User Representation Learning for Personalized Recommendation Systems

by

Chenglin Li

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Recommendation systems have become an indispensable part of our lives, providing an effective solution to information overload and enhancing user satisfaction by suggesting the most relevant content. In recent years, deep learning has facilitated the creation of recommendation technologies that rely on sophisticated deep neural networks to learn intricate user representations, resulting in remarkable success. However, the existence of persistent issues, such as data sparsity and various biases, presents significant obstacles for deep neural methods in the pursuit of creating effective and unbiased recommendation systems.

Cross-domain recommendation (CDR) is a promising approach that leverages data and knowledge from auxiliary domains to improve the performance of recommender systems when the data in the target domains is sparse. To address the challenge of cross-domain sequential recommendation with limited overlapping users, we developed a novel CDR method inspired by the real-world needs of industrial companies. Our method offers an effective solution to this problem. Furthermore, we extended our CDR approach to multi-target scenarios, where the objective is to enhance recommendation performance for three or more domains simultaneously, and we tackled the issue of negative transfer in this context.

The utilization of a temporal heterogeneous graph is a compelling technique for capturing the intricate interactions between users and items in recommendation systems. In order to achieve precise sequential recommendations through the use of temporal graphs, we introduce a novel continuous-time representation learning model that can extract high-quality user and item representations from a temporal heterogeneous information network. Moreover, to address the issue of biases in recommendation systems, we propose an unbiased sequential recommendation model that incorporates the potential outcome framework (POF) and employs a disentangled graph transformer on a temporal user-item interaction graph to enhance performance.

Preface

This dissertation proposes models to address data sparsity and bias issues for accurate user representation learning in various recommendation scenarios. Chapter 1 provides background information and motivations, while Chapter 2 reviews related literature. The remaining sections of this thesis are the result of collaborative work with Dr. Di Niu and other co-authors.

Chapter 3 has been published as "RecGURU: Adversarial learning of generalized user representations for cross-domain recommendation" by Li, Chenglin, et al. in the Proceedings of the fifteenth ACM international conference on web search and data mining in 2022.

Chapter 4 has been published as "One for All, All for One: Learning and Transferring User Embeddings for Cross-Domain Recommendation" by Li, Chenglin, et al. in the Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining in 2023.

Chapters 5 and 6 are the result of collaborative efforts with Yuanzhen Xie and Tao Xie from Tencent, as well as my advisor, Dr. Di Niu. These chapters are currently under submission.

To my family.

All things in their being are good for something.

Acknowledgments

I would like to express my sincere gratitude to all those who have contributed in some way to the work described in this dissertation. First and foremost, I am immensely thankful to my advisor, Professor Di Niu, for his invaluable advice, unwavering support, and remarkable patience during my Ph.D. study. His vast knowledge and extensive experience have been a constant source of inspiration and guidance for me throughout my academic research and daily life.

I am also grateful to Professor Li Cheng and Xingyu Li for their invaluable contributions as members of my supervisor committee. Their insightful feedback and constructive criticism have greatly enriched the quality of this dissertation. I would like to express my gratitude to my final defence committee Dr. Kui Wu and Lili Mou for taking the time to review my Ph.D. thesis and for providing valuable feedback and suggestions. Your constructive comments have helped me to refine my work and to deepen my understanding of the subject matter. Moreover, thank Professor Jie Chen for his time hosting the exam. His dedication and support throughout the procedure were greatly appreciated. I would also like to extend my heartfelt thanks to Professor Lei Ma and Zhigang Tian for their invaluable contributions as committee members of my Ph.D. candidacy exam. Their insightful feedback and rigorous evaluation have helped me to grow as a researcher and prepare for this important milestone in my academic journey.

My friends and colleagues at school have been an invaluable source of support throughout this journey, and I am deeply appreciative of their help. I want to express my gratitude to my friends, including but not limited to Yaochen Hu, Rui Zhu, Haolan Chen, Dashun Wang, Lingju Meng, Ting Zhang, Fred X. Han, Mingjun

Zhao, Yushi Wang, Jiuding Yang, Keith Mills, Qikai Lu, Yakun Yu, Liyao Jiang, and others who have been there for me. In particular, I collaborated with Rui Zhu, Mingjun Zhao, Jiuding Yang, Keith Mills, and Liyao Jiang, and I am grateful for their precious suggestions and insights into our work.

I am also very thankful for the support of Tencent and Wedge Networks Inc. during my research projects and internships. These experiences have been invaluable to me, and I have learned so much from my colleagues at both companies. I would like to express my appreciation to Huanming Zhang, Yuanzhen Xie, Tao Xie, BeiBei Kong, Jianming Yang, Lei Cheng, Mingcheng Yang, Hao Yin, Shihui Yang, and others at Tencent for their support of my internship. Additionally, I want to thank Hongwen Zhang, Husam Kinawi, and Shijia at Wedge for their support during my internship there.

I am immensely grateful to my parents, Zhengxiu Shui and Daiwen Li, for teaching me how to be a strong-minded individual. I also owe a debt of gratitude to my elder brother Maolin Li and his wife Lijuan Zhang for their love and support. My wife, Musiqing Yao, has been a pillar of strength throughout this journey, and I am so thankful for her unwavering love. I want to extend my appreciation to all my friends and family members who have supported me along the way. Finally, I am indebted to my grandparents Tongshu Luo and Shisong Li for raising me with care, love, and education. Grandpa, I will always miss you dearly.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Notations

Acronyms/Notation	**Definition**
RS	Recommendation System
CDR	Cross-domain Recommendation
STCDR	Single-target Cross-domain Recommendation
DTCDR	Dual-target Cross-domain Recommendation
MTCDR	Multi-target Cross-domain Recommendation
MF	Matrix Factorization
GNN	Graph Neural Networks
RNN	Recurrent Neural Networks
POF	Potential Outcome Framework
PS	Propensity Score
IPS	Inverse Propensity Score
GRU	Gated Recurrent Unit
MNAR	Missing Not At Randon
FFN	Fully-connected feedforward layer
MLP	Multilayer perceptron
$[eos]$	The End of Sequence token
$\frown$	Concatenation operator between two row vectors

Chapter 1

Introduction

In today's era of rapid internet growth, recommendation systems have become increasingly crucial in helping people manage information overload. These systems are designed to leverage data-driven algorithms and suggest relevant items to users across a range of domains, including products [1], [2], music [3], and movies [4], [5]. The primary goal of a recommendation system is to offer personalized and related items based on users' interests, preferences, and interactions [6]. Traditionally, either or both Collaborative Filtering [7], [8] and content-based filtering [9] are used to achieve personalized recommendations. In recent decades, machine learning algorithms have been developed to further improve recommendation performance, including shallow models like Matrix Factorization [10] and Factorization Machines [11], as well as deep models such as Neural Collaborative Filtering (NCF) [12], Deep & Wide [13], and AutoInt [14].

Sequential recommendation has taken the concept of general recommendation to the next level by explicitly modelling the correlations between users' successive behaviours. It has been hugely successful in modelling user preferences and has made significant strides with the introduction of advanced deep learning models, such as recurrent neural networks (RNNs) [15], attention mechanisms [16], and Graph Neural Networks (GNN) [17].

1.1 Data Sparsity and Cross-Domain Recommendation

Data sparsity is a well-known challenge in recommendation systems, occurring when the dataset contains only a few interactions between users and items. This can make it difficult to generate accurate recommendations based on limited data. Furthermore, data sparsity may result in overfitting or cold-start issues if there is insufficient data available to train the recommender effectively. Unfortunately, the data sparsity problem has been a persistent issue that can hinder recommendations within a single domain.

The cross-domain recommendation (CDR) has been proposed to alleviate the data sparsity issue by leveraging user behaviour in multiple domains to help recommendations in the target domain and has attracted much attention in both academia and industry. Recent work on cross-domain recommendation focuses on the transfer learning of user and item information from diverse perspectives. For example, mapping functions have been proposed to map user representations from one domain to another, by learning from the behaviour of users that appear in both domains [18]. Nevertheless, a common limitation of existing cross-domain recommendation methods is that they perform transfer learning primarily based on data of the overlapped users, and fail to function well when there are few or even no overlapped users in two domains [18]. However, in many real-world applications, there is often not a sufficient number of overlapped users.

Moreover, most prior research on cross-domain recommendation focuses on either the single-target CDR (STCDR) or dual-target CDR (DTCDR) [19] scenarios, with only two domains involved. STCDR aims to improve the recommendation accuracy in the target domain, while DTCDR tries to improve the performance in both domains simultaneously. Multi-target CDR (MTCDR) is a more general and challenging problem, which aims to improve the recommendation performance in multiple participating domains concurrently. Since previous methods solve the

STCDR and DTCDR by modelling a pair-wise domain-domain relationship, intuitively speaking, extending them to the MTCDR scenario with n domains involves handling at least $\binom{n}{2}$ pairs of relations, which is not practical when the number of domains is large.

Relatively fewer efforts have been put into MTCDR. Current state-of-the-art solutions usually generate a shared cross-domain user representation for each user, which, combined with domain-specific features, is used to boost recommendations in any given domain [20], [21]. HeroGRAPH [20] collects user behaviour in all domains to build a heterogeneous graph. It then applies the graph convolutional networks (GCN) [22] to generate the cross-domain user and item embeddings, which are directly transferred to target domains to boost recommendation. However, most recommender systems are built on users' sensitive data, e.g., check-in data, and browsing records, which are held by different domains and cannot be shared directly to form a large heterogeneous graph. MPF [21] learns a global embedding for each user, which is directly shared among all domains and optimized by the recommendation losses in all participating domains via multi-task learning. However, the cross-domain user representation, extracted directly from the collected data from all domains, may be severely biased by the domains with richer data and may fail to model the user preferences in sparse domains. The biased global representation of a user may negatively affect recommendation performance when transferred to a target domain. This unbalanced data problem also exists in HeroGRAPH [20].

1.2 Temporal Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as the new state-of-the-art approach for many recommendation problems, thanks to their strong ability to handle structured data and explore high-order information [23]. For capturing users' behaviour histories in recommendation systems, a temporal graph (or dynamic graph), which is represented as an ordered list or asynchronous "stream" of timed events, such as the addition or deletion of nodes and edges, is a natural data structure. There-

fore, recent research efforts have focused on developing effective temporal graph methods to enhance the performance of sequential recommendations [24]. Many real-world tasks can often be expressed as temporal heterogeneous graphs. However, accurately learning graph representation while dealing with the heterogeneity and temporal evolution of graph structure poses significant challenges.

1.3 Bias and Debias in Recommendation

Despite the success of sequential recommendation in modelling user preferences, with advanced deep-learning models such as Markov chains [25], recurrent neural networks (RNNs) [15], attention mechanisms [16], and Graph Neural Network (GNN) [17] being widely implemented, real-world scenarios often introduce biases such as selection bias and exposure bias which lead to missing not-at-random (MNAR) data that is skewed from the ideal distribution. While previous methods have achieved remarkable results based on observational data instead of experimental data, this could lead to issues such as the long-tail effect if recommendation performance is only optimized on this biased observed dataset.

Recent studies have attempted to address the bias problem in recommendation systems through techniques from the causal inference field, for example, the potential outcome framework (POF) and inverse propensity scoring (IPS). The effectiveness of these approaches depends on an accurate estimation of the propensity score (PS). In traditional methods, a small set of unbiased datasets is often required for the estimation of PS [26]. Some recent work has extended IPS to enable unbiased sequential recommendations. The difference between traditional methods and this approach lies in the specific model used for estimating PS. For example, USR [27] utilizes a GRU module which takes the historical item sequence of each user as input to learn its current representation and ultimately for estimating the propensity score. DEPS [28] further extends this by incorporating user sequences for the representation learning of items and achieves the so-called dually enhanced propensity score estimation. In sequential recommendations, obtaining unbiased datasets for

estimating PS (as traditional methods do) is extremely difficult or even impossible. Thus, USR and DEPS train their PS modules on biased observed datasets. This can lead to inaccurate estimation of PS and consequently biased recommendations.

Additionally, previous debias solutions focus solely on debiasing via an unbiased loss function while ignoring the importance of extracting unbiased representations of users and items which we argue is the essential and ultimate goal for the unbiased recommendation. For example, USR [27] and DEPS [28] rely on modelling users' preferences from their historical interaction records, without taking into account that such records may have been generated in a biased recommendation environment. This means that directly modelling the history sequence may result in biased user and item representations which cannot accurately represent users' preferences and degrade the recommendation performance. Moreover, few attempts have been made to tackle the selection bias in sequential recommendations with explicit user ratings.

1.4 Contributions and Thesis Outline

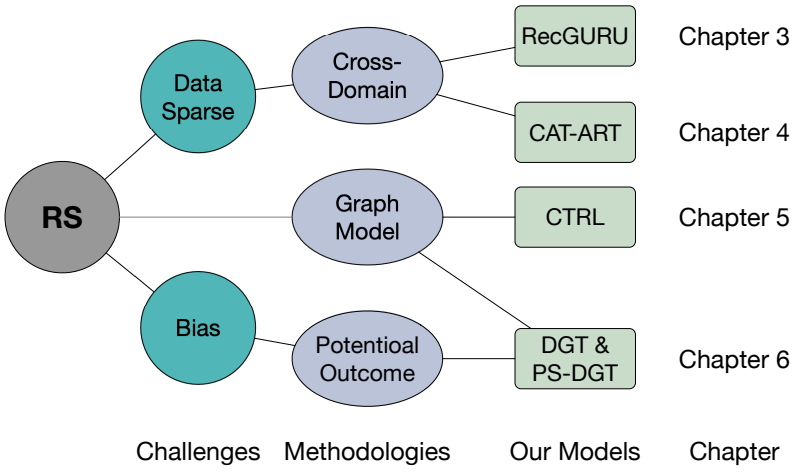


Figure 1.1. The framework of the components in this dissertation.

