measures the inequality within the two subgroups (i.e., the head-tailed group and the long-tailed group) among the Top-K recommendation list. Larger Gini@K indicates recommendations are of higher inequality between the head-tailed and the long-tailed group.

Implementation Details. To demonstrate our CFairER, we employ a simple matrix factorization (MF) as our recommendation model. We train the MF using train/test/validate sets split from user-item interactions in datasets with 60%/20%/20%. The same data-splitting method is applied in all baselines. We optimize the MF using stochastic gradient descent (SGD). The same gradient descent method is leveraged for baselines when required. Our graph representation module employs two graph convolutional layers with $\{64, 128\}$ output dimensions. FairKGAT baseline also keeps 2 layers. The graph representation module outputs embeddings for all user and item attributes with the embedding size $d = 128$. The embedding size for FairKGAT and CEF is also fixed as $d = 128$. The number of latent factors (as in Eq. (6.22)) of MF is set equal to the embedding size of our graph representation module. To generate the starting evaluation point of erasure-based evaluation, we fuse attribute embeddings with the trained user and item latent factors based on element-wise product fusion. The fused user and item embeddings are then used to produce Top-K recommendation lists.

We train our counterfactual fairness explanation model with SGD based on the REINFORCE [186] policy gradient. For baseline model compatibility, we follow CEF [56] to use "Sentires" [11] for constructing user-feature attention matrix and item-feature quality matrix for CEF. The hyper-parameters of training all baselines are chosen by the grid search, including learning rate, L_2 norm regularization, etc. The batch size for our CFairER and baselines are set as 256. The training epochs for our CFairER and baselines are set to a maximum of 400, and an early stopping strategy is performed to stop the

[11] https://github.com/evison/Sentires

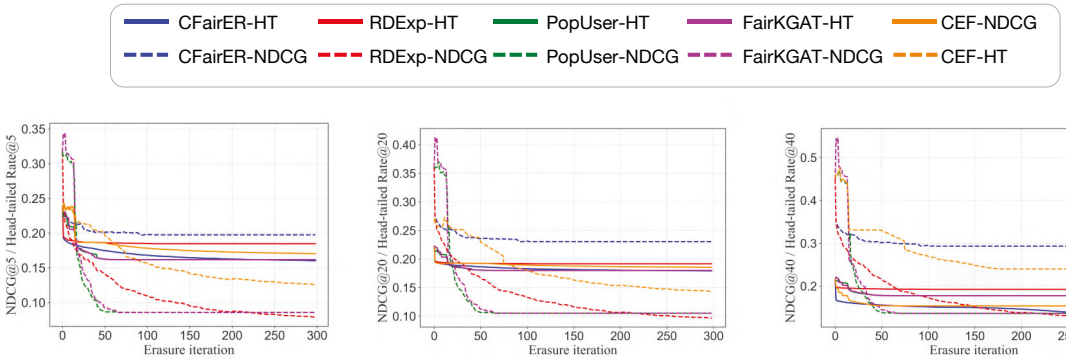
training process if the loss fails to decrease for consecutive 5 epochs. After all models have been trained, we freeze the model parameters and generate explanations accordingly. We report the erasure-based evaluation results by recursively erasing top E attributes from the generated explanations. The erasure length E is chosen from $E = [5, 10, 15, 20]$. The recommendation and fairness performance of our CFairER and baselines under different E is reported in Table 6.9. For the unique parameters of our CFairER, we determine the discount factor $\gamma^t \in [0, 1]$ in Eq. (6.32) by a grid search. The total reinforcement timestamp T is set to $T = 100$. The disparity change threshold $\epsilon \in [0, 1]$ in Eq. (6.28), and the fairness disparity trade-off $\lambda \in \{0, 1\}$ in Eq. (6.20) is determined by performing a grid search on the validation set. This enables us to choose the optimal value for a variety of recommendation tasks, including but not limited to business (`Yelp` dataset), movie (`Douban Movie` dataset), and music (`LastFM` dataset) recommendations.

6.2.5.2 Explanation Faithfulness (RQ1, RQ2)

Table 6.9: Recommendation and fairness performance after erasing top $E = [5, 10, 20]$ attributes from explanations.

Method	NDCG@40 ↑			Hit Ratio (HR)@40 ↑			Head-tailed Rate (HT)@40 ↓			Gini@40 ↓		
	$E=5$	$E=10$	$E=20$	$E=5$	$E=10$	$E=20$	$E=5$	$E=10$	$E=20$	$E=5$	$E=10$	$E=20$
						`Yelp`						
RDExp	0.0139	0.0125	0.0118	0.1153	0.1036	0.1029	0.1994	0.1976	0.1872	0.3870	0.3894	0.3701
PopUser	0.0141	0.0136	0.0128	0.1183	0.1072	0.1067	0.1776	0.1767	0.1718	0.3671	0.3642	0.3495
PopItem	0.0147	0.0139	0.0131	0.1182	0.1093	0.1084	0.1793	0.1846	0.1848	0.3384	0.3370	0.3359
FairKGAT	0.0153	0.0141	0.0138	0.1384	0.1290	0.1277	0.1802	0.1838	0.1823	0.3671	0.3508	0.3542
CEF	0.0254	0.0247	0.0231	0.1572	0.1608	0.1501	0.1496	0.1455	0.1420	0.3207	0.3159	0.3088
CFairER	**0.0316**	**0.0293**	**0.0291**	**0.1987**	**0.1872**	**0.1868**	**0.1345**	**0.1322**	**0.1301**	**0.2366**	**0.2068**	**0.1974**
						`Douban Movie`						
RDExp	0.0390	0.0346	0.0351	0.1278	0.1170	0.1172	0.1932	0.1701	0.1693	0.3964	0.3862	0.3741
PopUser	0.0451	0.0352	0.0348	0.1482	0.1183	0.1174	0.1790	0.1674	0.1658	0.3684	0.3591	0.3562
PopItem	0.0458	0.0387	0.0379	0.1523	0.1219	0.1208	0.1831	0.1458	0.1383	0.3664	0.3768	0.3692
FairKGAT	0.0534	0.0477	0.0421	0.1602	0.1377	0.1308	0.1654	0.1573	0.1436	0.3590	0.3472	0.3483
CEF	0.0831	0.0795	0.0809	0.1949	0.1973	0.1901	0.1043	**0.0998**	**0.0945**	0.3079	0.2908	0.3001
CFairER	**0.1290**	**0.0921**	**0.0901**	**0.2706**	**0.2441**	**0.2238**	**0.0841**	0.1183	0.1101	**0.2878**	**0.2648**	**0.2593**
						`LastFM`						
RDExp	0.0857	0.0568	0.0592	0.8436	0.7915	0.7831	0.7884	0.7732	0.7691	0.3707	0.3531	0.3540
PopUser	0.0786	0.0432	0.0431	0.7787	0.7697	0.7604	0.7979	0.8064	0.7942	0.3862	0.3729	0.3761
PopItem	0.0792	0.0479	0.0435	0.7803	0.7961	0.7914	0.7689	0.7638	0.7573	0.3673	0.3602	0.3618
FairKGAT	0.0832	0.0594	0.0621	0.8063	0.7938	0.8165	0.7451	0.7342	0.7408	0.3580	0.3458	0.3433
CEF	0.0962	0.1037	0.1001	0.8592	0.8408	0.8509	0.6873	0.6092	**0.5601**	0.3308	0.3298	0.3375
CFairER	**0.1333**	**0.1193**	**0.1187**	**0.9176**	**0.8867**	**0.8921**	**0.6142**	**0.5865**	0.5737	**0.2408**	**0.2371**	**0.2385**

Figure 6.10 explicitly shows how the fairness (i.e., Head-tailed Rate@K) and recommendation (i.e., NDCG@K) performance of our CFairER and baselines changes along with erasure, where the x-axis shows the erasure iteration and the y-axis demonstrates



(a) NDCG@5 and Head-tailed Rate@5 on `Yelp`

(b) NDCG@20 and Head-tailed Rate@20 on `Yelp`

(c) NDCG@40 and Head-tailed Rate@40 on `Yelp`

(d) NDCG@5 and Head-tailed Rate@5 on `Douban Movie`

(e) NDCG@20 and Head-tailed Rate@20 on `Douban Movie`

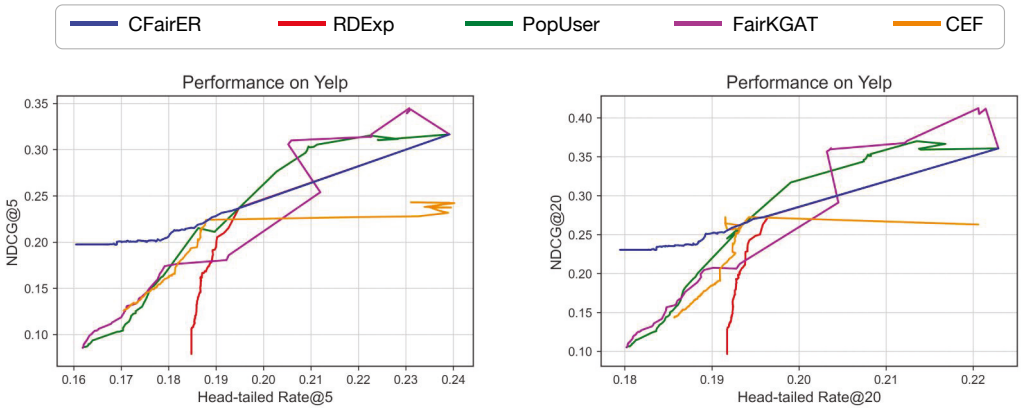
(f) NDCG@40 and Head-tailed Rate@40 on `Douban Movie`

(g) NDCG@5 and Head-tailed Rate@5 on `LastFM`

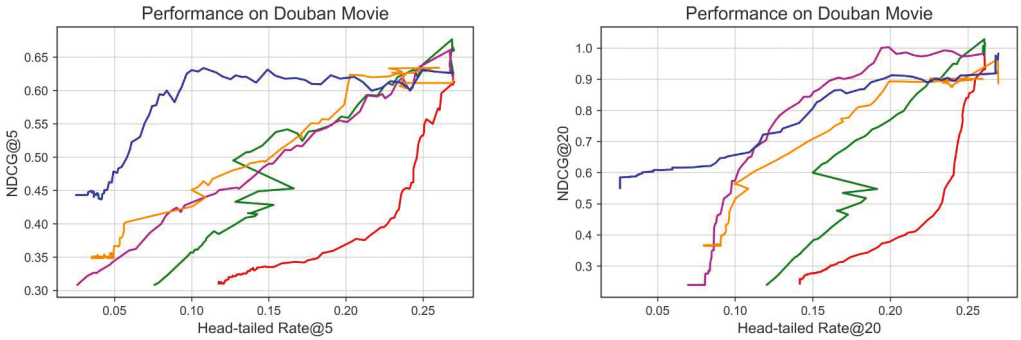
(h) NDCG@20 and Head-tailed Rate@20 on `LastFM`

(i) NDCG@40 and Head-tailed Rate@40 on `LastFM`

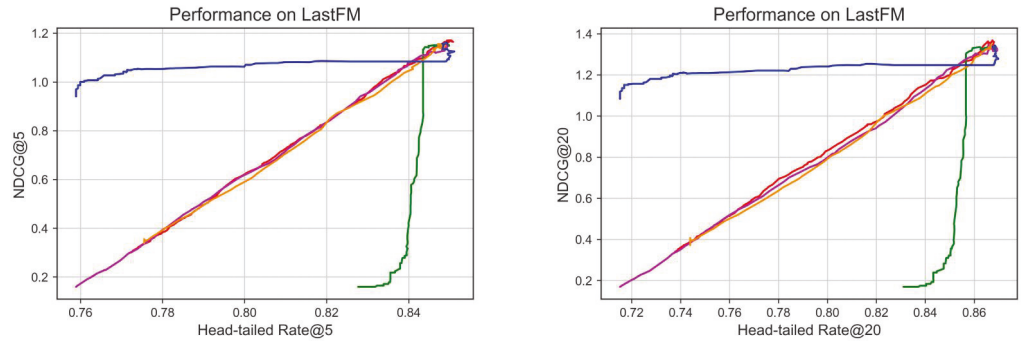
Figure 6.10: Erasure-based evaluation on Top-5, Top-20 and Top-40 recommendations. K is chosen from $K = \{5, 20, 40\}$. NDCG@K values are multiplied with 10 for better presentation. Each data point is generated while cumulatively erasing the top 10 (i.e., $E = 10$) attributes in explanations.