Our work addresses the challenges of data sparsity and bias in recommendation scenarios, making significant contributions to the field. Figure 1.1 provides

an overview of our proposed method, which involves three methodologies, cross-domain recommendation, temporal heterogeneous graph, and potential outcome framework, applied across four different recommendation scenarios. By tackling these challenges, we aim to improve user representation learning in RS. Specifically, we discuss the dual-target cross-domain sequential recommendation scenario in Chapter 3 [29] and we have made the following contributions:

- We propose a dual-target cross-domain sequential recommendation RecGURU which achieves state-of-the-art performance.
- Instead of knowledge transferring which requires user features from both domains, RecGURU learns global user representations through adversarial training which can be applied to non-overlapped users.
- We propose an effective and stable training procedure for RecGURU.

In Chapter 4, we extend the dual-target scenario into the multi-target cross-domain recommendation (MTCDR) [30] and our contributions are:

- We identified two beneficial embeddings for MTCDR: global user embeddings and domain-specific user embeddings in source domains.
- We propose a contrastive autoencoder (CAT) module to extract unbiased global user representations.
- An attention-based representation transfer (ART) unit is built in each target domain to integrate domain-specific user embeddings from source domains.
- We effectively avoid negative transfer in MTCDR.

In Chapter 5, we propose a representation learning algorithm from temporal heterogeneous information networks. Our contributions are:

- We propose an edge-based Hawks process to model the dynamic impact of historical neighbours of a node.

- We integrate the dynamic node centrality to weigh the importance of the neighbours of a node in the local aggregation of a GNN layer.
- We train the model by predicting the occurrence of temporal events (predefined subgraph structures) to capture the evolution of higher-order topological structures.

In Chapter 6, we discuss the bias and bias methods in recommendations and propose a novel disentangled graph transformer to alleviate the biases problems. Specifically, we have made the following contribution in this chapter.

- We studied the explicit feedback-based sequential recommendation and formulate the bias problem within the potential outcome framework.
- We propose a disentangled graph transformer (DGT) model for handling the selection bias.
- We introduce a new propensity score estimation module for unbiased training objectives and calibrating interaction records of users.

Finally, we conclude the dissertation in Chapter 7.

Chapter 2

Related Work

2.1 Cross-Domain Recommendation

2.1.1 Sequential Recommendation

The sequential recommendation has gained popularity and attracted more and more attention in recent years due to its capability of capturing the sequential behaviour patterns of users compared to traditional method [6], [31]. Early studies adopt Markov Chains to capture sequential patterns from users' historical interactions [25], [32]. Methods based on Recurrent Neural networks (RNN) and attention mechanisms have also been proposed, for example, GRU4Rec [33] and its improved version GRU4Rec+ [15] leverage Gated Recurrent Unit (GRU) with BPR loss to model a user's sequential behaviours. In SAS [16], unidirectional self-attention is adopted here to encode a user's historical behaviour. Bert4Rec [34] follows the idea of BERT and trained a bidirectional self-attention model. SHAN [35] adopts a hierarchical attention framework where two attention networks are used to model users' long- and short-term preferences. Graph neural networks and MLP models are also adopted in recent work, for example, FMLP-Rec [36] proposes an all-MLP model with trainable filters for the sequential recommendation and achieves decent performance. [37] adopt sequential hypergraphs to model the dynamic user preferences in the next-item recommendations. [17], [38] leverages graph neural network (GNN)

for sequential and session-based recommendation tasks. TGSRec [24] adopt time encoding and temporal graph neural network for the time-aware sequential recommendation. Some recent work focus on denoising and data augmentation for better modelling of sequential patterns, e.g., Causerec [39] and [40], [41]. However, most previous methods focus on sequential recommendation with implicit feedback (clicks). In this chapter, we focus on the problem of unbiased sequential recommendation with explicit user feedback and propose novel disentangled representations of users and items to present their preferences and dislikes.

2.1.2 Cross-domain Sequential Recommendation

Traditional Cross-domain Recommendation. Cross-domain recommendation alleviates data sparsity issues posed by single-domain recommendation via auxiliary information from other domains. CoNet [42] transfers and combines knowledge across different domains through cross-connections between feed-forward neural networks. [43] proposes to transfer the item embeddings across domains to avoid leakages of user privacy. However, these methods only focus on overlapped users. CATN [44] solves the cold-start problem via aspect transfer which requires side information of both users and items. [45] leverages meta-transfer learning to address the sparsity problem, but it requires multiple contextual information such as the user's average spend.

To transfer knowledge from the source domain to the target domain, EMCDR [18] learns a mapping function on overlapped users which maps user preferences across domains. DCDCSR [46] maps the latent factors in the target domain to fit the benchmark factors which combines the features in both the target and source domains. To reduce the dependency on overlapped users, SSCDR [47] adopts a semi-supervised strategy. DDTCDR [48] and its improved version DOML[49] adopt dual metric learning (DML). The DOML requires side information to get embeddings of users and items, however, we focus on the implicit feedback-based cross-domain sequential recommendation problem in this chapter.

Generative Adversarial Network (GAN) [50] is gaining popularity in cross-

domain recommendation[51], [52], [52]–[54]. CnGAN [51] introduces the use of the GAN to learn a better mapping function of user representations from the source domain to the target domain. Since the discriminator in CnGAN is trained to distinguish between real and synthetically mapped pairs, where the real mapped pairs only come from overlapped users. Thus, CnGAN is critically dependent on the quantity and quality of overlapped users, which usually cannot be guaranteed in reality. Additionally, the CnGAN tries to stabilize the training of GAN by introducing more synthetically mapped pairs. However, this makes the input data to the discriminator unbalanced and thus may fail to train a good mapping function. Similarly, RecSys-dan [52] adopts an adversarial approach to transfer representations of users or items from the source domain to the target domain. [53] adopt adversarial samples in the training process to improve the generalization ability of the cross-domain recommender system. ATLRec [54] transfers shareable features across domains, however, it focuses on overlapped users which show up in both the target and source domains.

Cross-domain Sequential Recommendation. There are also several studies in the field of cross-domain sequential recommendation. π-Net [55] is able to generate recommendations for both domains through a cross-domain transfer unit. However, it requires synchronously shared timestamps and can not be applied to non-overlapped users. [56] transfers users' novelty-seeking properties learned from the sequential data in the source domain to the target domain. [57] proposed a framework that is able to fine-tune a large pre-trained user embedding network to adapt to downstream tasks in the target domain. However, these studies have different problem settings from our work. And they can only be applied to overlapped users, but our method can handle both overlapped and non-overlapped users.

2.1.3 Single-Target Cross-Domain Recommendation

Previous single-target CDR (STCDR) works can be classified into content-based [58]–[60] and embedding-based approaches[18], [44], [47], [61]. In the former, [58] uses a deep neural network to learn the preferences of users based on their

search queries in different domains. [59] proposes a content-based domain adaptation model and a domain separation network for cross-domain recommendations. [60] proposes a deep domain adaptation model which only relies on the rating metrics in each domain for the cross-domain recommendation. In the latter, EMCDR [18] solves the cold-start problem in the target domain by learning a mapping function between the user embeddings of the source and the target domain. Semi-supervised learning is adopted for scenarios with limited user overlapping between domains [47]. CDIE-C [61] enhances item embedding learning by cross-domain co-clustering for the sequential recommendation.

2.1.4 Dual-Target Cross-Domain Recommendation

Dual-target CDR is a relatively new recommendation scenario in CDR, and it has attracted increasing attention in recent years. Given two domains, DTCDR is to improve the recommendation accuracy in both domains at the same time by leveraging their observed information[48], [49], [62]–[66]. [62] first proposed the DTCDR problem and a DTCDR framework that learns more representative embeddings of users and items based on multi-sources such as rating, review, user profile, item details, etc. Then, it combines and shares the embedding of common users across domains with three different strategies. Then, it combines and shares the embedding of common users across domains with three different strategies. [63] combine the embeddings of common users based on a fixed strategy, i.e., hyper-parameters and data sparsity degrees of common users. GA-DTCDR [64] employs graph embedding to generate more informative embeddings of users and items and employs element-wise attention to combine the embeddings of common users/items across domains. The deep dual transfer CDR (DDTCDR) [48] considers the bi-directional latent relations between users and items and applies a latent orthogonal mapping to extract user preferences. CATN [44] learns aspect correlations across domains with an attention mechanism. Some work focuses on extracting domain-invariant or domain-independent user attributes for CDR [67]–[69]. ACDN [68] models an individual's propensity from the aesthetic perspective and captures users' domain-

independent aesthetic preference for CDR.

2.1.5 Multi-Target Cross-Domain Recommendation

Although multi-target CDR is inspired by dual-target CDR, it aims to achieve a bigger goal, i.e. solving the data sparsity problem for all participating domains at the same time. The MTCDR methods [20], [21], [45], [57], [70], [71] have emerged to improve the recommendation performance of multiple domains simultaneously. [70] adopt recurrent neural networks (RNNs) to model the sequential behaviour of users in multiple domains simultaneously. [57], [71] achieve knowledge transfer through parameter sharing across multiple domains. However, they all focus on sequential recommendations. Recent works try to extract domain-specific and cross-domain features simultaneously, e.g., MSDCR [72], HeroGRAPH [20], and MPF [21]. Specifically, HeroGRAPH [20] constructs a heterogeneous graph from interactions between users and items from all domains and develops a graph embedding algorithm to extract common features for MTCDR. MPF [21] captures both the cross-site and site-specific preferences for multi-site video recommendations. MMT-Net [45] extracts contextual invariances across domains and transfers auxiliary information from a source domain to improve the recommendations in multiple domains. However, it requires extra contextual information. GA-MTCDR [73], extended from GA-DTCDR [64], employs element-wise attention to combine embeddings of overlapped users/items from all domains. However, it requires side information for graph construction in each domain. However, most of the previous MTCDR methods ignore the data isolation constraint between domains in practice, and none of them has considered the negative transfer problem. In this study, we try to solve the MTCDR problem in a more realistic scenario where user and item data are held by each individual domain and cannot be shared across domains. Furthermore, our framework is also designed to avoid the negative transfer problem in MTCDR.

2.2 Unbiased Sequential Recommendation

2.2.1 Debias in Recommendation

Recommendation systems (RS) face a variety of biases including selection bias [74]–[76], position bias [77], [78], exposure bias [79]–[83], and conformity bias [84] etc. Bias elimination (debias) in recommendation systems, has become a new trend and great research efforts have been put into introducing causal inference techniques such as the potential output framework and inverse propensity weighting into the RS for unbiased recommendation [85], For examples, [86], [87] leverage the Inverse Propensity Score (IPS) for an unbiased recommendation. [88] also corrects the exposure bias through IPS. [89] introduces the doubly robust method in recommendation systems to reduce the high variance caused by the PS method and get a more robust debias model. Counterfactual thinking from causal inference is also widely adopted in debias models [90]–[93]. Autodebias [26] combines the IPS and imputation approaches and proposes a universal debias framework for all types of bias in the recommendation. [94].

2.2.2 Debias in Sequential Recommendation

USR [27] first extends the debias problem into sequential recommendation and proposes to estimate propensity scores through a GRU channel based on the historical item sequence of users. DEPS [28] further takes the user's perspective into account when estimating the propensity score for a better and more robust sequential debias model. Unlike the traditional IPS-based method where a small unbiased dataset is collected in the training process to ensure the correctness of the IPS estimation, the USR and DEPS methods directly optimize their IPS modules on observational data which may result in a biased IPS and recommendation. However, collecting a bias-free dataset in the sequential recommendation scenario with explicit user feedback is extremely difficult (if impossible). In this chapter, we propose a novel IPS estimation module which first decomposes the propensity score of interactions into

user rating propensity, item rating propensity, and user-item correlation propensity and leverages the prior distributions of user and item propensities for a more stable and accurate PS estimation.

2.3 Representation Learning on Temporal Heterogeneous Graph

Graph embedding tries to represent nodes in a low-dimensional space while preserving node features and the topological structures of the graph [95], [96]. Traditional methods mainly focus on the representation learning on static homogeneous networks, e.g., Deepwalk [97], LINE [98], GCN [99], GAT [100], and do not consider the evolution of graphs.

Recent graph embedding studies focus on two more practical scenarios, i.e., the heterogeneous information network (HIN) and temporal networks. For HIN, traditional shallow methods model semantics and structures based on meta-path [101], e.g., Meta-path2vec [102] and HAN [103], while deep models incorporate heterogeneous node/edge information into the graph neural network through different techniques such as attention mechanism HetGNN [104] and HGT [105]. For temporal networks, most of the existing works resort to taking snapshots of the continuous-time temporal networks, and model the dynamics of node embeddings from a sequence of snapshots, e.g., DySAT [106], MTSN [107], and EvolveGCN [108]. Recent work focuses more on modelling the evolution of networks in a continuous-time manner, e.g., TGAT [109], TREND [110], CAW [111], and HVGNN [112]. Besides, some work focuses on exploring the evolution patterns of temporal graphs such as the triadic closure process [113] in social networks. There are also some works that aim at providing real-time node embedding services from temporal networks, e.g., APAN [114], TGL [115] and TGN [116].

There is an increasing trend that focuses on temporal HIN embedding. Most existing works use meta-path to capture semantics and structures in HIN and take

snapshots of the temporal network to model the dynamics of node embeddings, like DHNE [117], Change2vec [118], and DyHNE [119]. Other than taking snapshots, HDGAN [120] leverages time-level attention to model network evolution. Besides, some work takes advantage of the Hawkes process [121] to simulate the evolution of the temporal network, e.g., THINE [122] and HPGE [123] for HIN, and HTNE [124], MMDNE [125], and TREND [110] for monograph. However, these methods are not inductive to new nodes, thus, there is still a lack of deep methods for temporal HINs. TGSRec [24] applied the TGAT model to the sequential recommendation problem but does not explicitly consider the influence of node heterogeneity. Effectively modelling the temporal evolution of heterogeneous graph structures poses significant challenges for both graph representation learning and recommendation systems. To fill this gap, in this thesis, we propose a deep graph neural network for continuous-time representation learning on temporal HINs.

Chapter 3

Cross-Domain Sequential Recommendation

3.1 Introduction

In this chapter, we present RecGURU to address the data sparsity problem in the single-domain sequential recommendation by leveraging knowledge from a source domain, in which there are only a few overlapping users between the target domain and source domain. RecGURU consists of two parts: the Generalized User Representation Unit (GURU) to obtain a single Generalized User Representation (GUR) for each user, and the Cross-Domain Sequential Recommendation (CDSRec) unit to achieve cross-domain collaboration in sequential recommendation tasks. Instead of mapping user embeddings from one domain to another, we propose to generalize a user's embedding, i.e., its GUR, to incorporate information from both domains through an adversarial learning framework. Once a GUR is obtained for each user, we integrate the extracted GUR into the CDSRec unit using attention mechanisms to boost recommendation performance. Specifically, we make the following contributions:

First, in the GURU module, an autoencoder is proposed to generate informative user embeddings in each domain, which are to be unified later into a generalized embedding, the GUR, with adversarial learning. The autoencoder consists of a

self-attentive encoder with model weights shared across both the source and target domains to produce latent user embeddings and two decoders to reconstruct behaviour sequences of users in the source and target domains, respectively. We train the autoencoder through behaviour sequence reconstructions to generate meaningful preliminary embeddings for users with unsupervised self-learning.

The GURU module further performs adversarial training to unify domain-dependent user embeddings into a single global (GUR) for each user, which is domain-independent. Specifically, the encoder part of the proposed autoencoder serves as a generator to produce user embeddings, while a discriminator is trained to identify the origin domain of a generated embedding for a user randomly sampled from either the source or the target domain. The encoder and discriminator are trained alternately with adversarial objectives until the discriminator can not distinguish which domain a given user embedding comes from. This is when the user embeddings in the two domains become statistically indistinguishable and GURs are supposed to be generalized global user embeddings, incorporating information from both domains. This method does not rely on common users present in both target and source domains, therefore eliminating the dependency on overlapped users as required by the prior art.

Furthermore, we introduce an effective and stable training procedure for RecGURU, consisting of three phases. We first pre-train the autoencoder to substantially reduce the reconstruction loss which boost-starts the subsequent adversarial learning. In the adversarial learning phase, the reconstruction loss is further jointly optimized in a multi-task fashion which prevents the encoder from generating wild representations, stabilizing adversarial learning. In the meantime, RecGURU can still leverage overlapped users as prior work does, by introducing an l_2 penalty in the optimization procedure to explicitly force each common user to have the same shared embedding in different domains. Finally, the CDSRec module, which incorporates the GUR with attention mechanisms, is fine-tuned in the target domain with the next-item recommendation task to boost the recommendation performance.

Through extensive experiments, we show that RecGURU has achieved improve-

ment in the sequential recommendation, compared to several state-of-the-art single-domain and cross-domain recommendation methods. Specifically, we outperform all the baselines on various metrics by a large margin on the Amazon datasets including "Sport", "Clothing", "Movie", and "Book". Additionally, we have collected a large cross-domain recommendation dataset with two domains, i.e. “Wesee” and “Tencent Video”, from two real-world applications which provide video streams to millions of users. Ablation studies on the collected datasets with various portions of overlapped users are conducted to show the effectiveness of each proposed sub-module as well as the robustness of our method. The collected datasets will be made public to facilitate future research in the field of cross-domain and sequential recommendation.

3.2 Methodology

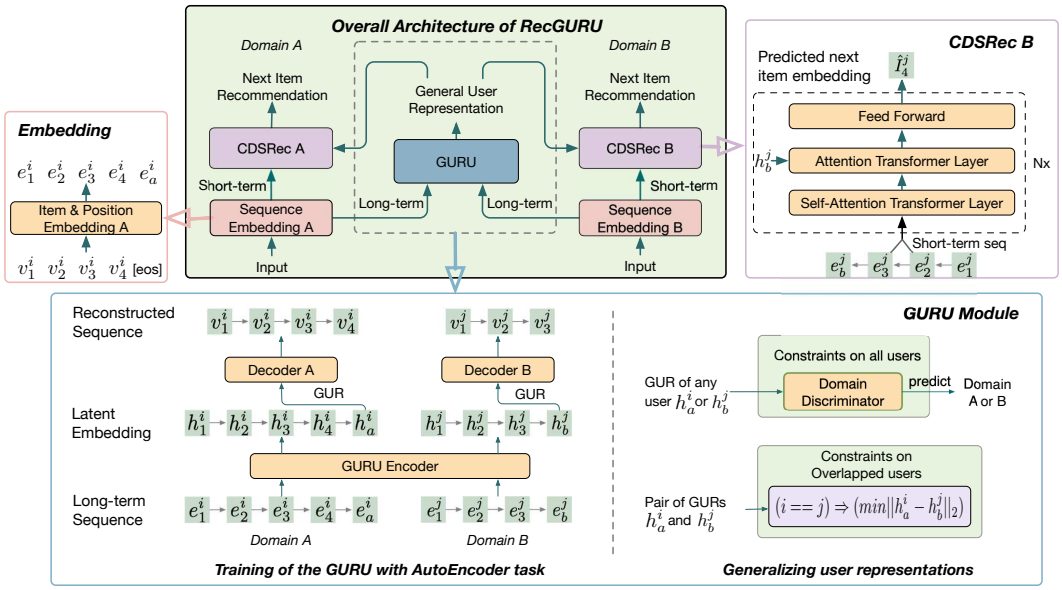


Figure 3.1. Model structure of the proposed RecGURU. User behaviour sequences in both domain A and B are fed into the GURU encoder to generate generalized latent user representations, i.e. $\boldsymbol{h}_a^i$ and $\boldsymbol{h}_b^i$. Then, the generated GUR is fed into the CDSRec model for the next-item recommendation in each individual domain.

3.2.1 Problem Definition

We first formulate the implicit feedback-based sequential recommendation problem in a single domain. Let $U = \{u_1, \cdots, u_{|U|}\}$ and $V = \{v_1, \cdots, v_{|V|}\}$ denote the sets of users and items, respectively, where $|U|$ and $|V|$ are the total number of users and items. For each user $u_i, \forall i \in \{1, \cdots, |U|\}$, its interactions with items, in chronological order, are denoted as $s^i = (v_1^i, \cdots, v_{|s^i|}^i)$, where $|s^i|$ is the length of behaviour sequence s^i. Formally speaking, given a user u_i, sequential recommendation aims to predict the next item that the user is most likely to interact with at the next time step $|s^i|+1$ based on his or her past behaviour sequence s^i, which can be formalized as modelling the probability over all items:

$$p(v_{|s^i|+1}^i = v|s^i). \tag{3.1}$$

In cross-domain scenarios, in order to improve the recommendation performance, user information from other domains is also taken into account. Specifically, for two domains A and B, given an overlapped user who appears in both domains, $u_i \in (U_A \cap U_B)$, cross-domain sequential recommendation tries to improve the recommendation accuracy of the next item in one domain by integrating user information from both domains. For example, for next-item recommendation in domain A, it can be formulated as modelling the probability over all possible candidates given the behaviour sequences s_a^i and s_b^i in domains A and B:

$$p(v_{|s_A^i|+1}^i = v|s_A^i, s_B^i). \tag{3.2}$$

For non-overlapped users, due to a lack of their behaviour in both domains, we can only exploit the implicit information from the other domain to enhance the recommendation performance in the target domain. Therefore, Equation (3.2) is reformulated as

$$p(v_{|s_A^i|+1}^i = v|s_A^i, \textit{info}(S_B)), \tag{3.3}$$

where $S_B = \{s_B^j\}, \forall u_j \in U_B$ denotes the collection of behaviour sequences for all users in domain B and *info*$(\cdot)$ represent a model which is used to extract any kinds of useful information from S_B to assist recommendation in domain A.

3.2.2 Overview of Proposed Method

Inspired by previous studies in single domain sequential recommendations which try to combine the long-term (static) and short-term (dynamic) preferences of users for next-item recommendation [35], [126], [127], we propose a novel RecGURU framework. It consists of two parts: a generalized user representation unit (GURU) which learns the generalized user representation across domains and a cross-domain sequential recommendation (CDSRec) unit which takes the GUR as input to achieve cross-domain collaboration for the next-item recommendation task. Here, the GUR represents the static preference containing the user information in both domains.

The overall architecture of the proposed model is shown in Figure 3.1. The GURU takes the long-term behaviour sequence as input and generates generalized representations for all users from both domains. Then, the short-term user behaviour together with the extracted general representation is fed into the CDSRec module to enhance sequential recommendation in each individual domain. To construct a GUR, we propose an adversarial training framework that involves an autoencoder for generating informative user representations and a domain discriminator to unify the representations across different domains. Furthermore, we apply an additional l_2 regularizer on overlapped users to further ensure each overlapped user has similar representations in both domains.

Note that, motivated by the idea of the cross-lingual language models [128], we adopt a shared encoder across domains to avoid over-parameterization and achieve further information sharing via parameter-sharing across domains.

Figure 3.1 gives a further example of how input sequences are processed to produce GURs and next-item recommendations. Specifically, the behaviour sequences $(v_1^i, v_2^i, v_3^i, v_4^i, [eos])$ and $(v_1^j, v_2^j, v_3^j, [eos])$ of user $u_i \in U_A$ and $u_j \in U_B$, where $[eos]$ denotes the end of sequence token, are fed into two individual embedding lay-

ers to produce the embedding vectors $(\boldsymbol{e}_1^i, \boldsymbol{e}_2^i, \boldsymbol{e}_3^i, \boldsymbol{e}_4^i, \boldsymbol{e}_a^i)$ and $(\boldsymbol{e}_1^j, \boldsymbol{e}_2^j, \boldsymbol{e}_3^j, \boldsymbol{e}_b^j)$ containing both item information and sequence position information. Then, the GURU encoder takes the embedded sequences as inputs and generates the latent user representations h_a^i and h_b^j of users u_i and u_j. To make sure the user representation is meaningful and informative, decoders are applied to reconstruct the original input sequences, such that, the encoder and decoder in each domain form an autoencoder framework. Furthermore, in order to get generalized representations which combine information from both the source and target domains, the domain discriminator is applied to h_a^i and h_b^j to pull the distributions of user representation in the two domains close to each other. Hereafter, in each domain, the sequential recommendation model combines the GUR and the short-term user behaviours to generate the next-item recommendation, for example, the predicted item embeddings $\hat{\boldsymbol{I}}_4^j$ at the 4th timestamp of user u_j in domain B.

3.2.3 Generalized User Representations

As shown in Figure 3.1, in the GURU module, user representations are learned in each individual domain, and information from the other domain is incorporated to make the representation "general" through regularizer achieved by domain discriminator on latent user representation.

User Representations in Single Domain. In each individual domain, we use an autoencoder to learn the latent user representation that is capable of reconstructing the original input sequence of the user. As compared to extracting user representations from the next-item prediction task, the autoencoder can produce meaningful representations through the reconstruction task to boost performance. The autoencoder in our framework consists of an embedding module, an encoder module, and a decoder module.

Formally, to extract a representation for any given user u_i with its behaviour sequence $s^i = (v_1^i, \cdots, v_t^i, \cdots, v_{|s^i|}^i)$, we apply the following autoencoder proce-

dures:

$$\begin{aligned} \boldsymbol{e}_1^i, \cdots, \boldsymbol{e}_t^i, \cdots, \boldsymbol{e}_{|s^i|}^i, \boldsymbol{e}^i &= \text{Embed}(v_1^i, \cdots, v_t^i, \cdots, v_{|s^i|}^i, [eos]), \\ \boldsymbol{h}_1^i, \cdots, \boldsymbol{h}_t^i, \cdots, \boldsymbol{h}_{|s^i|}^i, \boldsymbol{h}^i &= \text{Encoder}(\boldsymbol{e}_1^i, \cdots, \boldsymbol{e}_t^i, \cdots, \boldsymbol{e}_{|s^i|}^i, \boldsymbol{e}^i), \\ v_1^i, \cdots, v_t^i, \cdots, v_{|s^i|}^i &= \text{Decoder}(\boldsymbol{h^i}), \end{aligned} \tag{3.4}$$

where $\boldsymbol{h}_t^i$ is the latent representation of item v_t^i at position t, and $\boldsymbol{e}_t^i$ represents the sum of item embedding and positional embedding. $\boldsymbol{e^i}$ and $\boldsymbol{h^i}$ are the embedding and latent representation of the $[eos]$ token. We use bold font to indicate vector variables.

The input sequence of items and the $[eos]$ token are converted into real-valued vectors through the *Embed* module, which consists of an item embedding layer and a positional embedding layer to incorporate both the item information and the sequential information of behaviour sequences. To this end, we create two trainable embedding matrices $\mathbf{I} \subset \mathbb{R}^{(|\mathbf{V}|+\mathbf{1})\times \mathbf{d}}$ and $\mathbf{P} \subset \mathbb{R}^{(\mathbf{N}+\mathbf{1})\times \mathbf{d}}$ for the item and positional embeddings, respectively, where d represents the number of dimensions in the latent space and $N+1$ is the maximum length of input sequences including the $[eos]$ token. By summing up the output of item embeddings and positional embeddings with the point-wise summation, we derive the embedding representations for all items and the $[eos]$ token, denoted as $(\boldsymbol{e}_1^i, \cdots, \boldsymbol{e}_t^i, \cdots, \boldsymbol{e}_{|s^i|}^i, \boldsymbol{e}^i)$ shown in Equation (3.4).