(a) NDCG@5 and Head-tailed Rate@5 on `Yelp` (b) NDCG@20 and Head-tailed Rate@20 on `Yelp`



(c) NDCG@5 and Head-tailed Rate@5 on `Douban Movie` (d) NDCG@20 and Head-tailed Rate@20 on `Douban Movie`



(e) NDCG@5 and Head-tailed Rate@5 on `LastFM` (f) NDCG@20 and Head-tailed Rate@20 on `LastFM`

Figure 6.11: Fairness-accuracy trade-off of CFairER and baseline models on Top-5, Top-20 recommendations. The x-axis denotes the fairness performance, i.e., Head-tailed Rate@K; the y-axis demonstrates the recommendation performance, i.e., NDCG@K. NDCG@K values are multiplied with 10 for better presentation.

the corresponding fairness and recommendation performance at $K = \{5, 20\}$. Each data point in Figure 6.10 is generated by cumulatively erasing a batch of attributes. Those erased attributes are selected from, at most [12], the top 10 (i.e., $E = 10$) attribute sets of the explanation lists provided by each method [13]. As PopUser and PopItem baselines enjoy very similar data trends, we choose not to present them simultaneously in Figure 6.10. Besides, we plot the relationship between fairness and recommendation performance at each erasure iteration in Figure 6.11, showcasing the fairness-accuracy trade-off of our CFairER and baselines. We also use Table 6.9 to present the final recommendation and fairness performance of all methods after erasing $E = [5, 10, 20]$ attributes in explanations. Note that in both Figure 6.10, Figure 6.11 and Table 6.9, larger NDCG@K and Hit Ratio @K values indicate better recommendation performance while smaller Head-tailed Rate@K and Gini@K values represent better fairness. Analyzing Figure 6.10, Figure 6.11 and Table 6.9, we have the following findings.

From Table 6.9, it is observed that our CFairER achieves the best recommendation and fairness performance amongst all methods after erasing attributes from our explanations. For instance, CFairER beats the strongest baseline CEF by 25.9%, 24.4%, 8.3% and 36.0% for NDCG@40, Hit Ratio@40, Head-tailed Rate@40 and Gini@40 with erasure length $E = 20$ on `Yelp`. This indicates that explanations generated by CFairER are more faithful to explaining unfair factors while not harming recommendation accuracy compared to all baseline methods. Unlike heuristic approaches (i.e., RDExp, PopUser, and PopItem) that rely on random attribute selection, CFairER incorporates a more complex and rational attribute selection mechanism for fairness explanation generation. Particularly, CFairER prioritizes relevant attributes that change the model fairness by using a counterfactual reward (*cf.* Eq. (6.28)), incorporating two criteria of *Rationality* and *Proximity* to ensure the quality of generated counterfactual explanations. Besides, FairKGAT considers mitigating the unfairness of explanation diversity caused by different user activeness, which ignores the impact of item exposure imbalance on explanation fairness. Our CFairER mitigates the unfairness of item exposure to promote the fair allocation of user-preferred but less exposed items, thus achieving better recommendation and fairness performance than FairKGAT. Regarding CEF, although CEF generates counterfactual explanations as in our CFairER, it conducts feature-level optimizations and does not apply to discrete attributes such as gender and age. Our

[12] Our CFairER generates counterfactual explanations; thus, the explanation length is adaptive and may not reach $E = 10$. In such a case, we select all attributes as erased attributes.

[13] For example, given n explanation lists, the number of erasure attributes is $n \times 10$. We cumulatively erase m attributes in one batch within in total $(n \times 10)/m$ iterations.

CFairER uses an off-policy learning agent to directly visit attributes in a given HIN, adapting to both discrete and continuous attributes when finding the counterfactual explanations. As a result, our CFairER outperforms CEF due to its generalizability to discrete attributes. Another interesting finding is that PopUser and PopItem perform even worse than RDExp (i.e., randomly selecting attributes) on `LastFM`. Early recommendation models largely recommend items that have popular attributes favored by users. Though this is intuitive, recommending items with popular attributes would deprive the exposure of less-noticeable items, causing serious model unfairness and degraded recommendation performance. This further highlights the importance of mitigating item exposure unfairness in recommendations.

From Figure 6.10, the fairness of all models consistently improves while erasing attributes from explanations, shown by the decreasing trend of Head-tailed Rate@K values. Besides, Figure 6.10 also shows the decreasing trend of NDCG@K values, demonstrating a decreased recommendation performance. The improved fairness performance of all methods is reasonable, as erasing attributes, even those selected randomly from attribute sets, could potentially remove unfair factors to mitigate the representation gap between popular and long-tailed items. This finding is also consistent with the finding in CEF [56]. Unfortunately, we can observe the downgraded recommendation performance of all models in Figure 6.10, as also evidenced by CEF [56]. For example, in Figure 6.10, the NDCG@5 of CEF drops from approximately 1.17 to 0.60 on `LastFM` at erasure iteration 0 and 50. This is due to the well-known fairness-accuracy trade-off issue, in which the fairness constraint could be achieved with a sacrifice of recommendation performance. Facing this issue, both baselines suffer from huge declines in recommendation performance. On the contrary, our CFairER still enjoys favorable recommendation performance and outperforms all baselines. Besides, the decline rates of our CFairER are much slower than baselines on both datasets in Figure 6.10. We hence conclude that the attribute-level explanations provided by our CFairER can achieve a much better fairness-accuracy trade-off than other methods. This is because our CFairER finds minimal but vital attributes as explanations for model fairness. Those attributes produced by CFairER are fairness-related factors but not the ones that affect the recommendation accuracy. As a result, our CFairER could alleviate the item exposure unfairness while maintaining stable recommendation performance. Another finding is that our CFairER may not outperform FairKGAT and PopUser in fairness evaluations when the number of erasure iterations is insufficient. In Figure 6.10 (a), it is observed that the HT@5 of FairKGAT and PopUser exhibit more quick degradation compared to CFairER's degradation at

the beginning of the erasure iterations. This can be attributed to the fact that FairKGAT and PopUser construct explanations using a fixed length (i.e., $N = 20$), which may absorb more attributes in each explanation than CFairER's adaptive explanation length, i.e., minimal for each explanation. Consequently, CFairER may not have a fair opportunity to hit the attributes that explain model unfairness compared to FairKGAT and PopUser, resulting in slower degradation of HT@5 during the initial erasure iterations. However, as more erasure iterations are performed, CFairER exhibits stable performance and eventually surpasses FairKGAT and PopUser in fairness evaluations. This indicates that CFairER consistently discovers suitable explanations that align with model fairness during the learning process. These explanations generated by CFairER prioritize simple yet essential attributes, in contrast to the complex combinations of attributes utilized by FairKGAT and PopUser.

Figure 6.11 provides deeper insights into the fairness-accuracy trade-off issue, specifically the relationship between fairness and recommendation performance metrics, i.e., Head-tailed Rate@K and NDCG@K. By analyzing Figure 6.11, we find that our CFairER achieves the best fairness-accuracy trade-off compared to the baseline methods on the `Douban Movie` and `LastFM` datasets. This is evident from the blue curves positioned to the left-hand side of the diagonals of Figure 6.11 (c) (d) (e) (f). Moreover, we observe that RDExp performs the poorest on the `Douban Movie` dataset, while PopUser exhibits the worst performance on the `LastFM` dataset. In our experiments, the trade-off is caused by the disagreement between the goals of item exposure fairness and user preference. While we aim to align fair allocations to item exposures, the recommendation model primarily focuses on selecting items similar to those in users' historical interactions. We thus conclude the attributes found by our CFairER are not necessarily similar to the attributes of historical items. Instead, they are sensitive attributes that cause the recommendations to favor the historically popular items. Another finding is that our CFairER initially does not outperform other baselines during the early erasure process on the `Yelp` dataset, as depicted in Figure 6.11 (a) (b). We attribute this sub-optimal performance on `Yelp` to the extremely sparse nature of the dataset, i.e., a density of only 0.086%. Compared with `Douban Movie` (0.63% density) and `LastFM` (0.28% density), `Yelp` records a larger amount of users that have few interactions with items. While other baseline methods may rely on popular attributes to predict the preferences of those users, CFairER takes a different approach. It aims to identify sensitive attributes that may not necessarily be the most popular ones. Consequently, at the beginning of the erasure process, CFairER may struggle to adapt to the data sparsity of the `Yelp`

dataset, resulting in sub-optimal performance. However, as more erasure iterations are performed, CFairER surpasses baseline models and achieves the best fairness-accuracy trade-off. This demonstrates the effectiveness of CFairER in progressively refining its attribute selection process and achieving a favorable fairness-accuracy trade-off. With more iterations, CFairER can identify the appropriate attributes that better explain item exposure fairness and align with user preferences.

6.2.5.3 Ablation Study (RQ3)

We conduct an in-depth ablation study on the ability of our CFairER to achieve favorable generalizability, sample efficiency and bias alleviation. In particular, our CFairER includes three important components that contribute to the good performance of CFairER, i.e., recommendation model, attentive action pruning and counterfactual risk minimization-based optimization. We evaluate our CFairER with different variant combinations and show our main findings below.

Generalizability to Recommendation Models. We first do the ablation study on recommendation models, showcasing the generalizability of our CFairER. In particular, we consider three different benchmark recommendation models, i.e., BPR [162], NeuMF [81] and NGCF [205]:

- BPR [162]: a well-known matrix factorization-based model with a pairwise ranking loss to enable recommendation learning from implicit feedback.
- NeuMF [81]: extends collaborative filtering to neural network architecture. It maps users and items into dense vectors and feeds user and item vectors into a multi-layer perception to predict user preferences.
- NGCF [205]: a graph-based model that incorporates two graph convolutional networks to learn user and item embeddings. The learned embeddings are passed to a matrix factorization to capture the collaborative signal for recommendations.

We substitute our MF-based recommendation model by BPR, NeuMF and NGCF, respectively, utilizing the user and item embeddings produced by the three models for the latter explanation policy learning. The fairness and recommendation performance of our CFairER under the three recommendation models is reported in Table 6.10. We have the following findings. Overall, when utilizing NeuMF and NGCF, CFairER consistently demonstrates improved performance, evident from the increased NDCG@20 and HR@20 values, as well as the decreased HT@20 and Gini@20 values. NeuMF and NGCF are

two recent state-of-the-art (SOTA) recommendation models, which extend collaborative filtering (CF) to neural network architecture and graph-based modeling, respectively. NeuMF makes a notable extension of CF to neural networks, leveraging the power of deep learning to achieve the SOTA performance. NGCF enhances NeuMF by modeling complex user-item interactions in graph-structured data, capturing higher-order relationships among users and items. Compared to the simple MF used in CFairER, these advanced models capture intricate user behaviors, such as collaborative behavior in NeuMF and higher-order relationships in NGCF. Thus, the increased performance of CFairER with these advanced models highlights its potential for achieving better recommendations. Additionally, even when employing the basic MF model, our CFairER outperforms baseline approaches, as shown in Table 6.9. This highlights the effectiveness of CFairER regardless of the reliance on recommendation models. Another finding is that BPR leads to downgraded recommendation performance for CFairER. This could be attributed to BPR's reliance on binary implicit feedback, which may lack the expressive ability present in the rating data used by MF. In summary, the ablation study demonstrates the generalizability of CFairER and its potential to achieve better performance when combined with advanced recommendation models.

Table 6.10: Ablation study on recommendation models. Erasure length E is fixed as $E = 20$.