After the derivation of embeddings, we adopt the autoencoder to obtain the fixed-length representations of the behaviour sequences of users. Specifically, we adopt an encoder with a structure similar to *Transformer* [129] which is composed of a stack of identical transformer layers. Each transformer layer is composed of a multi-head, bidirectional self-attention layer, and a position-wise fully connected feed-forward layer. Shown in Equation (3.4), the *Encoder* outputs a list of latent vectors $(\boldsymbol{h}_1^i, \cdots, \boldsymbol{h}_{|s^i|}^i, \boldsymbol{h}^i)$ for all input items in the input sequence. Then, the latent vector of the $[eos]$ token $\boldsymbol{h}^i$ is treated as the representation of the whole behaviour sequence, i.e. user representation of u_i, and is further fed into the decoder.

We adopt a decoder that also consists of multiple transformer layers. In addition

to the self-attention and feed-forward sub-layers in each transformer layer, another multi-head attention layer over the user representation $\boldsymbol{h^i}$ is inserted right after the self-attention layer in each decoder layer. Additionally, the masking technique is applied to ensure that the predictions depend only on previous behaviour sequences. Formally, shown in Equation (3.4), our *Decoder* takes the latent user representation $\boldsymbol{h}^i$ as input and reconstructs the original input sequence in an auto-regressive manner [129].

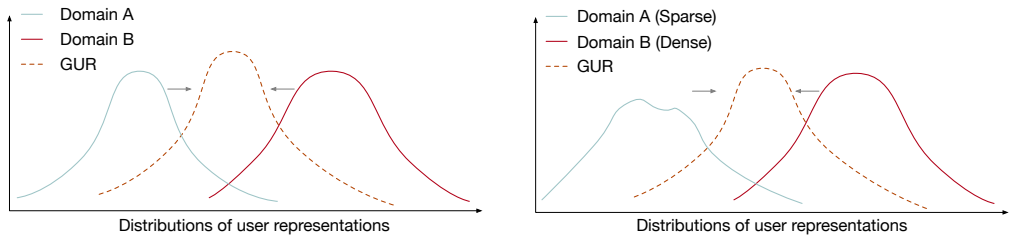
Illustrated in Figure 3.1, different users have different lengths of behaviour sequences, thus, we transform the input sequences of all users into fixed-length sequences with a length of $N + 1$, including the $[eos]$ token. Specifically, for users with sequences longer than N (except for the $[eos]$ token), we consider the most recent N items, for users with sequences shorter than N we add $[pad]$ to the left of its original sequence until it has a total length of N.

Following the convention of autoencoder, by optimizing the reconstruction error between the input samples and the reconstructed samples, our model can learn the most important attributes of the input behaviour sequence. In this chapter, the autoencoder structure is applied to sequence samples, thus we formulate the reconstruction loss for input sequence s^i as follows:

$$\begin{aligned} \mathcal{L}_{rec}(s^i, \hat{s}^i) &= -\log p_\theta(\hat{s}^i|s^i) \\ &= -\sum_{t=1}^{N} \log p_\theta(\hat{v}_t^i|\hat{v}_{<t}^i, \boldsymbol{h}^i), \end{aligned} \tag{3.5}$$

where s^i and $\hat{s}^i$ are the input and reconstruction output of autoencoder, and $\boldsymbol{h}^i = Encoder(Embed(s^i))$ denotes the latent user representation of user u_i as is shown in Equation (3.4). Moreover, at any position t, if its original item is the *[pad]* token, we simply ignore the reconstruction loss at this position.

Generalizing User Representation Across Domains. We propose to incorporate information from other domains into the user representation learned in a single domain to achieve cross-domain collaboration. To achieve this, we unify the distributions of user representations in both domains, as shown in Figure 3.2. Compared



(a) (b)

Figure 3.2. Toy examples demonstrate how to achieve cross-domain knowledge sharing by unifying the distributions of user representations in both domains.

to the single-domain sequential recommendation, the representations regularized by the distributional constraints are more general and contain extra information from other domains. A commonly used technique for autoencoder to integrate prior knowledge is to add extra penalty terms onto the reconstruction objective function:

$$\mathcal{L}_{rec}(s^i, \hat{s}^i) + \Omega(\boldsymbol{h}^i), \tag{3.6}$$

where $\Omega(\cdot)$ denotes the penalty function. For example, sparse autoencoder (SAE) [130] encourages sparsity of latent vectors by adding an l_1 or KL divergence penalty, variational autoencoder (VAE) [131] assumes that the latent representation h^i follows a Multidimensional Gaussian distribution.

Similarly, to integrate knowledge from both domains, a Kullback–Leibler (KL) divergence penalty can be added to the reconstruction loss, where the KL divergence measures the distance between the distributions of the learned user representations in both domains, leading to the following reconstruction objectives in both domains:

$$\begin{aligned} &\mathcal{L}_{rec}(s_A, \hat{s}_A) + \boldsymbol{KL}(\rho^A || \rho^B), \\ &\mathcal{L}_{rec}(s_B, \hat{s}_B) + \boldsymbol{KL}(\rho^B || \rho^A), \end{aligned} \tag{3.7}$$

where ρ^A and ρ^B denote the learned distributions of latent user representations in domain A and domain B, respectively. However, there is no closed-form expression for the distributions of latent user representations. Furthermore, the latent distributions are characterized by the parameters of the autoencoder, which is constantly

updated along the training process.

Instead of directly estimating the ground-truth distributions, which is extremely difficult, we propose to bypass this challenge by adopting an adversarial training strategy to implicitly minimize the KL divergence between the two learned distributions ρ^A and ρ^B. As illustrated in Figure 3.1, the encoders generate latent representations $\boldsymbol{h}_a^i \sim \rho^a$ and $\boldsymbol{h}_b^j \sim \rho^b$, where the distributions are characterized by the parameters of encoders in domain A and B, respectively. A discriminator is then built for a binary classification task, where the input is a single latent representation either from domain A or domain B, while the output is the prediction on which domain the input representation originates from. We adopt the neg log-likelihood loss as the objective function for the adversarial optimization process, denoted as:

$$\begin{aligned} \mathcal{L}_{dis}(h^i) &= -y \cdot log\sigma(f(h^i)) - (1-y) \cdot log(1-\sigma(f(h^i))), \\ y &= \mathbb{I}(u^i \in U^a) \end{aligned} \tag{3.8}$$

where $\mathbb{I}(\cdot)$ is the indicator function which equals to 1 when the condition $u^i \in U^a$ is true, otherwise, $y = 0$. $\sigma(\cdot)$ is the sigmoid function and the logits value $f(h^i)$ is calculated by the domain discriminator $f(\cdot)$.

Following the conventional optimization scheme of GAN, we alternately update the domain discriminator and the encoder until the model reaches the point where the discriminator is unable to tell whether a given user representation is from domain A or B. At this point, the distributions of user representations ρ^A and ρ^B are supposed to be close to each other, i.e., which contributes to reducing the KL divergence $KL(\rho^A||\rho^B)$ between the two distributions. Therefore, we have achieved information sharing through the distributional constraint on latent user representations without relying on the information of overlapped users.

In addition to implicitly unifying the user representations with all users, we also propose an l_2 penalty to enforce the explicit representation sharing over domains for each overlapped user. That is, the representations of the same user should be the same or close across domains. Illustrated in Figure 3.1, for an overlapped user, i.e.

Algorithm 1: RecGURU Training Algorithm

Input: User sets U^A, U^B, item sets V^A, V^B;
behaviour sets: $S_A = \{s_A^i\}, \forall u_i \in U^A$, $S_B = \{s_B^j\}, \forall u_j \in U^B$;
Overlapped users $U^o = U^A \cap U^B$.
Hyper-parameter: CRITIC_ITERS.
Initialization:
Initiate the item and positional embedding matrices: $\mathbf{I^A} \subset \mathbb{R}^{(|\mathbf{V^A}|+\mathbf{1})\times\mathbf{d}}$, $\mathbf{P^A} \subset \mathbb{R}^{(\mathbf{N}+\mathbf{1})\times\mathbf{d}}$ and $\mathbf{I^B} \subset \mathbb{R}^{(|\mathbf{V^B}|+\mathbf{1})\times\mathbf{d}}$, $\mathbf{P^A} \subset \mathbb{R}^{(\mathbf{N}+\mathbf{1})\times\mathbf{d}}$.
Initiate the GURU, CDSRec and the domain Discriminator.
Pre-training Phase:
Train autoencoders according to (3.5) on S_A and S_B.
Multi-task Adversarial Training Phase:
while *not converged* **do**
  for $i \leftarrow 0$ **to** *CRITIC_ITERS* **do**
    Sample two batches of users $\bar{U}^A \sim U^A$, $\bar{U}^B \sim U^B$.
    Optimize the discriminator loss (3.8).
  Sample two batches of users $\bar{U}^A \sim U^A$, $\bar{U}^B \sim U^B$ and one batch of overlapped users $\bar{U}^o \sim U^o$.
  Fix the parameters of the domain Discriminator.
  Train the model according to the loss defined in (3.13).
Fine-tune:
Fine-tune the *CDSRec* model according to the BPR loss defined in (3.12) in each domain.

$u_i = u_j$, we add an l_2 regularizer on its latent representations in domain A and B:

$$\mathcal{L}_{l_2} = ||\boldsymbol{h}_a^i - \boldsymbol{h}_b^j||_2, i = j. \tag{3.9}$$

By adding the l_2 regularizer to the loss function for overlapped users, representations of the same user in different domains are pushed to be equal or close to each other, exploiting the information of overlapped users in an explicit way.

3.2.4 Cross-domain Sequential Recommendation

The generalized user representation extracted by the proposed GURU module represents the overall preference of users in different domains, which is beneficial to the recommendation task in a specific domain.

For the next-item recommendation task in each domain, a sequential recom-

mender is built on the derived GURs and it is composed of multiple unidirectional attention layers and feed-forward layers. Specifically, the GURs are passed into the model through multi-head attention mechanisms.

Formally, for a given user u_i with sequence $s^i = (v_1^i, \cdots, v_{|s^i|}^i)$, the sequential recommender takes its generalized user representation $\boldsymbol{h}^i$, defined in Equation (3.4), and short-term behaviours $(v_{|s^i|-m}^i, \cdots, v_{|s^i|}^i)$ as the input and outputs the current preference vector of the user $\boldsymbol{q}^{i,|s^i|}$ at time step $|s^i|$ in the latent space, given by:

$$\begin{aligned} \boldsymbol{h}^i &= \text{Encoder}(Embed(v_1^i, \cdots, v_{|s^i|}^i, [eos])), \\ \boldsymbol{q}^{i,|s^i|} &= \text{CDSRec}(\boldsymbol{h}^i, (v_{|s^i|-m}^i, \cdots, v_{|s^i|}^i)), \end{aligned} \tag{3.10}$$

where m denotes the length of the short-term behaviour sequence and the GUR is extracted from the long-term behaviour sequence $(v_1^i, \cdots, v_{|s^i|}^i)$ of user u_i. *CDSRec* denotes the cross-domain sequential recommender.

The preference scores of the user to all the candidate items are then computed as the inner product between its current preference $\boldsymbol{q}^{i,|s^i|}$ and the item embeddings of all candidates, denoted as

$$r^{i,|s^i|,v} = \boldsymbol{q}^{i,|s^i|}\boldsymbol{I}_v, \forall v \in V^c, \forall u_i \in U, \tag{3.11}$$

where $\boldsymbol{I}_v$ is the item embedding of v and the candidate set $V^c \subset V$ is a subset of the the entire item set V. Candidate items are then ranked and recommended according to the calculated preference scores.

We adopt a Bayesian Personalized Ranking (BPR) loss [6] to train the recommendation model. For a given user u_i from domain A, we calculate the loss of an item recommendation at time step t as

$$\mathcal{L}_{bpr}^t = -\log \sigma(\boldsymbol{q}_A^{i,t}\boldsymbol{I}_v) - \log(1 - \sigma(\frac{1}{|N_s|}\sum_{v' \in N_s} \boldsymbol{q}_A^{i,t}\boldsymbol{I}_{v'})), \tag{3.12}$$

where v and $\boldsymbol{I}_v$ are the target item and its corresponding item embedding, N_s is the set of negative item samples and $\sigma(\cdot)$ represents the sigmoid function.

3.2.5 Training Strategy

We propose a three-phase training algorithm shown in Algorithm 1 to optimize the proposed model.

In the first phase, we pre-train the autoencoders in each domain individually with the reconstruction task. Through the pre-training process, the reconstruction loss is largely reduced, producing a boost-start for the following adversarial training. In the second phase, following the training process of GAN in [132], at each iteration, we first optimize the discriminator loss $\mathcal{L}_{dis}$ for *CRITIC_ITERS* steps (which equals to 5 in our implementation). Then, the reconstruction task, negative discriminator loss, and l_2 loss on overlapped users are jointly optimized in a multi-task fashion by minimizing the loss:

$$\mathcal{L} = \mathcal{L}_{rec} - \mathcal{L}_{dis} + \mathcal{L}_{l_2}. \tag{3.13}$$

With the reconstruction task, we prevent encoders from generating wild representation stabilizing the adversarial learning. Following the common practice in GAN training [132], we also adopt the gradient penalty term in the critic optimization step. Finally, we fine-tune the *CDSRec* model in each individual domain with the next-item recommendation task.

3.3 Experiments

In this section, we conduct extensive experiments on two public and one collected cross-domain sequential recommendation scenarios. Comparison with the state-of-the-art single-domain and cross-domain baselines shows the effectiveness of our proposed method. Furthermore, ablation tests demonstrate the impact of each proposed sub-modules on the recommendation result and the robustness of our model under scenarios of a different portion of overlapped users.

Table 3.1. Statistics for the three cross-domain scenarios. ("Avg. Seqlen." denotes averaged behaviour length and "#Overlap." is the number of overlapped users).

Domain	#Users	#Items	Avg. Seqlen.	#Overlap.
Sport	9,024	11,835	6.62	1,062
Cloth	46,810	42,139	7.51	
Movie	4,261	5,536	8.307	584
Book	42,940	51,366	9.490	
Wesee	1,952,403	335,648	18.196	1,692,893
Tencent Video	2,183,927	1,455,595	28.53	

3.3.1 Dataset and Experiment Setup

Datasets. Four publicly available Amazon datasets [133] and two collected datasets are used to form three cross-domain sequential recommendation datasets: "Sports-Clothing", "Movie-Book", and the collected dataset. On Amazon datasets, only recent positive reviews, posted after October 1, 2017, are selected. Then, we applied the following pre-processing steps on all four Amazon datasets. First, we selected all the ratings with a score larger than 2, thus, all the interactions in the selected dataset are supposed to be positive feedback. We further ignore the review scores for implicit feedback-based recommendations. Second, we get the 5-core dataset based on the selected interactions. Third, we re-code all the items from 1 to the total number of items in each dataset. Finally, we extracted the user IDs for all the overlapped users in the two cross-domain scenarios, i.e. "Sport-Cloth" and "Movie-Book".

Moreover, we have collected two sequential recommendation datasets from the real-world applications of "Wesee" and "Tencent Video". With over one hundred million users, "Tencent Video" is a Chinese video streaming website offering a wide range of popular movies, TV shows, and short videos. Meanwhile, "Wesee" is a short video-sharing platform boasting over ten million daily active users. Note that, in real-world scenarios, there are only a few overlapping users between these two applications (around 10%). However, for the purposes of this study, we chose

most of the overlapped users due to: firstly, there are still hundreds of thousands of them which is sufficient to train all the evaluated models; and secondly, with such a high overlapping rate we can manually adjust it for stress testing.

Specifically, the "Tencent Video" and "Wesee" datasets were collected within the same time frame to ensure consistency in user preferences during dataset collection, thus allowing for cross-domain collaboration. For each application, interaction information including action type, item IDs, and timestamps was recorded for all users. Positive interactions between users and video items include "clicks" as well as "finishes", i.e., completely watching the video. All other interactions are ignored. The behaviour of a single user over three consecutive days, from June 26, 2020, to June 28, 2020, is stored as three lists: action list, item ID list and timestamp list. Subsequently, entries with bad user IDs or item IDs (such as empty or garbled codes) are removed from the data set. Moreover, if an item appears multiple times in a row for the same user's behaviour sequence then only one copy is kept to prevent duplicated redundant entries from appearing in our results; any additional copies are dropped accordingly. Finally, only users with a behaviour sequence length larger than 5 have been used in this work; details regarding these two datasets can be found within our submission document itself.

The detailed breakdown of the three cross-domain sequential recommendation datasets is shown in Table 3.1. Note that, the collected dataset mostly consists of overlapped users, thus, we can manually adjust the portion of overlapped users in a wide range to test the robustness of our method.

Data split and metrics. Following common practices in sequential recommendation [34], for a given user the second to last item in the behaviour sequence is selected as the validation item, and the last item is used for testing, while the remaining items are used for training. Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [134] are adopted to evaluate the performance of all methods. We follow the strategy used in [12] to reduce the heavy computation cost. Specifically, for users from public datasets, we sample 200 negative items in the item list with respect to their frequencies, which together with the ground-truth

item, form the candidates for recommendation. On collected datasets, the size of the candidate set becomes 20,000. HR and NDCG with $k = 5, 10, 20$ are reported.

Competitors The following single-domain and cross-domain baseline algorithms are evaluated:

- **POP**: All items are recommended according to their popularities.
- **BPRMF** [6]: It optimizes the matrix factorization with the BPR loss.
- **SAS** [16]: It adopts unidirectional self-attention to model user behaviors.
- **Bert4Rec** [34]: It incorporates the idea of Bert [135] to the next item recommendation task.
- **AutoRec**: It is the single-domain version of the proposed model that adopts autoencoder to generate static user representation which is used for the sequential recommendation.
- **CMF** [136]: It simultaneously factors interaction matrices in both target and source domains.
- **MFEM**: It learns a mapping function on overlapped users. Thus, for non-overlapped users, we can get their embeddings in the source domain through the trained mapping function as well as the cross-domain recommendation through Equation (3.2). Here, the user embeddings are learned with the BPRMF model.
- **CnGAN** [51]: It adopts adversarial learning to learn a better mapping function that maps user embeddings from the source domain to the target domain or vice versa.
- **DOML** [49]: It adopts dual metrics learning in the cross-domain recommendation scenario when there are few overlapped users.
- **AutoEM**: As a variation of the proposed method, it also learns a mapping function on overlapped users to get the embeddings of non-overlapped users.

However, here, the item and user embeddings are learned by the proposed AutoRec model.

3.3.2 Implementation Details

We implement the models using PyTorch with python 3.6 and train our framework on Tesla P40 GPUs with a memory size of 22.38 GiB and a 1.53 GHz memory clock rate.

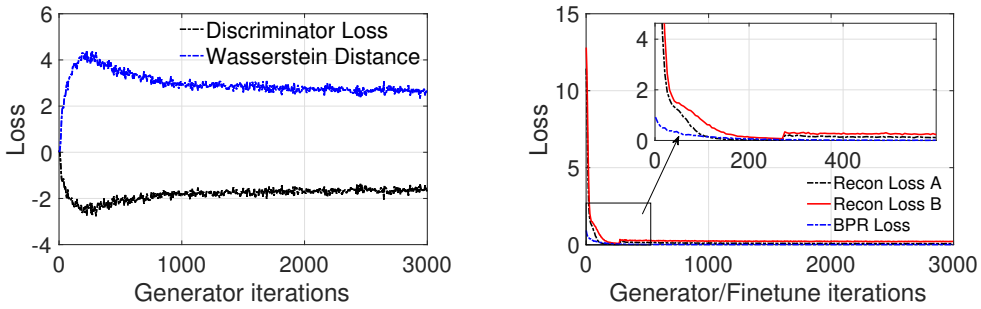
On Amazon datasets, 3 transformer layers are adopted, whereas 6 transformer layers are used on the collected datasets in the encoder and decoder module. We use 2 attention heads for all the attention layers throughout the model. And the dimension of all feed-forward layers is set to 512. For adversarial training, we build the domain discriminator with four fully connected layers with a hidden size of 128. Furthermore, we adopt the improved W-GAN [132] framework to alternately optimize the domain discriminator and the GURU model through the discriminator loss $\mathcal{L}_{dis}$. The implementation of W-GAN is based on a publicly available source on GitHub. [1]. For simplification, we adopt the same length of sequence for both autoencoder and recommendation tasks, which is 100 on all datasets. The dimensionality of user embedding is 64 on both public and collected datasets. Multiple Adam optimizers [137] are used to update different modules of the proposed RecGURU framework.

For all the transfer learning-based cross-domain baselines, we first concatenate the user embeddings from the source and target domains. Then, a fully connected layer is applied to get the cross-domain user embedding which is further used for the next item recommendation.

3.3.3 Experimental Results and Analysis

Training Loss. Figure 3.3 shows the training losses on the "Movie-Book" dataset. Specifically, the Wasserstein distance and the critic loss, i.e. discriminator loss, are

[1] https://github.com/igul222/improved_wgan_training



(a) losses of the adversarial training

(b) Reconstruction and BPR losses

Figure 3.3. Training losses on the "Movie-Book" dataset.

given in Figure 3.3 (a). Both processes converge around a thousand adversarial iterations. In the beginning, the discriminator is good at distinguishing representations from different domains with an increasing Wasserstein distance. After more adversarial training iterations, the Wasserstein distance is reduced and converged which is in line with the training process of standard GANs [132]. The reconstruction tasks in both target and source domains converge after 400 iterations as is shown in Figure 3.3 (b). The recommendation task in the target domain converges after only 100 steps of fine-tuning iterations.

Main Results. Due to the memory issue on the huge amounts of items and users, we aren't able to evaluate baselines such as BPRMF and CMF on the collected datasets, therefore, we leave this dataset for further ablation study. Table 3.2 and 3.3 summarize the performances of all baselines and the proposed methods on the two public cross-domain scenarios. Shown in Table 3.2, AutoRec, the single-domain version of our solution, outperforms SAS and Bert4rec on single-domain sequential recommendation tasks on most of the public datasets. The success of AutoRec can be attributed to the adoption of user representations extracted from the history of user behaviour through an autoencoder.

Shown in Table 3.3, RecGURU outperforms all other baselines with an improvement on HR@10 and NDCG@10 of 6.5%, 9.2%, and 22.5%, 31.3% on the "Sports" and "Movie" datasets, respectively, compared to the best given by all the other baselines. We can also outperform the state-of-the-art single-domain and cross-

Table 3.2. Comparison between the proposed method with single-domain baselines on "Sport-Cloth" and "Movie-Book" scenarios. All values are in percentage, **"AutoRec"** is the single-domain version of our proposed method.

Datasets	Metric	Single-domain algorithms				
		POP	BPRMF	SAS	Bert4Rec	**AutoRec**
Sport	HR@5	3.63	16.17	17.51	18.04	**18.82**
	HR@10	5.70	19.62	22.09	22.46	**23.37**
	HR@20	9.04	24.99	28.97	29.21	**29.84**
	NDCG@5	3.03	12.43	14.47	14.57	**14.93**
	NDCG@10	3.70	13.73	15.94	16.01	**16.36**
	NDCG@20	4.53	15.08	17.67	17.71	**17.97**
Cloth	HR@5	1.48	21.03	23.71	22.68	**23.89**
	HR@10	2.10	23.98	**28.40**	26.57	27.67
	HR@20	5.05	28.27	**35.17**	32.84	33.46
	NDCG@5	0.08	17.89	19.81	19.95	**20.51**
	NDCG@10	0.96	18.86	21.29	21.20	**21.70**
	NDCG@20	1.68	19.89	22.97	22.76	**23.11**
Movie	HR@5	2.07	12.80	12.18	13.78	**14.21**
	HR@10	5.37	17.28	17.20	19.39	**19.51**
	HR@20	11.59	23.97	24.76	27.41	**27.54**
	NDCG@5	1.26	9.36	8.76	9.79	**10.15**
	NDCG@10	2.30	10.85	10.38	11.55	**11.83**
	NDCG@20	3.85	12.50	12.27	13.56	**13.79**
Book	HR@5	2.69	22.37	27.15	27.86	**28.10**
	HR@10	5.31	31.28	36.85	37.58	**38.20**
	HR@20	10.44	42.54	47.81	48.30	**49.38**
	NDCG@5	1.79	16.32	19.50	19.93	**20.09**
	NDCG@10	2.62	19.09	22.60	23.03	**23.33**
	NDCG@20	3.90	21.80	25.32	25.73	**26.27**

domain methods on the "Cloth" and "Book" datasets in most cases but with a relatively smaller margin compared to the results reported on the "Sport" and "Movie" datasets. This is reasonable, as is shown in Table 3.1, the "Sport" and "Movie" datasets are much smaller and with more sparsity than the "Cloth" and "Book" datasets. Obviously, when the data in the source domain is highly sparse, we can only get a limited amount of information from the source domain to help with recommendations in the target domain. Therefore, AutoRec, AutoEM, and RecGURU

Table 3.3. Comparison between the proposed methods and cross-domain baselines on "Sport-Cloth" and "Movie-Book" scenarios. **"AutoEM"** is a variant of our proposed method. All values are in percentage

Datasets	Metric	Cross-domain algorithms					
		CMF	MFEM	CnGAN	DOML	**AutoEM**	**RecGURU**
Sport	HR@5	17.04	15.87	15.88	15.26	17.63	**20.78**
	HR@10	20.55	19.44	19.41	20.29	22.57	**24.88**
	HR@20	29.84	26.00	25.26	25.04	29.66	**30.79**
	NDCG@5	13.37	12.68	12.89	11.66	14.06	**16.56**
	NDCG@10	14.45	13.77	14.05	13.17	15.61	**17.87**
	NDCG@20	17.97	15.84	15.19	15.40	17.36	**19.31**
Cloth	HR@5	21.53	20.56	20.10	22.70	24.27	**25.68**
	HR@10	24.33	23.41	23.29	28.36	28.71	**29.16**
	HR@20	33.46	28.58	28.08	27.97	**35.31**	34.79
	NDCG@5	18.44	17.51	16.72	17.47	20.44	**22.44**
	NDCG@10	19.30	18.40	17.70	19.29	21.83	**23.52**
	NDCG@20	23.11	20.37	19.54	18.86	23.46	**24.92**
Movie	HR@5	12.34	14.70	14.03	14.41	12.93	**18.97**
	HR@10	15.46	19.52	18.49	19.78	18.94	**24.24**
	HR@20	27.54	22.73	26.26	25.30	27.11	**30.81**
	NDCG@5	8.36	10.80	10.18	9.75	8.67	**14.59**
	NDCG@10	8.75	12.31	11.57	11.94	10.61	**16.17**
	NDCG@20	13.79	10.46	13.99	13.26	12.61	**17.77**
Book	HR@5	22.70	23.17	23.24	20.67	**28.92**	28.15
	HR@10	31.56	32.33	32.33	30.86	**38.63**	38.35
	HR@20	49.38	42.81	43.89	43.94	**49.88**	49.68
	NDCG@5	16.38	16.63	16.63	14.18	**20.89**	20.07
	NDCG@10	19.19	19.36	19.36	17.14	**23.99**	23.06
	NDCG@20	26.27	21.98	21.90	22.06	**26.98**	26.81

achieve close performance on the "Book" dataset. However, overall RecGURU outperforms other baselines in most cases. The superiority of RecGURU over AutoRec and AutoEM can be attributed to the generalization of user representations achieved through the adversarial training of the domain discriminator and the autoencoder. Furthermore, compared with the improvements RecGURU has achieved over single-domain sequential recommendation baselines, cross-domain methods such as MFEM, CnGAN, and DOML achieved limited improvements over BPRMF.

Table 3.4. Dataset statistics of the four collected cross-domain scenarios with the portion of overlapped users ranging from 10% to 75%.