Recommendation Model	NDCG@20↑	HR@20 ↑	HT@20↓	Gini@20↓
		Yelp		
MF (CFairER)	0.0238	0.1871	0.1684	0.1990
BPR [162]	0.0273 (+14.71%)	0.1869 (−0.11%)	0.1579 (−6.23%)	0.2083 (+4.67%)
NeuMF [81]	0.0279 (+17.23%)	0.2021 (+8.02%)	0.1563 (−7.19%)	0.1832 (−7.94%)
NGCF [205]	0.0284 (+19.33%)	0.1927 (+2.99%)	0.1530 (−9.16%)	0.1801 (−9.50%)
		Douban Movie		
MF (CFairER)	0.0583	0.2043	0.1149	0.2871
BPR [162]	0.0592 (+1.54%)	0.1992 (−2.50%)	0.1084 (−5.66%)	0.2860 (−0.38%)
NeuMF [81]	0.0611 (+4.80%)	0.2101 (+2.89%)	0.1070 (−6.86%)	0.2771 (−3.48%)
NGCF [205]	0.0638 (+9.44%)	0.2178 (+6.66%)	0.1073 (−6.61%)	0.2708 (−5.67%)
		LastFM		
MF (CFairER)	0.1142	0.7801	0.6914	0.2670
BPR [162]	0.1138 (−0.35%)	0.7792 (−0.12%)	0.6908 (−0.09%)	0.2672 (+0.07%)
NeuMF [81]	0.1209 (+5.87%)	0.7983 (+2.34%)	0.6831(−1.20%)	0.2580 (−3.37%)
NGCF [205]	0.1271 (+11.30%)	0.8031 (+2.95%)	0.6744 (−2.46%)	0.2461 (−7.82%)

Sample Efficiency of Attentive Action Pruning. Our attentive action pruning reduces the action search space by specifying varying importance of attributes for each state. As a result, the sample efficiency can be increased by filtering out irrelevant

Table 6.11: Ablation study on core modules. Erasure length E is fixed as $E = 20$.

Variants	NDCG@20↑	HR@20 ↑	HT@20↓	Gini@20↓
		Yelp		
CFairER	0.0238	0.1871	0.1684	0.1990
CFairER ¬ Attentive Action Pruning	0.0164(−31.1%)	0.1682(−10.1%)	0.1903(+13.0%)	0.2159(+8.5%)
CRM loss → Cross-entropy [259] loss	0.0197(−17.2%)	0.1704(−8.9%)	0.1841(+9.3%)	0.2101(+5.6%)
		Douban Movie		
CFairER	0.0583	0.2043	0.1149	0.2871
CFairER ¬ Attentive Action Pruning	0.0374(−35.9%)	0.1537(−24.8%)	0.1592(+38.6%)	0.3574(+24.5%)
CRM loss → Cross-entropy [259] loss	0.0473(−18.9%)	0.1582(−22.6%)	0.1297(+12.9%)	0.3042(+6.0%)
		LastFM		
CFairER	0.1142	0.7801	0.6914	0.2670
CFairER ¬ Attentive Action Pruning	0.0987(−13.6%)	0.7451(−4.5%)	0.7833(+13.3%)	0.2942(+10.2%)
CRM loss → Cross-entropy [259] loss	0.0996(−12.8%)	0.7483(−4.1%)	0.7701(+11.4%)	0.2831(+6.0%)

attributes to promote an efficient action search. To demonstrate our attentive action pruning, we test CFairER without (¬) the attentive action pruning (i.e., *CFairER ¬ Attentive Action Pruning*), in which the candidate actions set absorbs all attributes connected with the current user and items. We report the performance of *CFairER ¬ Attentive Action Pruning* in Table 6.11. Through Table 6.11, we observed that removing the attentive action pruning downgrades CFairER performance. We attribute the downgraded performance of CFairER to the exponentially increased computational costs after removing the attentive action pruning. These costs require careful model optimization by needing a sufficient number of iterations and samples during training, as well as longer training times. Since our ablation study is conducted in a fair setting, wherein training iterations and samples remain the same, the downgraded performance after removing the attentive action pruning is reasonable. This is due to the limited training environment prepared for CFairER, which leads to a sub-optimal explanation policy facing such a high computational cost. This sheds light on the importance of the attentive action pruning module, as it alleviates the computational burden by filtering out irrelevant actions, resulting in enhanced sample efficiency. Besides, without pruning, the model should theoretically have a more extensive search space. However, a larger search space would introduce more irrelevant actions that can bias the model from finding appropriate actions. The attentive action pruning module plays a crucial role in filtering out these irrelevant actions, thereby facilitating more efficient searching for counterfactual explanations. By filtering out irrelevant actions from consideration, the model can focus on identifying the most impactful actions for generating accurate and meaningful explanations. Moreover, the performance of CFairER after removing the attentive action pruning downgrades severely on Douban Movie. This is because Douban

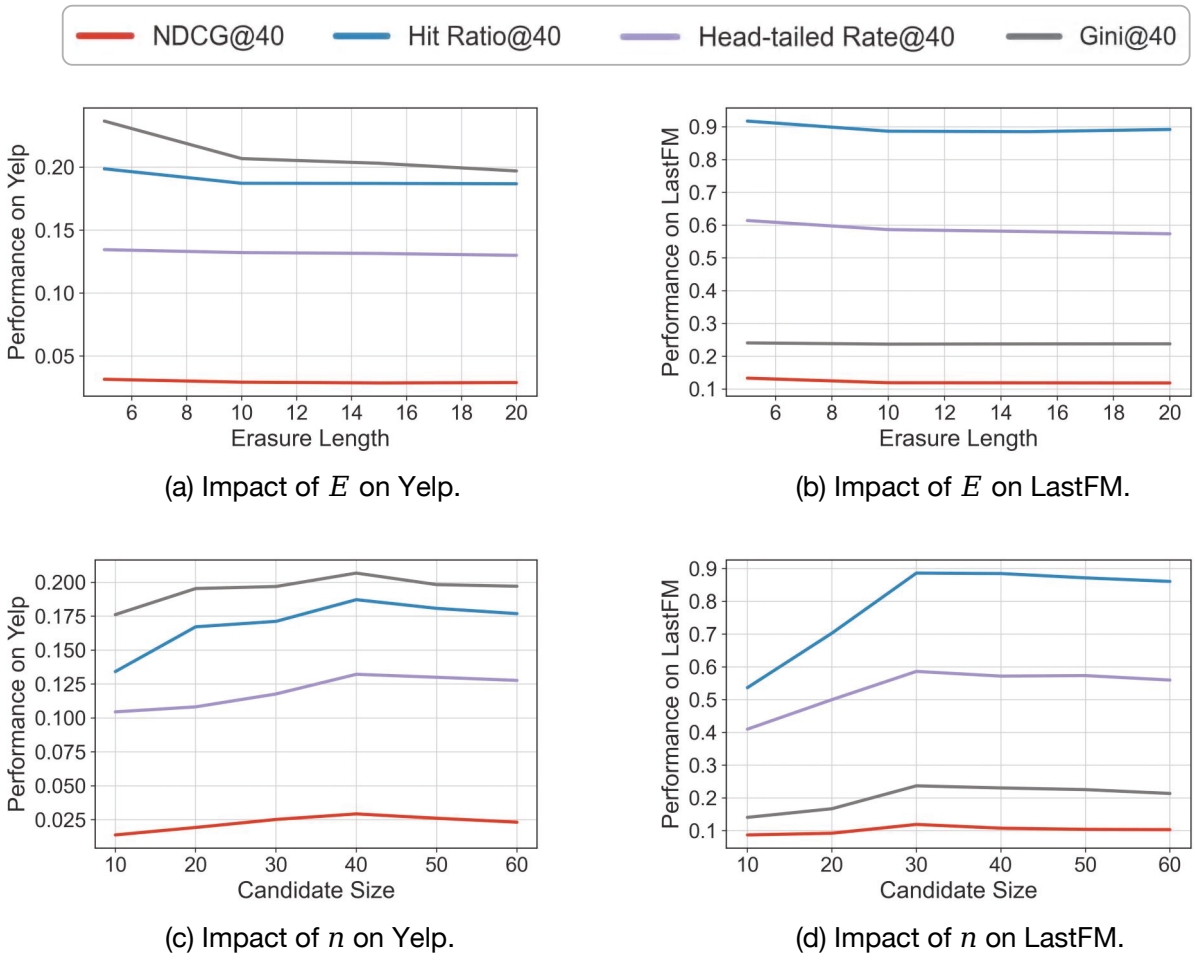
`Movie` has the largest number of attributes compared with the other two datasets (*cf.* Table 6.8), which challenges our CFairER to find suitable attributes as fairness explanations. These findings suggest the superiority of applying attentive action pruning in fairness explanation learning, especially when the attribute size is large.

Bias Alleviation with Counterfactual Risk Minimization. Our CFairER is optimized with a counterfactual risk minimization (CRM) loss to achieve unbiased policy optimization. The CRM loss (*cf.* Eq. (6.31)) corrects the discrepancy between the explanation policy and logging policy, thus alleviating the policy distribution bias in the off-policy learning setting. To demonstrate the CRM loss, we apply our CFairER with cross-entropy (CE) [259] loss (i.e., *CRM loss* → *Cross-entropy loss*) to show how it performs compared with CFairER on the CRM loss. We report the performance of *CRM loss* → *Cross-entropy loss* in Table 6.11. We observe our CFairER with CRM loss consistently outperforms the counterpart with CE loss on both fairness and recommendation performance. The sub-optimal performance of our CFairER with CE loss indicates that the bias issue in the off-policy learning can lead to downgraded performance for the learning agent. On the contrary, our CFairER takes advantage of CRM to learn a high-quality explanation policy. Hence, we conclude that performing unbiased optimization with CRM is critical to achieving favorable fairness explanation learning.

6.2.5.4 Parameter Analysis (RQ4)

We analyze how erasure length E and candidate size n (as in Eq. (6.26)) impact the performance of CFairER. Erasure length E is the number of erased attributes selected from each explanation, which determines the erasure size for evaluations. Candidate size n is the number of candidate actions selected by our attentive action pruning. We present the evaluation results of CFairER under different E and n on `Yelp` and `LastFM` in Figure 6.12. Since the results on `Douban Movie` draw similar conclusions to the other datasets, we choose not to present them here.

Apparently, the performance of CFairER demonstrates decreasing trends from $E = 5$, then becomes stable after $E = 10$. The decreased performance is due to the increasing erasure of attributes found by our generated explanations. This indicates that our CFairER can find valid attribute-level explanations that impact fair recommendations. The performance of CFairER degrades slightly after the bottom, then becomes stable. This is reasonable since the attributes number provided in datasets are limited, while increasing the erasure length would allow more overlapping attributes with previous erasures to be found.



(a) Impact of E on Yelp.

(b) Impact of E on LastFM.

(c) Impact of n on Yelp.

(d) Impact of n on LastFM.

Figure 6.12: Parameter sensitivity analysis: Impacts of parameters E and n on CFairER.

By varying candidate size n from $n = [10, 20, 30, 40, 50, 60]$ in Figure 6.12 (c) (d), we observe that CFairER performance first improves drastically as candidate size increases on both datasets. The performance of our CFairER reaches peaks at $n = 40$ and $n = 30$ on `Yelp` and `LastFM`, respectively. After the peaks, we can witness a downgraded model performance by increasing the candidate size further. We consider the poorer performance of CFairER before reaching peaks is due to the limited candidate pool, i.e., insufficient attributes limit the exploration ability of CFairER to find appropriate candidates as fairness explanations. Meanwhile, a too-large candidate pool (e.g., $n = 60$) would offer more chances for the agent to select inadequate attributes as explanations. Based on the two findings, we believe it is necessary for our CFairER to carry the attentive action search, such as to select high-quality attributes as candidates based on their contributions to the current state.

6.2.5.5 Time Complexity and Computation Costs (RQ5)

For time complexity, our recommendation model performs matrix factorization with a complexity of $O(|\mathscr{O}|)$. For the graph representation module, establishing node representations has complexity $O(\sum_{l=1}^{L}(|\mathscr{G}|+|\mathscr{O}^+|)d_l d_{l-1})$. For the off-policy learning process, the complexity is mainly determined by the attention score calculation, which has a time complexity of $O(2T|\mathscr{O}^+||\tilde{\mathscr{N}_e}|d^2)$. The total time complexity is $O(|\mathscr{O}|+\sum_{l=1}^{L}(|\mathscr{G}|+|\mathscr{O}^+|)d_l d_{l-1}+2T|\mathscr{O}^+|n_2 d^2)$. We evaluated the running time of FairKGAT and CEF baselines on the large-scale `Yelp` dataset. The corresponding results are 232s and 379s per epoch, respectively. CFairER has a comparable cost of 284s per epoch to these baselines. Considering that our CFairER achieves superior explainability improvements compared to the baselines, we believe that the increased cost of, at most, 52s per epoch is a reasonable trade-off.

6.2.6 Summary

This research work proposes a reinforcement learning-based fairness explanation learning framework named CFairER that operates over a heterogeneous information network (HIN). CFairER generates counterfactual explanations in the form of minimal sets of real-world attributes to provide insights into item exposure unfairness. Unlike previous approaches, CFairER takes a distinct approach to abandon feature-level optimizations and instead focuses on searching for real-world attributes from the HIN. This unique characteristic of CFairER allows it to adapt to both continuous and discrete attributes, providing flexibility in capturing various types of attribute information. In CFairER framework, we incorporate a counterfactual fairness explanation model and leverage attentive action pruning to reduce the search space. We define the counterfactual fairness explanation model as an off-policy learning agent, employing a counterfactual reward to enable the counterfactual reasoning for attribute-level counterfactual explanations. Finally, CFairER discovers high-quality counterfactual explanations that effectively capture fairness factors and meanwhile align with user preferences. Extensive evaluations conducted on three benchmark datasets validate the effectiveness of CFairER. The evaluation results demonstrate CFairER's ability to generate faithful explanations for fairness while balancing the fairness-accuracy trade-off.

CHAPTER 7

CONCLUSION

This thesis investigates the causal learning paradigm for building recommender systems (RecSys), addressing the four research questions detailed in Section 1.3. To summarize, the four research questions concern R1) providing preliminaries of causal notions for RecSys research questions; R2) Developing practical solutions to address bias issues using causal inference approaches; 3) Developing causality-enhanced methods for improved model accuracy and 4) Developing solutions to enhance model explainability and fairness through causal means.

To answer R1, Chapter 3 defines RecSys scientific questions from a causal perspective, harmonizing terminology and concepts across both disciplines. This alignment helps mitigate misunderstandings in problem formulations and enables more rigorous problem definitions in causal recommendations. In particular, this thesis mainly investigates challenging tasks in data-centric and model-centric RecSys, covering data bias, model accuracy, and model explainability and fairness. Chapter 3 provides comprehensive causal notions and practical causal solutions to address those challenging tasks. Consequently, Chapter 3 provides a distinct understanding of which causal approaches are suitable for specific research scenarios, clarifying the perplexing causal notions and approaches within a particular RecSys task.

To answer R2, Chapter 4 introduces two research works that alleviate data bias issues using causal inference. Specifically, two bias issues are considered, namely exposure bias and popularity bias, showcasing the most representative bias issues in RecSys. The proposed DENC model (De-bias Network Confounding in Recommendation) mitigates

exposure bias under the missing not at random (MNAR) phenomenon. Propensity scores in causal learning are applied to achieve unbiased model building in the presence of exposure bias. The proposed CaDSI model (Causal Disentanglement for Semantics-Aware Intent Learning) mitigates popularity bias in context information-aware recommendation scenarios. A disentangled learning approach is applied to separate users' true intent from the popularity bias.

To answer R3, Chapter 6 discusses research works that improve the accuracy of recommendations by refining 1) the architecture of the neural network and 2) the objective of optimizing the model. The NCGCF model (Neural Causal Graph Collaborative Filtering) connects the Graph Convolutional Network (GCN) to the Structural Causal Model (SCM) by using a Neural Causal Model, thereby improving the accuracy of the conventional graph collaborative filtering method. The HINpolicy model (HIN augmented off-policy learning) introduces a counterfactual estimator for counterfactual policy learning. This estimator employs counterfactual risk minimization to address the discrepancy between the logging policy and the target policy in an off-policy learning setting, ensuring unbiased policy learning for the reinforcement learning agent.

To answer R4, Chapter 5 presents research works that enhance recommendation models with explainability and fairness. Using counterfactual learning, the CERec model (Counterfactual Explainable Recommendation) generates item attribute-level counterfactual explanations to elucidate which item attributes attract users' interests. The CFairER model (Counterfactual Explanation for Fairness) generates attribute-level counterfactual explanations to identify which attributes cause item exposure unfairness.

In summary, this thesis provides a comprehensive understanding of causality-centric recommender systems, demonstrating their numerous advantages over traditional recommendation methods, including higher accuracy, improved explainability, enhanced fairness, and better generalization ability. The proposed causal methods offer valuable insights for future research in areas such as recommendation accuracy uplift, explainable recommender systems, fairness-aware recommender systems, and reinforcement learning-based recommender systems.

CHAPTER 8

OPEN PROBLEMS AND FUTURE DIRECTIONS

New theories, methodologies, and applications in causality-centric recommendation are emerging at a rapid pace but are still in a relatively early stage. There are many promising directions and interesting open problems that require extensive efforts to explore. In particular, important future directions include the following aspects.

Discovery of Causal Graphs for Recommendation. Existing causality-centric recommender systems largely rely on predefined causal graphs or structural causal models built from domain knowledge, which have two significant limitations. First, the assumed causal relationships within the causal graph may be inaccurate, and the domain knowledge may lack validity. Since the proposed framework hinges on the quality of the causal graph conceptualization, the model's effectiveness remains unstable. Second, these manually crafted causal graphs are often simplistic, typically involving only a few variables such as user conformity, user interest, and user behavior. However, in real-world scenarios, users' decision-making processes involve many factors, and these hidden variables are actually the true causes but are not captured by the causal graph. Causal discovery methods, on the other hand, do not depend on explicit domain knowledge but learn causal relationships from real-world data. These methods identify conditional independence relationships between pairs of variables and then construct a directed acyclic graph based on these relationships. Therefore, focusing on discovering causal relations and leveraging these learned causal relations to enhance recommendations is a promising and crucial future direction. Moreover, developing evaluation metrics that validate whether the causal graph is effective within the context of recommender systems

is promising, as this would fundamentally affect the performance of causal-enhanced recommender systems.

Evaluation Metrics for Causality-based Methods. Existing causality-based methods rely on conventional evaluation metrics used in information retrieval and recommendations to evaluate model performance. Those conventional evaluation metrics focus primarily on accuracy, ranking quality, and user engagement. However, for causality-based methods, additional considerations are necessary to measure the effectiveness of causal interventions. One direction is to design causality-aware evaluation metrics that explicitly assess how well a model captures causal effects in recommendations. For example, we could consider evaluating counterfactual stability to measure the stability of recommendations under counterfactual interventions, i.e., how much the recommended items change when key causal factors are altered. In addition, evaluating interventional fairness is another opportunity, in which we could analyze whether causal interventions mitigate biases rather than simply reflecting correlation-based fairness adjustments. In my future research, I will further explore this promising direction and propose effective metrics to enhance the evaluation of causal recommender systems.

Causal Assumptions in Recommendations. As discussed in Chapter 3, causal inference approaches rely on several causal assumptions to learn causality from observational data, as these observational data are not derived from ideal experimental conditions. For example, the potential outcome framework assumes ignorability (unconfoundedness), meaning there are no factors influencing both treatment assignment and potential outcomes. However, many causality-centric recommender systems fail to explicitly clarify or adhere to these assumptions. Instead, they often ignore these important assumptions when using causal inference approaches or make simplistic assumptions about data distributions. For instance, the positivity assumption is essential in potential outcome framework-based approaches for unbiased causal effect estimation, but the data sparsity problem in recommender systems makes it challenging to satisfy. As a result, many causality-centric recommender systems choose to overlook the positivity assumption, leading to estimation inconsistency when using causal inference approaches. Therefore, estimating the impact of violating causal assumptions in causality-centric recommender systems is crucial for achieving consistent estimations. Given that even small differences in recommendation accuracy can lead to significant changes in platform revenue, it is important to investigate the effects of violating causal assumptions and to develop feasible and reliable assumptions.

Transfer Learning and Out-of-distribution Recommendation. Many causality-

centric recommender systems require domain knowledge from recommender system experts to construct causal graphs, which form the foundation of the proposed framework. Additionally, data sparsity in recommender systems is a significant issue due to major concerns such as user privacy and platform benefits. Transfer learning can address both the limited domain knowledge and data sparsity issues by transferring user and item knowledge from other domains to improve recommendation performance, especially in cold start scenarios and domain knowledge adaptation. Moreover, user preferences may naturally shift over time, violating the i.i.d. assumption and leading to out-of-distribution (OOD) recommendation issues. For example, users might shift their interests from fashion to household goods after getting married, but their inherent preference for luxury brands may remain unchanged. Thus, a solution to OOD recommendation is to capture users' inherent and unchanging preferences when facing OOD issues. The core of both transfer learning and OOD recommendation is to transfer beneficial shared knowledge, such as users' inherent and unchanging preferences. This process can sometimes be formulated as invariant learning. As causal learning aims to uncover unchangeable causal relationships in data, as introduced in Chapter 2, causal inference approaches can be applied to enhance robustness and generalization for cross-domain or OOD recommendations.

Dynamic Recommendation. In real-world scenarios, a recommender system typically operates as a feedback loop, where the data generation process evolves over time by incorporating users' dynamic feedback on recommended items. Within this loop, the recommended items selected by the system significantly influence users' decisions and interests, which in turn affect the feedback the system receives. Consequently, the recommender system is dynamic, continually adapting to shifting user feedback and evolving user interests. However, the causal graph is constructed as a directed acyclic graph representing the data generation process, and such an acyclic graph cannot capture the dynamic nature of the recommender system loop. Therefore, it is critical to introduce loops into the causal graph to enable the modeling of the dynamic data generation process. There have been some prior efforts in this direction, such as the works in [71, 239]. However, introducing loops into causal graphs presents several challenges. Firstly, designing a causal graph with loops to describe the dynamic and iterative data generation process of recommendations lacks theoretical support and rational discussion. As a result, whether the recommendation performance can be ensured remains unknown and requires further assessment. Secondly, uncontrolled feedback loops may cause significant issues such as the Matthew effect, echo chambers, and bias amplification, leading to

severe bias problems. Traditional debiasing methods, such as back-door adjustment, cannot be applied directly due to the change in the form of causal models. Therefore, designing causal graphs with loops to adapt to dynamic recommendations still require substantial efforts and further research.

Causality-aware Simulator and Environment for Recommendation. Simulation creates a virtual environment where recommendation algorithms can be tested and evaluated. By recreating user interactions and behaviors in a controlled manner, simulators allow researchers and developers to assess the performance of recommendation models before deploying them in real-world scenarios. Simulation is an important task in both industry and academia. For industry developers, testing recommendation algorithms in a simulated environment helps identify potential issues without risking user dissatisfaction or other negative consequences. For researchers, simulations enable the testing of proposed methods even when facing restrictions on accessing real-world recommender systems. Existing simulations largely rely on reinforcement learning to simulate the decision-making process of users, aiming to replicate the data generation process. For example, Pseudo Dyna-Q [265] proposes a customer simulator called the World Model, designed to simulate the recommendation environment using off-policy learning. However, these simulators seldom consider causality in the data generation process. This is a missed opportunity because causality helps uncover true user interests, ensuring that the generated data genuinely reflects users' actual preferences. Incorporating causality into simulations leads to more stable and accurate estimations, resulting in a more robust simulation environment. Moreover, traditional simulators may suffer from various bias issues, such as the selection bias in Pseudo Dyna-Q caused by the off-policy learning setting. Causal models can identify and mitigate these biases, enhancing the stability and accuracy of the simulation environment. In summary, causality-aware simulators and environments for recommender systems are essential for advancing the field. They provide a robust platform for developing, testing, and refining recommendation algorithms by incorporating causal reasoning, leading to more effective recommendations.