Ratio	Dataset	#User	#Item	Avg.Seqlen	#Overlap
10%	Wesee	1203194	288155	16.16	194756
	Tencent Video	1434999	1311440	26.46	
30%	Wesee	1398645	303942	16.18	585580
	Tencent Video	1630372	1356831	26.49	
50%	Wesee	1594256	316338	16.19	976556
	Tencent Video	1825737	1397521	26.51	
75%	Wesee	1838276	329679	16.19	1464731
	Tencent Video	2069892	1438803	26.53	

This can be attributed to the fact that these baselines either rely on overlapped users (MFEM, CnGAN) or need side information for more general user and item embeddings (DOML).

3.3.4 Ablation Study

Variants of the proposed method are evaluated on the collected datasets for ablation studies to show the impact of each proposed sub-modules. We incrementally accommodate different modules into the single-domain sequential recommendation model SeqRec, until we incorporate all the proposed sub-modules and features. Specifically, the following models are evaluated:

- **SeqRec**: Sequential recommendation model without autoencoder.
- **+Auto**: The AutoRec model introduced in Sec. 3.3.1.
- **+GURU**: The proposed RecGURU model.

Furthermore, to test the robustness of the proposed method under a variety of overlapping rates, we manually change the number of overlapped users to form four new datasets with an overlapping rate of 10%, 30%, 50%, and 75%. Table 3.4 provides an overview of the four manufactured datasets. For privacy concerns, we

Table 3.5. Ablation studies on customized collected datasets with the portion of overlapped users ranging from 10% to 75%. Note that "N" denotes "NDCG" and all values are in percentage.

Models	Ratio	Wesee				Tencent Video			
		HR@5	HR@10	N@5	N@10	HR@5	HR@10	N@5	N@10
	30%	15.19	22.47	10.1	12.43	25.95	33.88	18.66	21.16
	50%	15.19	22.39	10.05	12.34	27.14	35.89	19.09	21.91
	75%	15.59	23.08	10.31	12.68	26.71	34.92	18.91	21.55
+Auto	10%	15.41	22.71	10.36	12.71	**28.48**	35.8	**21.49**	**23.87**
	30%	15.49	22.84	10.55	12.91	**29.21**	36.41	**21.7**	**24.01**
	50%	15.52	23.08	10.37	12.8	31.99	**39.76**	**23.91**	**26.4**
	75%	16.12	23.9	10.89	13.39	31.60	**39.76**	**24.33**	**26.78**
+GURU	10%	**16.13**	**23.61**	**10.91**	13.34	28.26	**36.11**	20.78	23.32
	30%	**17.11**	**24.87**	**11.64**	14.14	29.14	**37.12**	21.41	23.98
	50%	**17.15**	**24.87**	**11.66**	14.11	**32.18**	38.49	23.12	25.51
	75%	**16.59**	**24.34**	**11.31**	13.79	**31.23**	39.14	23.39	25.9

did not include users' real ID; thus, for each overlapped user we generated a random number to decide whether to keep it as an overlapped user or put it in either "Wesee" or "Tencent Video". If an overlapped user is put into one domain as a "non-overlapped" user, then its behaviour sequence in the other domain is removed. Therefore, there may be slight differences in the number of items across different manufactured datasets.

Table 3.5 summarizes the results of ablation tests of all the introduced variants on all datasets with various overlapping rates. On the "Wesee" dataset, each time we add a new sub-module or feature incrementally on top of the previous model, we can observe improvement in the overall recommendation performance, which illustrates the effectiveness of autoencoder and GURU modules. Moreover, this phenomenon appears on all four datasets with overlapping ranging from 10% to 75% which shows the robustness of our method to user overlapping rate. Similar to the "Book" dataset, on the "Tencent Video" dataset, the single-domain version of our proposed method, AutoRec, is slightly better than its cross-domain version, which is also due to the sparsity issue in the source domain, i.e. "Wesee", as we have explained before.

3.4 Conclusion

In this chapter, we propose RecGURU, a novel cross-domain sequential recommendation framework based on Generalized User Representations (GURs). Different from previous work which aims to transfer knowledge across domains, in the RecGURU system, we propose the GURU module that is capable of extracting a generalized user representation unified over different domains via adversarial learning. Specifically, an autoencoder is adopted to generate user representations in each individual domain. Cross-domain generalization of user representations is achieved by adversarially training a discriminator and the encoder until the domain-dependent embeddings are statistically indistinguishable among different domains. We further propose various schemes to stabilize and boost the learning effectiveness of RecGURU. Experimental results on both publicly available datasets and collected datasets show the effectiveness of the proposed method.

Chapter 4

Multi-Target Cross-Domain Recommendation

4.1 Introduction

For effective multi-target cross-domain recommendation, we observe that two types of user embeddings can both be helpful in cross-domain recommendations, including 1) the global user embedding, which represents the overall domain-invariant characteristics of a user, and 2) domain-specific user embeddings, which model the user behaviour in various individual domains. Two questions naturally arise concerning the interaction of these embeddings. First, can we generate a global user representation only based on the user's embeddings obtained from individual domains? If this global user embedding is representative enough and not biased toward a single domain, it can be directly used to improve recommendations in all domains without worrying about any negative transfer issue on the target domains. We refer to the first goal as "One for All", as in One global user embedding For recommendations in All domains. To ensure better data isolation, ideally, the global user embedding should be synthesized only based on domain-specific user embeddings without accessing raw behaviour data in each domain.

On the other hand, what is missing in prior work on MTCDR [20], [21] is the transfer of domain-specific user embeddings to assist with recommendations in the

target domain. However, directly transferring these features may cause performance degradation in the target domain due to various reasons, e.g., low-quality embeddings transferred from irrelevant domains. This phenomenon is often referred to as negative transfer [138]. The second question we ask is–can we also transfer domain-specific user embeddings in one domain to help improve prediction in another domain while avoiding negative transfer? We refer to the second goal as "All for One", as in All domain-specific user embeddings for helping the recommendation in One domain.

To address the aforementioned challenges, we propose CAT-ART, a novel multi-target CDR framework, which starts by using traditional matrix factorization to pre-train domain-specific user embeddings in each domain. To achieve the concept of One for All, we propose a Contrastive AuToencoder (CAT) to learn a global user embedding solely based on the pre-trained domain-specific user embeddings from all the domains, without directly accessing raw behaviour data in each domain. To attain the goal of All for One, we build an Attention-based Representation Transfer (ART) unit in each target domain, which transfers and utilizes the pre-trained domain-specific embeddings to boost its recommendation performance while minimizing the impact of negative transfer. Our contributions can be summarized as follows:

We introduce a contrastive autoencoder (CAT), which learns a general global embedding for each user by reconstructing the concatenated sequence of domain-specific user embeddings. To further benefit from self-supervised representation learning and extract robust representations from domain-specific embeddings, we randomly mask some domain in the input sequence to the autoencoder and learn to reconstruct the original sequence, while a contrastive self-supervised loss is used to ensure the masked and unmasked domain-specific user embedding sequences can map to similar global user embeddings for the same user.

We propose Attention-based Representation Transfer (ART), which judiciously adapts the domain-specific user embeddings from other domains to the target domain according to an attention mechanism. ART then combines the target domain

user embedding, the global user embedding, as well as the adapted domain-specific user embeddings to jointly improve recommendations in the target domain.

To evaluate our method in real scenarios, we have collected a large dataset involving over a million users spanning 5 domains, including App installation (App-install), Recent App usage (App-use), article viewing, short-video viewing, and long-video viewing. Each of these domains has its own user behaviour history and independently pre-trained user embeddings. We conduct extensive experiments on the collected data and demonstrate the superiority of the proposed CAT-ART method by comparing it with a range of state-of-the-art MTCDR baselines. Experimental results suggest that CAT-ART significantly outperforms all baselines in most of the domains, e.g., App-install, App-use, article, and long-video, on several evaluation metrics. Moreover, we show that while other state-of-the-art baselines are severely impacted by negative transfer, CAT-ART can effectively avoid the negative transfer issue.

4.2 Methodology

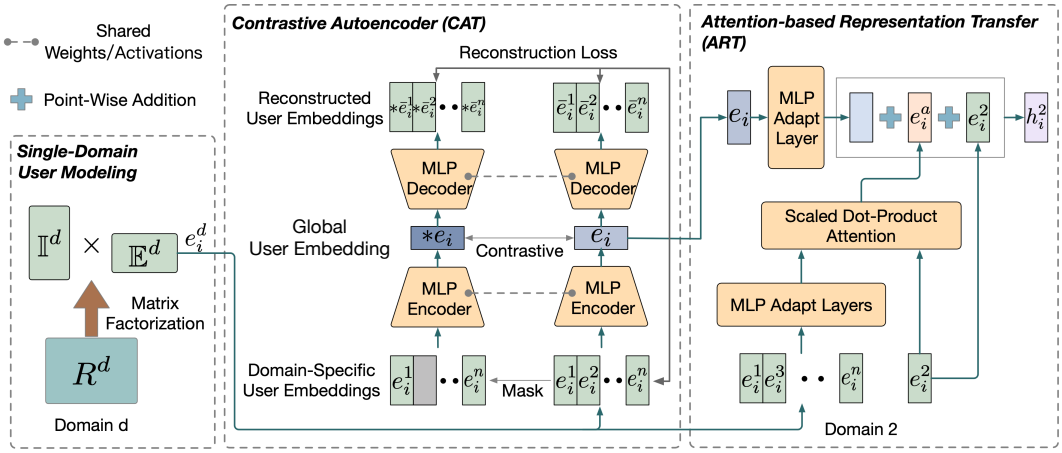


Figure 4.1. The architecture of the CAT-ART model. The CAT module takes domain-specific user embeddings as input and generates global user representation in a self-supervised manner. Then, the global user embedding e_i and the domain-specific embeddings from all the other domains are transferred to a target domain, e.g., domain 2, for boosted recommendations.

4.2.1 Problem Definition

We focus on the MTCDR problem with a global user set U, and item sets $\{V_1, \cdots V_n\}$ in $n \geq 3$ domains. User-item interactions in domain $d \in [1, \cdots, n]$ are represented by a matrix R^d with the shape of $|U| \times |V_d|$ where $|U|$ is the number of users and $|V_d|$ denotes the number of items in domain d. Under the implicit feedback setting, all the elements in the matrix R^d are with the value of either 1 or 0, which indicates whether there is an interaction between a given user-item pair, i.e., for user i and item j in domain d, $r_{ij}^d \in [0, 1], \forall i \in |U|, \forall j \in |V_d|$. We further consider the scenario of data isolation in practical applications, that is, the interactive information between users and items in a specific domain is not observable by other domains. Our goal is to improve the recommendation accuracy in all n domains simultaneously based on the interaction matrices.

4.2.2 Architecture Overview of the Proposed CAT-ART

We set two objectives when dealing with the multi-target CDR problem. That is, 1) One for All: extracting global user representation that is used for recommendations in all domains. And 2) All for One: transferring domain-specific embeddings from all available domains to assist the recommendation in a target domain without negative transfer. To achieve these two objectives and avoid the direct use of raw data across domains, we propose the CAT-ART model where the Contrastive Autoencoder (CAT) module and the Attention-based Representation Transfer (ART) unit are designed for the above two objectives, respectively.

Figure 4.1 provides an overview of the CAT-ART model. First, the domain-specific user embeddings are pre-trained within each domain independently using BPRMF [6]. Then, the CAT module takes the domain-specific embeddings collected from all domains as input and generates global user representation. To create unbiased cross-domain user embeddings, we combine the reconstruction loss and contrastive self-supervised loss in model training, so that the CAT module is capable of extracting informative global user representation. Finally, the ART module

transfers the domain-specific user embeddings from all the other domains to boost the recommendation performance in a single domain, e.g., domain 2 in Figure 4.1. By incorporating attention mechanisms into the ART module, the contribution of each domain can be judiciously adjusted according to their relevance so as to address the negative transfer issue.

4.2.3 Domain-specific User Embedding

Shown in Figure 4.1, in the Single-Domain User Modeling unit, we adopt the widely used Matrix Factorization (MF) model [10] with the Bayes Personalized Ranking (BPR) [6] loss to get the user and item embeddings in each domain. Formally, to factorize the interaction matrix R^d in domain d, we create two trainable embedding matrices $\mathbb{I}^d \subset \mathbb{R}^{(|V_d|+1)\times m}$ and $\mathbb{E}^d \subset \mathbb{R}^{(|U|+1)\times m}$ to represent item and user embeddings in domain d, respectively. Where m represents the number of dimensions in the latent space. For simplicity, we set the embedding dimensions of users and items in all domains to be m. In domain d, given a user u_i with embedding $\boldsymbol{e}_i^d \in \mathbb{E}^d$ and an item $v_j^d \in V^d$ with embedding $\boldsymbol{I}_j^d \in \mathbb{I}^d$, the preference score of the user to the item is computed as the inner product between their embeddings, $r_{ij}^d = \boldsymbol{e}_i^d \boldsymbol{I}_j^d$. Note that, we use bold font to denote vector variables. Then, the BPR loss in domain d is formulated as:

$$\mathcal{L}_{bpr}^d = -\sum_{i \in U} \sum_{j \in p_i^d} \sum_{l \notin p_i^d} \log \sigma(r_{ij}^d - r_{il}^d), \tag{4.1}$$

where p_i^d is the set of items that user i has interacted with in domain d, and $\sigma(\cdot)$ represents the sigmoid function.

By minimizing the BPR loss, we obtain domain-specific user embeddings suitable for recommendation in each domain. Instead of sharing raw user data, we only share the pre-trained domain-specific user embeddings across domains to enable knowledge sharing across domains under the raw data isolation constraint.