Causal Neural Networks. Neural networks serve as the foundation of artificial intelligence (AI) because they are capable of learning complex patterns and representations from data, enabling advanced AI applications in various fields such as computer vision, natural language processing, and recommender systems. However, neural networks primarily conduct correlation learning from data, which often overlooks underlying causal relationships. For instance, as discussed in Chapter 5, while Graph Convolutional Net-

works (GCNs) are widely used in recommendation models, they struggle to capture the intricate causal relationships within the user-item interaction graph. Nowadays, there are countless neural networks and their variants; how to integrate neural networks and causal modeling to facilitate causality-aware neural networks is a promising direction.

BIBLIOGRAPHY

[1] H. ABDOLLAHPOURI, R. BURKE, AND B. MOBASHER, *Controlling popularity bias in learning-to-rank recommendation*, in Proceedings of the eleventh ACM conference on recommender systems, 2017, pp. 42–46.

[2] D. AGARWAL AND B.-C. CHEN, *Regression-based latent factor models*, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 19–28.

[3] I. M. AL JAWARNEH, P. BELLAVISTA, A. CORRADI, L. FOSCHINI, R. MONTANARI, J. BERROCAL, AND J. M. MURILLO, *A pre-filtering approach for incorporating contextual information into deep learning based recommender systems*, IEEE Access, 8 (2020), pp. 40485–40498.

[4] B. ALTAF, U. AKUJUOBI, L. YU, AND X. ZHANG, *Dataset recommendation via variational graph autoencoder*, in 2019 IEEE International Conference on Data Mining (ICDM), IEEE, 2019, pp. 11–20.

[5] P. C. AUSTIN AND E. A. STUART, *Moving towards best practice when using inverse probability of treatment weighting (iptw) using the propensity score to estimate causal treatment effects in observational studies*, Statistics in medicine, 34 (2015), pp. 3661–3679.

[6] K. BALOG AND F. RADLINSKI, *Measuring recommendation explanation quality: The conflicting goals of explanations*, in Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020, pp. 329–338.

[7] E. BAREINBOIM, J. D. CORREA, D. IBELING, AND T. ICARD, *On pearl's hierarchy and the foundations of causal inference*, in Probabilistic and Causal Inference: The Works of Judea Pearl, 2022, pp. 507–556.

[8] T. BEGLEY, T. SCHWEDES, C. FRYE, AND I. FEIGE, *Explainability for fair machine learning*, arXiv preprint arXiv:2010.07389, (2020).

[9] G. BEIGI, A. MOSALLANEZHAD, R. GUO, H. ALVARI, A. NOU, AND H. LIU, *Privacy-aware recommendation with private-attribute protection using adversarial learning*, in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 34–42.

[10] R. V. D. BERG, T. N. KIPF, AND M. WELLING, *Graph convolutional matrix completion*, arXiv preprint arXiv:1706.02263, (2017).

[11] A. BEUTEL, J. CHEN, T. DOSHI, H. QIAN, L. WEI, Y. WU, L. HELDT, Z. ZHAO, L. HONG, E. H. CHI, ET AL., *Fairness in recommendation ranking through pairwise comparisons*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2212–2220.

[12] A. V. BODAPATI, *Recommendation systems with purchase data*, Journal of marketing research, 45 (2008), pp. 77–93.

[13] K. BOLLACKER, R. COOK, AND P. TUFTS, *Freebase: A shared database of structured general human knowledge*, in AAAI, vol. 7, 2007, pp. 1962–1963.

[14] S. BONNER AND F. VASILE, *Causal embeddings for recommendation*, in Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 104–112.

[15] F. BORISYUK, K. KENTHAPADI, D. STEIN, AND B. ZHAO, *Casmos: A framework for learning candidate selection models over structured queries and documents*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 441–450.

[16] L. BOTTOU, *Large-scale machine learning with stochastic gradient descent*, in Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[17] L. BOTTOU, J. PETERS, J. QUIÑONERO-CANDELA, D. X. CHARLES, D. M. CHICKERING, E. PORTUGALY, D. RAY, P. SIMARD, AND E. SNELSON, *Counterfactual reasoning and learning systems: The example of computational advertising.*, Journal of Machine Learning Research, 14 (2013).

[18] J. S. BREESE, D. HECKERMAN, AND C. KADIE, *Empirical analysis of predictive algorithms for collaborative filtering*, arXiv preprint arXiv:1301.7363, (2013).

[19] R. M. BYRNE, *The rational imagination: How people create alternatives to reality*, MIT press, 2007.

[20] I. CANTADOR, P. BRUSILOVSKY, AND T. KUFLIK, *2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)*, in Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011, New York, NY, USA, 2011, ACM.

[21] Z. CAO, T. QIN, T.-Y. LIU, M.-F. TSAI, AND H. LI, *Learning to rank: from pairwise approach to listwise approach*, in Proceedings of the 24th international conference on Machine learning, 2007, pp. 129–136.

[22] Y.-W. CHANG, C.-J. HSIEH, K.-W. CHANG, M. RINGGAARD, AND C.-J. LIN, *Training and testing low-degree polynomial data mappings via linear svm.*, Journal of Machine Learning Research, 11 (2010).

[23] C. CHEN, W. MA, M. ZHANG, C. WANG, Y. LIU, AND S. MA, *Revisiting negative sampling vs. non-sampling in implicit recommendation*, ACM Transactions on Information Systems, 41 (2023), pp. 1–25.

[24] 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.

[25] J. CHEN, H. DONG, X. WANG, F. FENG, M. WANG, AND X. HE, *Bias and debias in recommender system: A survey and future directions*, arXiv preprint arXiv:2010.03240, (2020).

[26] ——, *Bias and debias in recommender system: A survey and future directions*, ACM Transactions on Information Systems, 41 (2023), pp. 1–39.

[27] J. CHEN, C. WANG, M. ESTER, Q. SHI, Y. FENG, AND C. CHEN, *Social recommendation with missing not at random data*, in 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 29–38.

[28] J. CHEN, X. WANG, F. FENG, AND X. HE, *Bias issues and solutions in recommender system: Tutorial on the recsys 2021*, in Proceedings of the 15th ACM Conference on Recommender Systems, 2021, pp. 825–827.

[29] L. CHEN, R. MA, A. HANNÁK, AND C. WILSON, *Investigating the impact of gender on rank in resume search engines*, in Proceedings of the 2018 chi conference on human factors in computing systems, 2018, pp. 1–14.

[30] M. CHEN, A. BEUTEL, P. COVINGTON, S. JAIN, F. BELLETTI, AND E. H. CHI, *Top-k off-policy correction for a reinforce recommender system*, in Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 456–464.

[31] X. CHEN, L. YAO, J. MCAULEY, G. ZHOU, AND X. WANG, *A survey of deep reinforcement learning in recommender systems: A systematic review and future directions*, arXiv preprint arXiv:2109.03540, (2021).

[32] Y. CHEN, Y. WANG, X. ZHAO, J. ZOU, AND M. D. RIJKE, *Block-aware item similarity models for top-n recommendation*, ACM Transactions on Information Systems (TOIS), 38 (2020), pp. 1–26.

[33] Z. CHEN, M. XU, B. GAO, G. SUGIHARA, F. SHEN, Y. CAI, A. LI, Q. WU, L. YANG, Q. YAO, ET AL., *Causation inference in complicated atmospheric environment*, Environmental Pollution, 303 (2022), p. 119057.

[34] W. CHENG, Y. SHEN, L. HUANG, AND Y. ZHU, *Incorporating interpretability into latent factor models via fast influence analysis*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 885–893.

[35] J. CHUNG, C. GULCEHRE, K. CHO, AND Y. BENGIO, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, arXiv preprint arXiv:1412.3555, (2014).

[36] P. COVINGTON, J. ADAMS, AND E. SARGIN, *Deep neural networks for youtube recommendations*, in Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 191–198.

[37] Y. DELDJOO, A. BELLOGIN, AND T. DI NOIA, *Explaining recommender systems fairness and accuracy through the lens of data characteristics*, Information Processing & Management, 58 (2021), p. 102662.

[38] Y. DELDJOO, M. G. CONSTANTIN, H. EGHBAL-ZADEH, B. IONESCU, M. SCHEDL, AND P. CREMONESI, *Audio-visual encoding of multimedia content for enhancing movie recommendations*, in Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 455–459.

[39] R. DEY AND F. M. SALEM, *Gate-variants of gated recurrent unit (gru) neural networks*, in 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), IEEE, 2017, pp. 1597–1600.

[40] F. DIAZ, B. MITRA, M. D. EKSTRAND, A. J. BIEGA, AND B. CARTERETTE, *Evaluating stochastic rankings with expected exposure*, in Proceedings of the 29th ACM international conference on information & knowledge management, 2020, pp. 275–284.

[41] Y. DONG, N. V. CHAWLA, AND A. SWAMI, *metapath2vec: Scalable representation learning for heterogeneous networks*, in Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 135–144.

[42] Y. DONG, S. WANG, J. MA, N. LIU, AND J. LI, *Interpreting unfairness in graph neural networks via training node attribution*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 7441–7449.

[43] Z. DONG, Z. WANG, J. XU, R. TANG, AND J. WEN, *A brief history of recommender systems*, arXiv preprint arXiv:2209.01860, (2022).

[44] A. EDMUNDS AND A. MORRIS, *The problem of information overload in business organisations: a review of the literature*, International journal of information management, 20 (2000), pp. 17–28.

[45] C. EKSOMBATCHAI, P. JINDAL, J. Z. LIU, Y. LIU, R. SHARMA, C. SUGNET, M. ULRICH, AND J. LESKOVEC, *Pixie: A system for recommending 3+ billion items to 200+ million users in real-time*, in Proceedings of the 2018 world wide web conference, 2018, pp. 1775–1784.

[46] 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.

[47] C. FERNANDEZ, F. J. PROVOST, AND X. HAN, *Explaining data-driven decisions made by AI systems: The counterfactual approach*, CoRR, abs/2001.07417 (2020).

[48] S. FLAXMAN, S. GOEL, AND J. M. RAO, *Filter bubbles, echo chambers, and online news consumption*, Public opinion quarterly, 80 (2016), pp. 298–320.

[49] N. FROSST AND G. E. HINTON, *Distilling a neural network into a soft decision tree*, in Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017, T. R. Besold and O. Kutz, eds., vol. 2071 of CEUR Workshop Proceedings, CEUR-WS.org, 2017.

[50] J. FU, R. GAO, Y. YU, J. WU, J. LI, D. LIU, AND Z. YE, *Contrastive graph learning long and short-term interests for poi recommendation*, Expert Systems with Applications, 238 (2024), p. 121931.

[51] 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.

[52] C. GAO, X. HE, D. GAN, X. CHEN, F. FENG, Y. LI, T.-S. CHUA, AND D. JIN, *Neural multi-task recommendation from multi-behavior data*, in 2019 IEEE 35th international conference on data engineering (ICDE), IEEE, 2019, pp. 1554–1557.

[53] Y. GE, W. HUA, K. MEI, J. TAN, S. XU, Z. LI, Y. ZHANG, ET AL., *Openagi: When llm meets domain experts*, Advances in Neural Information Processing Systems, 36 (2024).

[54] Y. GE, S. LIU, R. GAO, Y. XIAN, Y. LI, X. ZHAO, C. PEI, F. SUN, J. GE, W. OU, ET AL., *Towards long-term fairness in recommendation*, in Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 445–453.

[55] Y. GE, S. LIU, Z. LI, S. XU, S. GENG, Y. LI, J. TAN, F. SUN, AND Y. ZHANG, *Counterfactual evaluation for explainable ai*, arXiv preprint arXiv:2109.01962, (2021).

[56] Y. GE, J. TAN, Y. ZHU, Y. XIA, J. LUO, S. LIU, Z. FU, S. GENG, Z. LI, AND Y. ZHANG, *Explainable fairness in recommendation*, in SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022, E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, and G. Kazai, eds., ACM, 2022, pp. 681–691.

[57] Y. GE, X. ZHAO, L. YU, S. PAUL, D. HU, C.-C. HSIEH, AND Y. ZHANG, *Toward pareto efficient fairness-utility trade-off in recommendation through reinforcement learning*, in Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 316–324.

[58] F. GEDIKLI, D. JANNACH, AND M. GE, *How should i explain? a comparison of different explanation types for recommender systems*, International Journal of Human-Computer Studies, 72 (2014), pp. 367–382.

[59] S. GENG, S. LIU, Z. FU, Y. GE, AND Y. ZHANG, *Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)*, in Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 299–315.

[60] S. GENG, J. TAN, S. LIU, Z. FU, AND Y. ZHANG, *Vip5: Towards multimodal foundation models for recommendation*, arXiv preprint arXiv:2305.14302, (2023).

[61] 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.

[62] A. GHAZIMATIN, R. SAHA ROY, AND G. WEIKUM, *Fairy: A framework for understanding relationships between users' actions and their social feeds*, in Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 240–248.

[63] A. GILOTTE, C. CALAUZÈNES, T. NEDELEC, A. ABRAHAM, AND S. DOLLÉ, *Offline a/b testing for recommender systems*, in Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 198–206.

[64] D. GOLDBERG, D. NICHOLS, B. M. OKI, AND D. TERRY, *Using collaborative filtering to weave an information tapestry*, Communications of the ACM, 35 (1992), pp. 61–70.

[65] A. GRAVES, *Long short-term memory*, in Supervised sequence labelling with recurrent neural networks, Springer, 2012, pp. 37–45.

[66] A. GROVER AND J. LESKOVEC, *node2vec: Scalable feature learning for networks*, in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2016, pp. 855–864.

[67] A. GRUSON, P. CHANDAR, C. CHARBUILLET, J. MCINERNEY, S. HANSEN, D. TARDIEU, AND B. CARTERETTE, *Offline evaluation to make decisions about playlistrecommendation algorithms*, in Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 420–428.

[68] Z. GUAN, C. WANG, J. BU, C. CHEN, K. YANG, D. CAI, AND X. HE, *Document recommendation in social tagging services*, in Proceedings of the 19th international conference on World wide web, 2010, pp. 391–400.

[69] G. GUO, J. ZHANG, AND N. YORKE-SMITH, *Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings*, in Proceedings of the AAAI conference on artificial intelligence, vol. 29, 2015.

[70] H. GUO, R. TANG, Y. YE, Z. LI, AND X. D. HE, *a factorization-machine based neural network for ctr prediction. arxiv 2017*, arXiv preprint arXiv:1703.04247.

[71] P. GUPTA, A. SHARMA, P. MALHOTRA, L. VIG, AND G. SHROFF, *Causer: Causal session-based recommendations for handling popularity bias*, in Proceedings of the 30th ACM international conference on information & knowledge management, 2021, pp. 3048–3052.

[72] S. GUPTA, H. WANG, Z. LIPTON, AND Y. WANG, *Correcting exposure bias for link recommendation*, in International Conference on Machine Learning, PMLR, 2021, pp. 3953–3963.

[73] O. H. HAMID, *From model-centric to data-centric ai: A paradigm shift or rather a complementary approach?*, in 2022 8th International Conference on Information Technology Trends (ITT), IEEE, 2022, pp. 196–199.

[74] W. HAMILTON, Z. YING, AND J. LESKOVEC, *Inductive representation learning on large graphs*, Advances in neural information processing systems, 30 (2017).

[75] 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.

[76] T. U. HAQUE, N. N. SABER, AND F. M. SHAH, *Sentiment analysis on large scale amazon product reviews*, in 2018 IEEE international conference on innovative research and development (ICIRD), IEEE, 2018, pp. 1–6.

[77] L. HE, H. CHEN, D. WANG, S. JAMEEL, P. YU, AND G. XU, *Click-through rate prediction with multi-modal hypergraphs*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 690–699.

[78] R. HE AND J. MCAULEY, *Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering*, in proceedings of the 25th international conference on world wide web, 2016, pp. 507–517.

[79] 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, CIKM '15, New York, NY, USA, 2015, Association for Computing Machinery, p. 1661–1670.

[80] 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.

[81] 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.

[82] X. HE, J. PAN, O. JIN, T. XU, B. LIU, T. XU, Y. SHI, A. ATALLAH, R. HERBRICH, S. BOWERS, ET AL., *Practical lessons from predicting clicks on ads at facebook*, in Proceedings of the eighth international workshop on data mining for online advertising, 2014, pp. 1–9.

[83] X. HE, H. ZHANG, M.-Y. KAN, AND T.-S. CHUA, *Fast matrix factorization for online recommendation with implicit feedback*, in Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 549–558.

[84] J. L. HERLOCKER, J. A. KONSTAN, AND J. RIEDL, *Explaining collaborative filtering recommendations*, in Proceedings of the 2000 ACM conference on Computer supported cooperative work, 2000, pp. 241–250.

[85] J. M. HERNÁNDEZ-LOBATO, N. HOULSBY, AND Z. GHAHRAMANI, *Probabilistic matrix factorization with non-random missing data*, in International Conference on Machine Learning, 2014, pp. 1512–1520.

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

[87] D. W. HOSMER JR, S. LEMESHOW, AND R. X. STURDIVANT, *Applied logistic regression*, vol. 398, John Wiley & Sons, 2013.

[88] 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.

[89] J. HU, B. HOOI, S. QIAN, Q. FANG, AND C. XU, *Mgdcf: Distance learning via markov graph diffusion for neural collaborative filtering*, IEEE Transactions on Knowledge and Data Engineering, (2024).

[90] Y. HU, Y. KOREN, AND C. VOLINSKY, *Collaborative filtering for implicit feedback datasets*, in 2008 Eighth IEEE International Conference on Data Mining, Ieee, 2008, pp. 263–272.

[91] Z. HU, Y. DONG, K. WANG, AND Y. SUN, *Heterogeneous graph transformer*, in Proceedings of The Web Conference 2020, 2020, pp. 2704–2710.

[92] W. HUANG, K. LABILLE, X. WU, D. LEE, AND N. HEFFERNAN, *Achieving user-side fairness in contextual bandits*, Human-Centric Intelligent Systems, 2 (2022), pp. 81–94.

[93] 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.

[94] O. JEUNEN AND B. GOETHALS, *Pessimistic reward models for off-policy learning in recommendation*, in Fifteenth ACM Conference on Recommender Systems, 2021, pp. 63–74.

[95] O. JEUNEN, D. ROHDE, F. VASILE, AND M. BOMPAIRE, *Joint policy-value learning for recommendation*, in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1223–1233.

[96] B. JIANG, T. LIN, S. MA, AND S. ZHANG, *Structured nonconvex and nonsmooth optimization: algorithms and iteration complexity analysis*, Computational Optimization and Applications, 72 (2019), pp. 115–157.

[97] T. JOACHIMS, A. SWAMINATHAN, AND M. DE RIJKE, *Deep learning with logged bandit feedback*, in International Conference on Learning Representations, 2018.

[98] 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.

[99] T. KAMISHIMA, S. AKAHO, H. ASOH, AND J. SAKUMA, *Efficiency improvement of neutrality-enhanced recommendation.*, in Decisions@ RecSys, Citeseer, 2013, pp. 1–8.

[100] ——, *Correcting popularity bias by enhancing recommendation neutrality.*, RecSys posters, 10 (2014).

[101] S. KANWAL, S. NAWAZ, M. K. MALIK, AND Z. NAWAZ, *A review of text-based recommendation systems*, IEEE Access, 9 (2021), pp. 31638–31661.

[102] C. KARAKO AND P. MANGGALA, *Using image fairness representations in diversity-based re-ranking for recommendations*, in Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, 2018, pp. 23–28.

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

[104] ——, *Variational graph auto-encoders*, arXiv preprint arXiv:1611.07308, (2016).

[105] J. A. KONSTAN, B. N. MILLER, D. MALTZ, J. L. HERLOCKER, L. R. GORDON, AND J. RIEDL, *Grouplens: Applying collaborative filtering to usenet news*, Communications of the ACM, 40 (1997), pp. 77–87.

[106] Y. KOREN, *Factorization meets the neighborhood: a multifaceted collaborative filtering model*, in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 426–434.

[107] Y. KOREN AND R. BELL, *Advances in collaborative filtering*, in Recommender systems handbook, Springer, pp. 77–118.

[108] Y. KOREN, R. BELL, AND C. VOLINSKY, *Matrix factorization techniques for recommender systems*, Computer, 42 (2009), pp. 30–37.

[109] V. KULESHOV, A. CHAGANTY, AND P. LIANG, *Tensor factorization via matrix factorization*, in Artificial Intelligence and Statistics, PMLR, 2015, pp. 507–516.

[110] S. KUMAR, S. DATTA, V. SINGH, S. K. SINGH, AND R. SHARMA, *Opportunities and challenges in data-centric ai*, IEEE Access, (2024).

[111] R. LAI, L. CHEN, R. CHEN, AND C. ZHANG, *A survey on data-centric recommender systems*, arXiv preprint arXiv:2401.17878, (2024).

[112] Y. LEI, H. PEI, H. YAN, AND W. LI, *Reinforcement learning based recommendation with graph convolutional q-network*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1757–1760.

[113] P. LI, Z. WANG, Z. REN, L. BING, AND W. LAM, *Neural rating regression with abstractive tips generation for recommendation*, in Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, 2017, pp. 345–354.

[114] Q. LI, X. WANG, Z. WANG, AND G. XU, *Be causal: De-biasing social network confounding in recommendation*, ACM Trans. Knowl. Discov. Data, (2022).

[115] Q. LI, X. WANG, AND G. XU, *Be causal: De-biasing social network confounding in recommendation*, arXiv preprint arXiv:2105.07775, (2021).

[116] W. LI, M. GAO, W. RONG, J. WEN, Q. XIONG, R. JIA, AND T. DOU, *Social recommendation using euclidean embedding*, in 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 589–595.

[117] Y. LI, H. CHEN, Z. FU, Y. GE, AND Y. ZHANG, *User-oriented fairness in recommendation*, in Proceedings of the Web Conference 2021, 2021, pp. 624–632.

[118] Y. LI, H. CHEN, S. XU, Y. GE, J. TAN, S. LIU, AND Y. ZHANG, *Fairness in recommendation: A survey*, arXiv preprint arXiv:2205.13619, (2022).

[119] Z. LI, H. XIE, G. XU, Q. LI, M. LENG, AND C. ZHOU, *Towards purchase prediction: A transaction-based setting and a graph-based method leveraging price information*, Pattern Recognition, 113 (2021), p. 107824.

[120] D. LIANG, L. CHARLIN, AND D. M. BLEI, *Causal inference for recommendation*, in Causation: Foundation to Application, Workshop at UAI, AUAI, 2016.

[121] D. LIANG, R. G. KRISHNAN, M. D. HOFFMAN, AND T. JEBARA, *Variational autoencoders for collaborative filtering*, in Proceedings of the 2018 world wide web conference, 2018, pp. 689–698.

[122] W. LIN, S. A. ALVAREZ, AND C. RUIZ, *Collaborative recommendation via adaptive association rule mining*, Data Mining and Knowledge Discovery, 6 (2000), pp. 83–105.

[123] G. LINDEN, B. SMITH, AND J. YORK, *Amazon. com recommendations: Item-to-item collaborative filtering*, IEEE Internet computing, 7 (2003), pp. 76–80.

[124] R. J. LITTLE AND D. B. RUBIN, *Statistical analysis with missing data*, vol. 793, John Wiley & Sons, 2019.

[125] B. LIU, A. SHETH, U. WEINSBERG, J. CHANDRASHEKAR, AND R. GOVINDAN, *Adreveal: Improving transparency into online targeted advertising*, in Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, 2013, pp. 1–7.

[126] F. LIU, R. TANG, X. LI, W. ZHANG, Y. YE, H. CHEN, H. GUO, Y. ZHANG, AND X. HE, *State representation modeling for deep reinforcement learning based recommendation*, Knowledge-Based Systems, 205 (2020), p. 106170.

[127] Q. LIU, F. YAN, X. ZHAO, Z. DU, H. GUO, R. TANG, AND F. TIAN, *Diffusion augmentation for sequential recommendation*, in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 1576–1586.

[128] W. LIU, F. LIU, R. TANG, B. LIAO, G. CHEN, AND P. A. HENG, *Balancing between accuracy and fairness for interactive recommendation with reinforcement learning*, in Pacific-asia conference on knowledge discovery and data mining, Springer, 2020, pp. 155–167.