4.2.4 Contrastive Autoencoder

The pre-trained domain-specific user embeddings are collected and fed into the CAT module to get global user representations. To do this, we adopt an autoencoder framework which takes the domain-specific embeddings of a user as input and generates latent user representation. Specifically, we first stitch all domain-specific user embeddings, which is a set of real-valued dense vectors, into a large one-dimensional vector in a predefined order, e.g., from domain 1 to domain n. Then, an encoder is used to extract the latent user presentation which is further fed into a decoder to reconstruct the input domain-specific embeddings. We use Multi-Layer Perception (MLP) [139] to build both the encoder and decoder modules. Formally, for a given user u_i with its pre-trained domain-specific embeddings $\{\boldsymbol{e}_i^1, \boldsymbol{e}_i^2, \cdots, \boldsymbol{e}_i^n\}$, we apply the following procedures to get its latent representation:

$$\begin{aligned} \boldsymbol{e}_i &= \text{MLP}_{\text{enc}}(\boldsymbol{e}_i^1 \frown \boldsymbol{e}_i^2 \frown \cdots \frown \boldsymbol{e}_i^n) \\ \bar{\boldsymbol{e}}_i^1 \frown \bar{\boldsymbol{e}}_i^2 \frown \cdots \frown \bar{\boldsymbol{e}}_i^n &= \text{MLP}_{\text{dec}}(\boldsymbol{e}_i), \end{aligned} \tag{4.2}$$

where $\boldsymbol{e}_i$ is the latent representation of user u_i, and $\boldsymbol{e}_i^1 \frown \boldsymbol{e}_i^2$ denotes the long one-dimensional vector after concatenating the vectors $\boldsymbol{e}_i^1$ and $\boldsymbol{e}_i^2$. Here $\bar{\boldsymbol{e}}_i^d$ represents the reconstruction of user embedding $\boldsymbol{e}_i^d$ in domain d. Note that the latent vector $\boldsymbol{e}_i$ has the same size as domain-specific embeddings in individual domains, i.e., $|\boldsymbol{e}_i| = |\boldsymbol{e}_i^d| = m, \forall d \in [1, n]$.

By optimizing the mean square reconstruction error (4.3) between the input and the reconstructed embeddings, the encoder can learn the most important global attributes to reconstruct the domain-specific user embeddings in each domain.

$$\mathcal{L}_{rec} = \frac{1}{|U|} \sum_{i \in U} \sum_{d=1}^{n} ||\boldsymbol{e}_i^d - \bar{\boldsymbol{e}}_i^d||_2. \tag{4.3}$$

On the other hand, the effectiveness of the pre-trained domain-specific user embeddings, which are the input of the autoencoder, is highly affected by the data quality and sparsity in each domain. For example, under-trained user embeddings

from a sparse domain may introduce noise to the input of the autoencoder. Furthermore, the autoencoder may be biased towards domains with a higher quality of user embedding, as it is easier to reconstruct a well-trained embedding than an under-trained embedding with noise.

Therefore, we adopt contrastive self-supervised learning [140] to further train the autoencoder for a more general and robust latent user representation that does not bias to any specific domain. Contrastive self-supervised learning is gaining popularity in the field of representation learning for various visual and natural language processing (NLP) tasks [141]. The core idea of contrastive self-supervised learning is to make the representation of an input sample agree with that of an augmented sample, e.g., obtained by applying Gaussian noise or Cutout [140].

In our problem, contrastive self-supervised learning is integrated into the autoencoder framework to extract the global representations of users from their domain-specific embeddings. As shown in Figure 4.1, we adopt an *Mask* operation to generate the "augmentations" for an input, e.g., $\boldsymbol{e}_i^1 \frown \boldsymbol{e}_i^2 \frown \cdots \frown \boldsymbol{e}_i^n$. Specifically, with *Mask*, we generate "augmented" inputs by removing the domain-specific user embeddings from several randomly selected domains. Then, the "augmentations" are also fed into the encoder module for generating the latent representations. Formally, the following procedure is applied:

$$\begin{aligned} \boldsymbol{e}_i^1 \frown \widehat{\boldsymbol{e}_m} \frown \cdots \frown \boldsymbol{e}_i^n &= \text{Mask}(\boldsymbol{e}_i^1 \frown \boldsymbol{e}_i^2 \frown \cdots \frown \boldsymbol{e}_i^n) \\ *\boldsymbol{e}_i &= \text{MLP}_{\text{enc}}(\boldsymbol{e}_i^1 \frown \widehat{\boldsymbol{e}_m} \frown \cdots \frown \boldsymbol{e}_i^n) \end{aligned} \tag{4.4}$$

where $\boldsymbol{e}_m$ is a trainable vector used to replace the domain-specific embeddings of masked domains, e.g., domain 2 in Figure 4.1. After random masking and padding, the "augmented" sample, i.e., $\boldsymbol{e}_i^1 \frown \widehat{\boldsymbol{e}_m} \frown \cdots \frown \boldsymbol{e}_i^n$, is fed into the encoder MLP_{enc} to get the latent representation $*\boldsymbol{e}_i$. Then, a contrastive loss function is defined for the contrastive prediction task. Given a set of latent embeddings $\{\boldsymbol{e}_1, \cdots \boldsymbol{e}_k, *\boldsymbol{e}_1, \cdots, *\boldsymbol{e}_k\}$ where $(\boldsymbol{e}_i, *\boldsymbol{e}_i)$ forms a positive pair of representations, the contrastive prediction is to identify $*\boldsymbol{e}_i$ in $\{*\boldsymbol{e}_1, \cdots, *\boldsymbol{e}_k\}$ for a given $\boldsymbol{e}_i$, and vice versa, to identify $\boldsymbol{e}_i$ in

$\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_k\}$ for a given $*\boldsymbol{e}_i$.

We randomly sample a minibatch of N users and define the contrastive autoencoder task on the pairs of "augmented" and original user embeddings derived from the minibatch, resulting in $2N$ representations. Let

$$\phi(\boldsymbol{e}_i, *\boldsymbol{e}_i) = \frac{\boldsymbol{e}_i * \boldsymbol{e}_i^T}{|\boldsymbol{e}_i| * \boldsymbol{e}_i|} \tag{4.5}$$

denotes the cosine similarity between two row vectors $\boldsymbol{e}_i$ and $*\boldsymbol{e}_i$. Then, the contrastive loss of user i in the minibatch is computed as:

$$l_i = -log\frac{exp(\phi(\boldsymbol{e}_i, *\boldsymbol{e}_i)/\tau)}{\sum_{k=1}^{N} exp(\phi(\boldsymbol{e}_i, *\boldsymbol{e}_k)/\tau)} - log\frac{exp(\phi(*\boldsymbol{e}_i, \boldsymbol{e}_i)/\tau)}{\sum_{k=1}^{N} exp(\phi(*\boldsymbol{e}_i, \boldsymbol{e}_k)/\tau)}, \tag{4.6}$$

where τ is a temperature parameter. The first term defines the contrastive prediction loss when identifying $*e_i$ given e_i and the second term is the loss for identifying e_i given $*e_i$. By minimizing the contrastive loss, the encoder is trained to put the embeddings of a user and its "augmentations", i.e., e_i and $*e_i$, close to each other in the latent embedding space. By doing this, we can extract a more general latent user representation that is robust to noisy input introduced by low-quality domain-specific user embeddings from a sparse domain.

Apart from the contrastive loss, we further put the $*e_i$ into the decoder MLP_{dec} to reconstruct the original input. That is, instead of minimizing the reconstruction error between the "augmentation", i.e., $\boldsymbol{e}_i^1 \frown \boldsymbol{e}_m \frown \cdots \frown \boldsymbol{e}_i^n$, and its reconstruction, i.e., $*\bar{\boldsymbol{e}}_i^1 \frown * \bar{\boldsymbol{e}}_i^2 \frown \cdots \frown * \bar{\boldsymbol{e}}_i^n$, we optimize the reconstruction loss between the original domain-specific embeddings and the reconstructed embeddings from $\text{MLP}_{\text{dec}}(*\boldsymbol{e}_i)$, that is,

$$\begin{aligned} *\bar{\boldsymbol{e}}_i^1 \frown * \bar{\boldsymbol{e}}_i^2 \frown \cdots \frown * \bar{\boldsymbol{e}}_i^n &= \text{MLP}_{\text{dec}}(*\boldsymbol{e}_i), \\ \mathcal{L}_{rec}^* &= \frac{1}{|U|} \sum_{i \in U} \sum_{d=1}^{n} ||e_i^d - *\bar{\boldsymbol{e}}_i^d||_2. \end{aligned} \tag{4.7}$$

By minimizing $\mathcal{L}_{rec}^*$, the encoder can extract latent representation which is informative enough to reconstruct the masked domains even when the embeddings of masked domains are missing. This further encourages the encoder to extract unbi-

ased global representations of users. Finally, the loss function to train the CAT module can be summarized as:

$$\mathcal{L}_{cat} = \alpha_1 \mathcal{L}_{rec} + \alpha_2 \mathcal{L}^*_{rec} + (1 - \alpha_1 - \alpha_2) \sum_{i=1}^{|U|} l_i, \tag{4.8}$$

where α_1 and α_2 are hyper-parameters to control the weights of each component. The third term represents the sum of the contrastive loss over all users.

The extracted global representation represents the general preferences and overall characteristics of users, that are beneficial to the recommendation task in each domain. In addition, because the global user representation is not biased to any domain, it can be directly transferred to any target domain without worrying about the negative transfer issue.

4.2.5 Attention-based Representation Transfer

The domain-specific embeddings represent the users' preferences in different domains, which are useful features to boost recommendations in a specific domain. However, the direct transfer of domain-specific user embeddings is prone to negative transfers in the target domain. First, if two domains are unrelated, the transferred embeddings may have little or even a negative impact on the target domain. Second, if a domain-specific user embedding is under-trained in its original domain due to data sparsity or insufficient data quality, directly transferring such an embedding may introduce noise to the target domain, leading to performance degradation.

Therefore, we build an attention-based representation transfer (ART) unit in each domain, to integrate the domain-specific embeddings from other domains for better recommendations in the current domain. Specifically, we first build an MLP-based domain adaptation layer for domain-specific embeddings from each domain. Then, we adopt the scaled-dot produce attention [129] which uses the user embedding in the target domain as the query and attends to the domain-specific embed-

dings from other domains. Formally, in domain d, we have

$$\begin{aligned} Q &= \boldsymbol{e}_i^d \\ K = V &= \text{MLP}_{\text{adapt}}(\{\boldsymbol{e}_i^k\}, k \neq d) \\ \boldsymbol{e}_i^a = Attention(Q, K, V) &= \text{softmax}(\frac{QK^T}{\sqrt{m}})V, \end{aligned} \tag{4.9}$$

where m is the dimensionality of the user embedding $\boldsymbol{e}_i^d$ in domain d. With attention, the ART module can assign more weights to the embeddings from the most related domains and reduce the influence of noisy embeddings from sparse or unrelated domains, and thus can effectively alleviate negative transfers in MTCDR.

Final User Embedding. We build an MLP-based domain adaptation module, i.e., $MLP_{ind}(\cdot)$, which adapts the global embedding to the current domain for recommendations. Then, we combine features from the global and domain-specific user embeddings with point-wise addition. For example, in domain d, we have

$$\boldsymbol{h}_i^d = \boldsymbol{e}_i^d + \text{MLP}_{\text{ind}}(\boldsymbol{e}_i) + \boldsymbol{e}_i^a, \tag{4.10}$$

where the first term is the user embedding in the current domain, the second term represents the information transferred from the global user representation, and the last term denotes features from all the other domains. With the user embedding h_i^d, the preference score of the user i to an item j in domain d is recalculated as $r_{ij}^d = h_i^d I_j^d$.

4.2.6 Model Training

The training procedure of the CAT-ART model is given in Algorithm 2. Specifically, the CAT-ART is trained with three stages. First, we apply the BPRMF model in individual domains to get the pre-trained domain-specific user and item embeddings based on BPR loss (4.1). Then, we train the CAT module in a self-supervised manner with both the reconstruction loss and contrastive loss (4.8). Finally, we fix the domain-specific user embeddings and the CAT module and optimize the param-

Algorithm 2: CAT-ART Algorithm

Input: User set U, and item sets: $\{V_1, \cdots V_n\}$, $n \geq 3$;
User-Item interaction matrices: $\{R^1, \cdots, R^n\}$;
Stage 1 (parallel):
for $\forall d \in [1, n]$ **do**
    Train domain-specific user and item embeddings: $\mathbb{E}^d$ and $\mathbb{I}^d$ according to BPR loss.
Stage 2:
Train the CAT module according to reconstruction and contrastive loss.
Stage 3 (parallel):
Fix the CAT module and domain-specific user embeddings in all domains.
for $\forall d \in [1, n]$ **do**
    Train the ART unit according to the BPR loss and the enhanced user embedding.

eters of the $MLP_{ind}(\cdot)$ and ART module according to the BPR loss in individual domains.

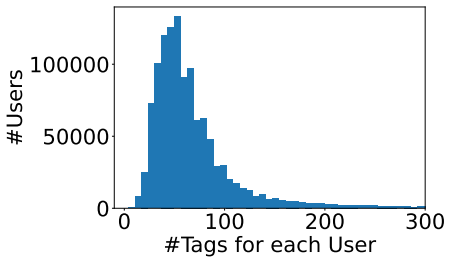
4.3 Experiments

We conduct extensive experiments on a collected dataset and compare our approach with state-of-the-art MTCDR methods.