[129] S.-C. B. LO, H.-P. CHAN, J.-S. LIN, H. LI, M. T. FREEDMAN, AND S. K. MUN, *Artificial convolution neural network for medical image pattern recognition*, Neural networks, 8 (1995), pp. 1201–1214.

[130] H. LUO, F. ZHUANG, R. XIE, H. ZHU, D. WANG, Z. AN, AND Y. XU, *A survey on causal inference for recommendation*, The Innovation, (2024).

[131] 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.

[132] H. MA, D. ZHOU, C. LIU, M. R. LYU, AND I. KING, *Recommender systems with social regularization*, in Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 287–296.

[133] J. MA, Z. ZHAO, X. YI, J. YANG, M. CHEN, J. TANG, L. HONG, AND E. H. CHI, *Off-policy learning in two-stage recommender systems*, in Proceedings of The Web Conference 2020, 2020, pp. 463–473.

[134] J. MA, C. ZHOU, P. CUI, H. YANG, AND W. ZHU, *Learning disentangled representations for recommendation*, NeurIPS, (2019).

[135] M. MANSOURY, H. ABDOLLAHPOURI, M. PECHENIZKIY, B. MOBASHER, AND R. BURKE, *Fairmatch: A graph-based approach for improving aggregate diversity in recommender systems*, in Proceedings of the 28th ACM conference on user modeling, adaptation and personalization, 2020, pp. 154–162.

[136] B. M. MARLIN AND R. S. ZEMEL, *Collaborative prediction and ranking with non-random missing data*, in Proceedings of the third ACM conference on Recommender systems, 2009, pp. 5–12.

[137] J. MCAULEY, C. TARGETT, Q. SHI, AND A. VAN DEN HENGEL, *Image-based recommendations on styles and substitutes*, in Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, 2015, pp. 43–52.

[138] B. MCMAHAN, *Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization*, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 525–533.

[139] A. MNIH AND R. R. SALAKHUTDINOV, *Probabilistic matrix factorization*, in Advances in neural information processing systems, 2008, pp. 1257–1264.

[140] C. MORRIS, M. RITZERT, M. FEY, W. L. HAMILTON, J. E. LENSSEN, G. RATTAN, AND M. GROHE, *Weisfeiler and leman go neural: Higher-order graph neural networks*, in Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 4602–4609.

[141] R. K. MOTHILAL, A. SHARMA, AND C. TAN, *Explaining machine learning classifiers through diverse counterfactual explanations*, in Proceedings of the 2020 conference on fairness, accountability, and transparency, 2020, pp. 607–617.

[142] S. MU, Y. LI, W. X. ZHAO, S. LI, AND J.-R. WEN, *Knowledge-guided disentangled representation learning for recommender systems*, ACM Transactions on Information Systems (TOIS), 40 (2021), pp. 1–26.

[143] A. MÜLLER, *Integral probability metrics and their generating classes of functions*, Advances in Applied Probability, (1997), pp. 429–443.

[144] N. MURRAY AND F. PERRONNIN, *Generalized max pooling*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 2473–2480.

[145] H. NARASIMHAN, A. COTTER, M. GUPTA, AND S. WANG, *Pairwise fairness for ranking and regression*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 5248–5255.

[146] Y. NARITA, S. YASUI, AND K. YATA, *Debiased off-policy evaluation for recommendation systems*, in Fifteenth ACM Conference on Recommender Systems, 2021, pp. 372–379.

[147] T. T. NGUYEN, P.-M. HUI, F. M. HARPER, L. TERVEEN, AND J. A. KONSTAN, *Exploring the filter bubble: the effect of using recommender systems on content diversity*, in Proceedings of the 23rd international conference on World wide web, 2014, pp. 677–686.

[148] X. NI, F. XIONG, Y. ZHENG, AND L. WANG, *Graph contrastive learning with kernel dependence maximization for social recommendation*, in Proceedings of the ACM on Web Conference 2024, 2024, pp. 481–492.

[149] C. NÓBREGA AND L. MARINHO, *Towards explaining recommendations through local surrogate models*, in Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019, pp. 1671–1678.

[150] S. OHSAWA, Y. OBARA, AND T. OSOGAMI, *Gated probabilistic matrix factorization: learning users' attention from missing values*, in Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 1888–1894.

[151] R. PAN, Y. ZHOU, B. CAO, N. N. LIU, R. LUKOSE, M. SCHOLZ, AND Q. YANG, *One-class collaborative filtering*, in 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 502–511.

[152] A. PAPADIMITRIOU, P. SYMEONIDIS, AND Y. MANOLOPOULOS, *A generalized taxonomy of explanations styles for traditional and social recommender systems*, Data Mining and Knowledge Discovery, 24 (2012), pp. 555–583.

[153] J. PEARL, *Causal inference in statistics: An overview*, Statistics surveys, 3 (2009), pp. 96–146.

[154] ——, *Causality*, Cambridge university press, 2009.

[155] J. PEARL AND D. MACKENZIE, *The book of why: the new science of cause and effect*, Basic books, 2018.

[156] M. PERC, *The matthew effect in empirical data*, Journal of The Royal Society Interface, 11 (2014), p. 20140378.

[157] J. QIN, K. REN, Y. FANG, W. ZHANG, AND Y. YU, *Sequential recommendation with dual side neighbor-based collaborative relation modeling*, in Proceedings of the 13th international conference on web search and data mining, 2020, pp. 465–473.

[158] Y. RAITA, C. A. CAMARGO JR, L. LIANG, AND K. HASEGAWA, *Big data, data science, and causal inference: A primer for clinicians*, Frontiers in Medicine, 8 (2021), p. 678047.

[159] S. RENDLE, *Factorization machines*, in 2010 IEEE International conference on data mining, IEEE, 2010, pp. 995–1000.

[160] ——, *Factorization machines*, in 2010 IEEE International Conference on Data Mining, 2010, pp. 995–1000.

[161] S. RENDLE, C. FREUDENTHALER, Z. GANTNER, AND L. SCHMIDT-THIEME, *Bpr: Bayesian personalized ranking from implicit feedback*, in Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, Arlington, Virginia, USA, 2009, AUAI Press, p. 452–461.

[162] ——, *Bpr: Bayesian personalized ranking from implicit feedback*, arXiv preprint arXiv:1205.2618, (2012).

[163] Y. S. RESHEFF, Y. ELAZAR, M. SHAHAR, AND O. S. SHALOM, *Privacy and fairness in recommender systems via adversarial training of user representations*, arXiv preprint arXiv:1807.03521, (2018).

[164] P. RESNICK, N. IACOVOU, M. SUCHAK, P. BERGSTROM, AND J. RIEDL, *Grouplens: An open architecture for collaborative filtering of netnews*, in Proceedings of the 1994 ACM conference on Computer supported cooperative work, 1994, pp. 175–186.

[165] M. RICHARDSON, E. DOMINOWSKA, AND R. RAGNO, *Predicting clicks: estimating the click-through rate for new ads*, in Proceedings of the 16th international conference on World Wide Web, 2007, pp. 521–530.

[166] K. J. ROTHMAN AND S. GREENLAND, *Causation and causal inference in epidemiology*, American journal of public health, 95 (2005), pp. S144–S150.

[167] D. B. RUBIN, *Estimating causal effects of treatments in randomized and nonrandomized studies.*, Journal of educational Psychology, 66 (1974), p. 688.

[168] Y. SAITO AND T. JOACHIMS, *Counterfactual learning and evaluation for recommender systems: Foundations, implementations, and recent advances*, in Proceedings of the 15th ACM Conference on Recommender Systems, 2021, pp. 828–830.

[169] 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.

[170] J. B. SCHAFER, D. FRANKOWSKI, J. HERLOCKER, AND S. SEN, *Collaborative filtering recommender systems*, in The adaptive web, Springer, 2007, pp. 291–324.

[171] T. SCHNABEL, A. SWAMINATHAN, A. SINGH, N. CHANDAK, AND T. JOACHIMS, *Recommendations as treatments: Debiasing learning and evaluation*, arXiv preprint arXiv:1602.05352, (2016).

[172] M. SCHUSTER AND K. K. PALIWAL, *Bidirectional recurrent neural networks*, IEEE transactions on Signal Processing, 45 (1997), pp. 2673–2681.

[173] U. SHALIT, F. D. JOHANSSON, AND D. SONTAG, *Estimating individual treatment effect: generalization bounds and algorithms*, in International Conference on Machine Learning, PMLR, 2017, pp. 3076–3085.

[174] C. SHI, B. HU, W. X. ZHAO, AND S. Y. PHILIP, *Heterogeneous information network embedding for recommendation*, IEEE transactions on knowledge and data engineering, 31 (2018), pp. 357–370.

[175] C. SHI, Y. LI, J. ZHANG, Y. SUN, AND S. Y. PHILIP, *A survey of heterogeneous information network analysis*, IEEE Transactions on Knowledge and Data Engineering, 29 (2016), pp. 17–37.

[176] C. SHI, Z. ZHANG, Y. JI, W. WANG, P. S. YU, AND Z. SHI, *Semrec: a personalized semantic recommendation method based on weighted heterogeneous information networks*, World Wide Web, 22 (2019), pp. 153–184.

[177] 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.

[178] Y. SHI, A. KARATZOGLOU, L. BALTRUNAS, M. LARSON, N. OLIVER, AND A. HANJALIC, *Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering*, in Proceedings of the Sixth ACM Conference on Recommender

Systems, RecSys '12, New York, NY, USA, 2012, Association for Computing Machinery, p. 139–146.

[179] E. SHULMAN AND L. WOLF, *Meta decision trees for explainable recommendation systems*, in Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 365–371.

[180] J. SINGH AND A. ANAND, *Posthoc interpretability of learning to rank models using secondary training data*, CoRR, abs/1806.11330 (2018).

[181] H. STECK, *Training and testing of recommender systems on data missing not at random*, in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 713–722.

[182] ——, *Evaluation of recommendations: rating-prediction and ranking*, in Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 213–220.

[183] X. SU AND T. M. KHOSHGOFTAAR, *A survey of collaborative filtering techniques*, Advances in artificial intelligence, 2009 (2009).

[184] J. SUN, Z. CHENG, S. ZUBERI, F. PÉREZ, AND M. VOLKOVS, *Hgcf: Hyperbolic graph convolution networks for collaborative filtering*, in Proceedings of the Web Conference 2021, 2021, pp. 593–601.

[185] Y. SUN AND J. HAN, *Mining heterogeneous information networks: a structural analysis approach*, Acm Sigkdd Explorations Newsletter, 14 (2013), pp. 20–28.

[186] R. S. SUTTON, D. MCALLESTER, S. SINGH, AND Y. MANSOUR, *Policy gradient methods for reinforcement learning with function approximation*, Advances in neural information processing systems, 12 (1999).

[187] A. SWAMINATHAN AND T. JOACHIMS, *Batch learning from logged bandit feedback through counterfactual risk minimization*, The Journal of Machine Learning Research, 16 (2015), pp. 1731–1755.

[188] ——, *Counterfactual risk minimization: Learning from logged bandit feedback*, in International Conference on Machine Learning, PMLR, 2015, pp. 814–823.

[189] 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.

[190] J. TANG, H. GAO, AND H. LIU, *mTrust: Discerning multi-faceted trust in a connected world*, in Proceedings of the fifth ACM international conference on Web search and data mining, ACM, 2012, pp. 93–102.

[191] J. TANG, X. HU, AND H. LIU, *Social recommendation: a review*, Social Network Analysis and Mining, 3 (2013), pp. 1113–1133.

[192] N. TINTAREV AND J. MASTHOFF, *Evaluating the effectiveness of explanations for recommender systems*, User Modeling and User-Adapted Interaction, 22 (2012), pp. 399–439.

[193] K. H. TRAN, A. GHAZIMATIN, AND R. SAHA ROY, *Counterfactual explanations for neural recommenders*, in Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1627–1631.

[194] R. VAN METEREN AND M. VAN SOMEREN, *Using content-based filtering for recommendation*, in Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop, vol. 30, Barcelona, 2000, pp. 47–56.

[195] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, Ł. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, Advances in neural information processing systems, 30 (2017).

[196] P. VELIČKOVIĆ, G. CUCURULL, A. CASANOVA, A. ROMERO, P. LIO, AND Y. BENGIO, *Graph attention networks*, arXiv preprint arXiv:1710.10903, (2017).

[197] S. VERMA, J. DICKERSON, AND K. HINES, *Counterfactual explanations for machine learning: A review*, arXiv preprint arXiv:2010.10596, (2020).

[198] J. VIG, S. SEN, AND J. RIEDL, *Tagsplanations: explaining recommendations using tags*, in Proceedings of the 14th international conference on Intelligent user interfaces, 2009, pp. 47–56.

[199] C. WANG, T. ZHOU, C. CHEN, T. HU, AND G. CHEN, *Off-policy recommendation system without exploration*, Advances in Knowledge Discovery and Data Mining, 12084 (2020), p. 16.

[200] 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.

[201] S. WANG, Y. WANG, J. TANG, K. SHU, S. RANGANATH, AND H. LIU, *What your images reveal: Exploiting visual contents for point-of-interest recommendation*, in Proceedings of the 26th international conference on world wide web, 2017, pp. 391–400.

[202] W. WANG, F. FENG, X. HE, L. NIE, AND T.-S. CHUA, *Denoising implicit feedback for recommendation*, in Proceedings of the 14th ACM international conference on web search and data mining, 2021, pp. 373–381.

[203] W. WANG, F. FENG, X. HE, X. WANG, AND T.-S. CHUA, *Deconfounded recommendation for alleviating bias amplification*, arXiv preprint arXiv:2105.10648, (2021).

[204] 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.

[205] X. WANG, X. HE, M. WANG, F. FENG, AND T.-S. CHUA, *Neural graph collaborative filtering*, in Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019, pp. 165–174.

[206] X. WANG, H. JI, C. SHI, B. WANG, Y. YE, P. CUI, AND P. S. YU, *Heterogeneous graph attention network*, in The World Wide Web Conference, 2019, pp. 2022–2032.

[207] X. WANG, H. JIN, A. ZHANG, X. HE, T. XU, AND T.-S. CHUA, *Disentangled graph collaborative filtering*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1001–1010.

[208] X. WANG, Q. LI, D. YU, P. CUI, Z. WANG, AND G. XU, *Causal disentanglement for semantics-aware intent learning in recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2022).

[209] ——, *Causal disentanglement for semantics-aware intent learning in recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2022).

[210] X. WANG, Q. LI, D. YU, Q. LI, AND G. XU, *Constrained off-policy learning over heterogeneous information for fairness-aware recommendation*, ACM Trans. Recomm. Syst., (2023).
Just Accepted.

[211] ——, *Counterfactual explanation for fairness in recommendation*, ACM Transactions on Information Systems, (2023).

[212] ——, *Reinforced path reasoning for counterfactual explainable recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2024).

[213] ——, *Reinforced path reasoning for counterfactual explainable recommendation*, IEEE Transactions on Knowledge and Data Engineering, (2024), pp. 1–17.

[214] X. WANG, Q. LI, D. YU, Z. WANG, H. CHEN, AND G. XU, *Mgpolicy: Meta graph enhanced off-policy learning for recommendations*, in Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, New York, NY, USA, 2022, Association for Computing Machinery, p. 1369–1378.

[215] X. WANG, Q. LI, D. YU, AND G. XU, *Off-policy learning over heterogeneous information for recommendation*, in Proceedings of the ACM Web Conference 2022, WWW '22, New York, NY, USA, 2022, Association for Computing Machinery, p. 2348–2359.

[216] X. WANG, Q. LI, W. ZHANG, G. XU, S. LIU, AND W. ZHU, *Joint relational dependency learning for sequential recommendation*, in Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2020, pp. 168–180.

[217] 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.

[218] X. WANG, Y. XU, X. HE, Y. CAO, M. WANG, AND T.-S. CHUA, *Reinforced negative sampling over knowledge graph for recommendation*, in Proceedings of The Web Conference 2020, 2020, pp. 99–109.

[219] X. WANG, R. ZHANG, Y. SUN, AND J. QI, *Doubly robust joint learning for recommendation on data missing not at random*, in International Conference on Machine Learning, 2019, pp. 6638–6647.

[220] Y. WANG, D. LIANG, L. CHARLIN, AND D. M. BLEI, *The deconfounded recommender: A causal inference approach to recommendation*, arXiv preprint arXiv:1808.06581, (2018).

[221] ——, *Causal inference for recommender systems*, in Proceedings of the 14th ACM Conference on Recommender Systems, 2020, pp. 426–431.

[222] Y. WANG, W. MA, M. ZHANG*, Y. LIU, AND S. MA, *A survey on the fairness of recommender systems*, ACM Journal of the ACM (JACM), (2022).

[223] Y. WANG, S. TANG, Y. LEI, W. SONG, S. WANG, AND M. ZHANG, *Disenhan: Disentangled heterogeneous graph attention network for recommendation*, in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1605–1614.

[224] Z. WANG AND A. C. BOVIK, *Mean squared error: Love it or leave it? a new look at signal fidelity measures*, IEEE signal processing magazine, 26 (2009), pp. 98–117.

[225] 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*, in Proceedings of the 17th ACM International Conference on Web Search and Data Mining, 2024, pp. 806–815.

[226] S. WEISBERG, *Applied linear regression*, vol. 528, John Wiley & Sons, 2005.

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

[228] J. WOODWARD, *Making Things Happen: A Theory of Causal Explanation*, Oxford Studies in the Philosophy of Science, Oxford University Press, New York, 2004.

[229] F. WU, A. SOUZA, T. ZHANG, C. FIFTY, T. YU, AND K. WEINBERGER, *Simplifying graph convolutional networks*, in International conference on machine learning, PMLR, 2019, pp. 6861–6871.

[230] B. XIA, T. LI, Q. LI, AND H. ZHANG, *Noise-tolerance matrix completion for location recommendation*, Data Mining and Knowledge Discovery, 32 (2018), pp. 1–24.

[231] K. XIA, K.-Z. LEE, Y. BENGIO, AND E. BAREINBOIM, *The causal-neural connection: Expressiveness, learnability, and inference*, Advances in Neural Information Processing Systems, 34 (2021), pp. 10823–10836.

[232] T. XIAO, Z. CHEN, AND S. WANG, *Reconsidering learning objectives in unbiased recommendation: A distribution shift perspective*, in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 2764–2775.

[233] T. XIAO AND D. WANG, *A general offline reinforcement learning framework for interactive recommendation*, in The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, 2021.

[234] 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.

[235] Y. XIE, Z. XU, J. ZHANG, Z. WANG, AND S. JI, *Self-supervised learning of graph neural networks: A unified review*, IEEE transactions on pattern analysis and machine intelligence, 45 (2022), pp. 2412–2429.

[236] 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.

[237] J. XU, Z. LI, B. DU, M. ZHANG, AND J. LIU, *Reluplex made more practical: Leaky relu*, in 2020 IEEE Symposium on Computers and communications (ISCC), IEEE, 2020, pp. 1–7.

[238] K. XU, C. LI, Y. TIAN, T. SONOBE, K.-I. KAWARABAYASHI, AND S. JEGELKA, *Representation learning on graphs with jumping knowledge networks*, in International Conference on Machine Learning, PMLR, 2018, pp. 5453–5462.

[239] S. XU, J. TAN, Z. FU, J. JI, S. HEINECKE, AND Y. ZHANG, *Dynamic causal collaborative filtering*, in Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 2301–2310.

[240] Y. YANG, M. LI, X. HU, G. PAN, W. HUANG, J. WANG, AND Y. WANG, *Exploring exposure bias in recommender systems from causality perspective*, in 2021 IEEE

21st International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2021, pp. 425–432.

[241] L. YAO, S. LI, Y. LI, M. HUAI, J. GAO, AND A. ZHANG, *Representation learning for treatment effect estimation from observational data*, Advances in Neural Information Processing Systems, 31 (2018).

[242] S. YAO AND B. HUANG, *Beyond parity: Fairness objectives for collaborative filtering*, Advances in neural information processing systems, 30 (2017).

[243] M. YIN AND M. ZHOU, *Semi-implicit variational inference*, in International Conference on Machine Learning, PMLR, 2018, pp. 5660–5669.

[244] R. YING, R. HE, K. CHEN, P. EKSOMBATCHAI, W. L. HAMILTON, AND J. LESKOVEC, *Graph convolutional neural networks for web-scale recommender systems*, in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 974–983.

[245] J. YU, M. GAO, J. LI, H. YIN, AND H. LIU, *Adaptive implicit friends identification over heterogeneous network for social recommendation*, in Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 357–366.

[246] X. YU, X. REN, Q. GU, Y. SUN, AND J. HAN, *Collaborative filtering with entity similarity regularization in heterogeneous information networks*, IJCAI HINA, 27 (2013).

[247] 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.

[248] H. ZAMANI AND W. B. CROFT, *Learning a joint search and recommendation model from user-item interactions*, in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 717–725.

[249] M. ZEČEVIĆ, D. S. DHAMI, P. VELIČKOVIĆ, AND K. KERSTING, *Relating graph neural networks to structural causal models*, arXiv preprint arXiv:2109.04173, (2021).

[250] H. ZENATI, A. BIETTI, M. MARTIN, E. DIEMERT, AND J. MAIRAL, *Counterfactual learning of continuous stochastic policies*, arXiv preprint arXiv:2004.11722, (2020).

[251] D. ZHA, Z. P. BHAT, K.-H. LAI, F. YANG, Z. JIANG, S. ZHONG, AND X. HU, *Data-centric artificial intelligence: A survey*, arXiv preprint arXiv:2303.10158, (2023).

[252] C. ZHANG, K. ZHANG, AND Y. LI, *A causal view on robustness of neural networks*, Advances in Neural Information Processing Systems, 33 (2020), pp. 289–301.

[253] J. ZHANG, X. CHEN, AND W. X. ZHAO, *Causally attentive collaborative filtering*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 3622–3626.

[254] S. ZHANG, Z. HAN, Y.-K. LAI, M. ZWICKER, AND H. ZHANG, *Stylistic scene enhancement gan: mixed stylistic enhancement generation for 3d indoor scenes*, The Visual Computer, 35 (2019), pp. 1157–1169.

[255] W. ZHANG, T. CHEN, J. WANG, AND Y. YU, *Optimizing top-n collaborative filtering via dynamic negative item sampling*, in Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, 2013, pp. 785–788.

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

[257] Y. ZHANG, F. FENG, X. HE, T. WEI, C. SONG, G. LING, AND Y. ZHANG, *Causal intervention for leveraging popularity bias in recommendation*, arXiv preprint arXiv:2105.06067, (2021).

[258] 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.

[259] Z. ZHANG AND M. SABUNCU, *Generalized cross entropy loss for training deep neural networks with noisy labels*, Advances in neural information processing systems, 31 (2018).

[260] X. ZHAO, L. ZHANG, Z. DING, L. XIA, J. TANG, AND D. YIN, *Recommendations with negative feedback via pairwise deep reinforcement learning*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1040–1048.

[261] Y. ZHENG, C. GAO, X. LI, X. HE, Y. LI, AND D. JIN, *Disentangling user interest and conformity for recommendation with causal embedding*, in Proceedings of the Web Conference 2021, WWW '21, New York, NY, USA, 2021, Association for Computing Machinery, p. 2980–2991.

[262] ——, *Disentangling user interest and conformity for recommendation with causal embedding*, in Proceedings of the Web Conference 2021, 2021, pp. 2980–2991.

[263] S. ZHOU, X. DAI, H. CHEN, W. ZHANG, K. REN, R. TANG, X. HE, AND Y. YU, *Interactive recommender system via knowledge graph-enhanced reinforcement learning*, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 179–188.

[264] Z. ZHU, Y. HE, X. ZHAO, AND J. CAVERLEE, *Popularity bias in dynamic recommendation*, in Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 2439–2449.

[265] L. ZOU, L. XIA, P. DU, Z. ZHANG, T. BAI, W. LIU, J.-Y. NIE, AND D. YIN, *Pseudo dyna-q: A reinforcement learning framework for interactive recommendation*, in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 816–824.