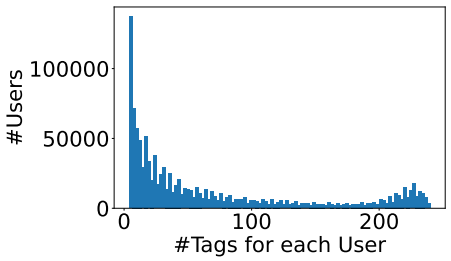
4.3.1 Datasets

We collected user logs from five domains through multiple real apps owned by Tencent. Specifically, they are Application installation preferences ("APP-Ins"), Application usage preferences ("APP-Use"), "Articles", Short Videos ("Video-S"), and Long Videos ("Video-L"). Among them, the "APP-Ins" and "APP-Use" data is collected from January 2021 to August 20, 2021, and the data of "Articles", "Video-S", and "Video-L" are collected from June 2021 to August 2021. To avoid the tremendous item sets, we use the tags that can best represent the visited IDs' features instead of using the item IDs that a user has interacted with. TF-IDF [142] is applied to select a fixed number of the most informative tags in each domain, resulting in 100,000 tags in the "APP-Ins" and "APP-Use" domains and 50,000

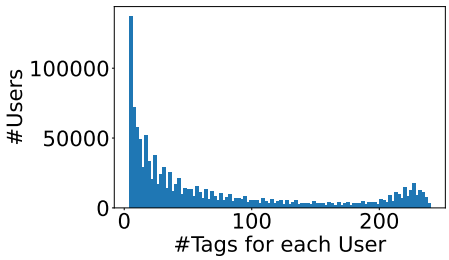
tags in the rest of the domains. We filter out users with fewer than 5 visited tags. Furthermore, the number of tags a user has visited in each domain is truncated to a fixed number, i.e., a user can have a maximum of 100 tags in the "Video-S" domain, and a maximum of 300 tags in the other domains. All the above data processing procedures, e.g., tag selection and cutoff in each domain, are provided by the data owner (Tencent), and have proven to be effective in real-world tasks such as gender and age prediction and user profiling, Table 4.1 provides detailed statistics of the collected dataset with 5 domains.



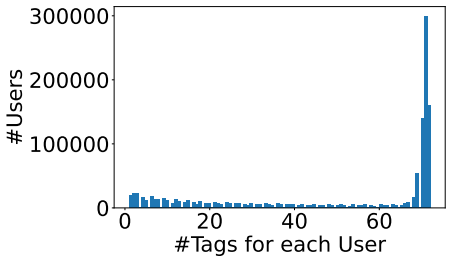
(a) Sparsity Distribution in "APP-Ins"



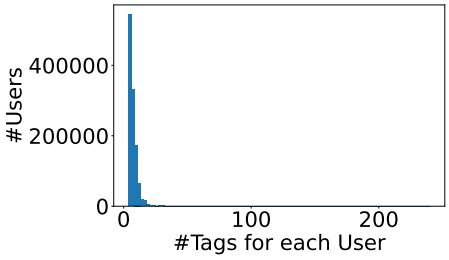
(b) Sparsity Distribution in "APP-Use"



(c) Sparsity Distribution in "Article"



(d) Sparsity Distribution in "Video-S"



(e) Sparsity Distribution in "Video-L"

Figure 4.2. Histograms that show how many tags users have visited in the 5 domains on the train part of our collected dataset.

Table 4.1. Statistics of the Collected Dataset with 5 Domains.

Domain	#Users	#Items	#Interactions	Density(‰)
App-Ins		100,000	101,981,793	0.874
APP-Use		100,000	18,156,535	0.155
Articles	1,166,552	50,000	102,832,656	1.763
Video-S		50,000	74,911,020	1.284
Video-L		50,000	11,412,988	0.196

The histograms, provided in Figure 4.2, show how many tags the user has visited in each domain (on the training part) which shows the data sparsity in a more visual way. We can see the distributions in the "APP-Ins", "APP-Use", and "Article" follow the typical long-tail distribution. The small peak at the end of the distribution is caused by the truncation operation.

4.3.2 Experimental Setup

Our experiments aim at answering the following questions:

- **RQ1.** How does CAT-ART perform vs. state-of-the-art baselines in the MTCDR task?
- **RQ2.** Does our model handle the problem of negative transfer?
- **RQ3.** How do the sub-modules help the model succeed in solving the MTCDR problem?

To train and evaluate a model, we randomly split the interactions of a user in each domain into three parts training (70%), validation (10%), and testing (20%). The evaluation metrics are Precision@K, Recall@K, and NDCG@K [134] that are computed by the all-ranking protocol where all items/tags are ranked. We repeat all experiments three times and give the average and standard deviation of all metrics.
Compared Methods. We compare our model with single-domain recommendation methods(SMF), STCDR methods(CMF), and MTCDR methods(MPF, GA-

MTCDR, HeroGRAPH).

- **SMF**: It factorizes the user-tag interaction matrix of each domain separately based on the BPR loss [6].
- **CMF** [136]: It collects interactions from all domains to form a single matrix which is further factorized for recommendations.
- **HeroGRAPH-L**: HeroGRAPH [20] learns the cross-domain and domain-specific representations. Since HeroGRAPH does not open-source its code, we implement this method based on Lightgcn [23].
- **MPF** [21]: It captures the cross-domain preference with user's behaviour in all domain, and combine it with the user embedding in the target domain for recommendations.
- **GA-MTCDR** [73]: It adopts the node2vec [143] model to pre-train the user/item embeddings in each domain and uses element-wise attention to transfer embeddings among multiple domains.

4.3.3 Model Implementation and Complexity

Environment. We implement our model using PyTorch [144] with python 3.6 and train the framework on Tesla P40 GPUs with a memory size of 22.38 GiB and a 1.53 GHz memory clock rate.

Proposed Method[1]. We use the BPRMF [6] model with the user and item embedding size of $m = 64$ for single domain recommendation and pretraining of domain-specific user embeddings. We reformulate the contrastive loss function given in [140] for the contrastive autoencoder task in our model. For the CAT module, we build the autoencoder with Multi-Layer Perceptron [139] where the sizes of the hidden layers are set to be $[5 \times m, 3 \times m, m]$ and $[m, 3 \times m, 5 \times m], m = 64$ for the encoder and decoder modules, respectively. We adopt the PReLU activation

[1] `https://github.com/Chain123/CAT-ART`

function introduced by [145] between layers. We use a minibatch size of $N = 4096$ in the training of the CAT module and set $\tau = 0.1$ in the contrastive loss (4.6). For each user, we mask out the domain-specific user embedding of one randomly selected domain out of the five domains in our collected dataset. In the loss function $\mathcal{L}_{cat}$, defined in (4.8), we set $\alpha_1 = \alpha_2 = 0.4$. The ART units in each domain have an identical structure, in which all adaptation modules are MLP layers with only one hidden layer which has the same size as its input, and the attention module is with only one head.

Model Complexity. The time complexity of the model training is $\mathcal{O}(\max_{i\in[1,n]} I_i * d + |U|nd^2)$ where n is the number of domains, I_i is the number of interactions in domain i, $|U|$ is the number of users and d denotes the embedding dimension of user.

4.3.4 Experimental Results

Table 4.2 and 4.3 summarize the performances of our proposed CAT-ART model and all the baseline methods on the collected multi-target CDR scenario with 5 domains. The proposed CAT-ART model outperforms all the other baselines in most of the domains. CAT-ART achieves the best performance in "APP-Ins", "'APP-Use', "Article", and "Video-L" domains (**RQ1**). CAT-ART also avoids negative transfer and outperforms most of the baselines in "Video-S". HeroGRAPH-L achieves a slightly better performance on "Video-S". The reasons for this phenomenon are twofold. First, the user behaviours in the "Video-S" domain are richer and more diverse which makes it less likely to be affected by information from other domains, that is, it is hard to improve the recommendation performance in "Video-S" through cross-domain information, as is shown in Table 4.2. This is reasonable. the user behaviours in the "Video-S" domain are richer and more diverse, see Table 4.1, which is less likely to be affected by information from other domains, that is, it is hard to improve the recommendation in "Video-S" through CDR, but its also robust to the negative transfer issue, shown in Table 4.2. Therefore, the recommendation performance in "Video-S" mainly depends on how we

model user and item embeddings from the diverse user behaviour data in the single domain. Secondly, we use the state-of-the-art graph-based recommendation model, i.e., Lightgcn [23], in the HeroGRAPH-L method, which is a deep Graph Convolutional Networks (GCN) that can generate better user and item embeddings in single domain compared to the MF model [146]. In summary, HeroGRAPH-L can get a slightly better performance than our method in the "Video-S" domain due to the superiority of GCN over the MF model. However, HeroGRAPH-L still suffers from the negative transfer problem causing great performance reduction in other domains, such as "APP-Use", and "Video-L". Furthermore, CAT-ART outperforms all the baselines in the other four domains.

Furthermore, the user behaviour in "Video-S" is much more diverse, making the recommendation accuracy of all methods much low than the performance achieved in other domains. Combining these two factors, we argue that, the success of the HeroGRAPh on the "Video-S" domain is mainly attributed to its adoption of the powerful GCN for user and item embedding generation. However, it still suffers from the negative transfer issue in other domains, see Table 4.2.

Compared with the SMF, we can see that CAT-ART effectively handles the negative transfer problem in all domains (**RQ2**). We attribute the success of the CAT-ART framework to the architecture design of the CAT and ART, which are specifically structured to avoid the negative transfer issue while trying to integrate as much useful information as possible from other domains to boost performance. Specifically, the CAT module generates high-quality global user embeddings for recommendations in all domains. And the ART module adoption of the attention mechanism to integrate domain-specific embeddings from other domains. We give a detailed analysis of each module in ablation studies in the following subsection. On the other hand, all the baseline cross-domain methods are severely affected by the negative transfer problem, causing significant performance degradation in many domains, e.g., "APP-Use", "Articles", and "Video-L". This is reasonable. First, as shown in Table 4.1, the "APP-Use" and "Video-L" domains are much sparser than the other domains. Furthermore, user behaviour in "App-Ins", "APP-Use" and

Table 4.2. Results (in %) of the Proposed Method and Baselines. The ↓ represents negative transfer compared with SMF.

Model	Domain	Precision		Recall	
		@10	@20	@10	@20
SMF	APP-Ins	33.82±0.70	25.46±0.88	21.51±0.39	31.91±1.22
	APP-Use	20.91±0.23	12.21±0.26	65.5±0.89	75±1.50
	Article	16.02±0.73	12.05±0.58	16.64±0.43	23.25±0.40
	Video-S	3.9±0.03	3.86±0.02	3.59±0.44	6.9±0.77
	Video-L	5.98±0.20	3.91±0.10	26.73±0.87	34.6±0.88
CMF	APP-Ins	33.57±0.37$^{\downarrow}$	25.19±0.37$^{\downarrow}$	21.8±0.19	32.05±0.43
	APP-Use	20.41±0.11$^{\downarrow}$	12.17±0.05$^{\downarrow}$	64.91±0.27$^{\downarrow}$	75.54±0.16
	Article	10.29±0.27$^{\downarrow}$	8.37±0.19$^{\downarrow}$	8.83±0.23$^{\downarrow}$	13.79±0.28$^{\downarrow}$
	Video-S	3.87±0.12	3.81±0.12$^{\downarrow}$	4.08±0.17	7.6±0.29
	Video-L	4.74±0.03$^{\downarrow}$	3.26±0.01$^{\downarrow}$	21.44±0.12$^{\downarrow}$	29.14±0.06$^{\downarrow}$
MPF	APP-Ins	36.08±1.53	27.11±0.41	23.28±0.99	34.29±0.44
	APP-Use	20.95±0.12	12.26±0.16	65.55±0.44	75.18±0.84
	Article	14.55±0.16$^{\downarrow}$	11.14±0.11$^{\downarrow}$	15.35±0.07$^{\downarrow}$	21.72±0.12$^{\downarrow}$
	Video-S	3.63±0.29$^{\downarrow}$	3.67±0.13$^{\downarrow}$	3.71±0.30	7.16±0.68
	Video-L	2.74±0.95$^{\downarrow}$	2.09±0.52$^{\downarrow}$	11.96±4.31$^{\downarrow}$	18.2±4.66$^{\downarrow}$
GA-MTCDR	APP-Ins	16.77±0.05$^{\downarrow}$	10.35±0.02$^{\downarrow}$	11.7±0.01$^{\downarrow}$	14.37±0.03$^{\downarrow}$
	APP-Use	13.88±0.05$^{\downarrow}$	10.46±0.01$^{\downarrow}$	45.44±0.13$^{\downarrow}$	67.2±0.16$^{\downarrow}$
	Article	4.62±0.13$^{\downarrow}$	3.73±0.03$^{\downarrow}$	4.12±0.14$^{\downarrow}$	6.37±0.11$^{\downarrow}$
	Video-S	3.44±0.03$^{\downarrow}$	3.1±0.02$^{\downarrow}$	3.48±0.08$^{\downarrow}$	6.03±0.06$^{\downarrow}$
	Video-L	3.18±0.15$^{\downarrow}$	2.22±0.07$^{\downarrow}$	14.21±0.74$^{\downarrow}$	19.76±0.54$^{\downarrow}$
HeroGRAPH-L	APP-Ins	34.05±2.01	24.47±1.16$^{\downarrow}$	22.34±1.14	31.61±1.35$^{\downarrow}$
	APP-Use	20.68±0.36$^{\downarrow}$	11.98±0.15$^{\downarrow}$	66.11±0.83	74.96±0.61$^{\downarrow}$
	Article	11.27±0.12$^{\downarrow}$	8.61±0.12$^{\downarrow}$	15.01±0.2$^{\downarrow}$	20.68±0.33$^{\downarrow}$
	Video-S	**3.99**±0.14	3.7±0.15	**5.29**±0.21	**8.97**±0.34
	Video-L	5.42±0.29$^{\downarrow}$	3.65±0.15$^{\downarrow}$	24.62±1.22$^{\downarrow}$	32.84±1.29$^{\downarrow}$
CAT-ART	APP-Ins	**38.36**±0.58	**27.96**±0.31	**24.86**±0.34	**35.46**±0.39
	APP-Use	**21.23**±0.18	**12.33**±0.18	**66.53**±0.65	**75.66**±1.02
	Article	**16.82**±0.21	**12.4**±0.13	**18.76**±0.56	**25.47**±0.6
	Video-S	3.93±0.08	**3.93**±0.06	3.83±0.50	7.35±0.82
	Video-L	**6.08**±0.09	**3.96**±0.08	**27.18**±0.39	**35.01**±0.67

"Article" are monotonous in nature compared with "Video-S" (therefore, the SMF achieves a much higher precision score in these domains). Obviously, when data from all domains are collected and trained together, domains with more sparsity

Table 4.3. NDCG results (in %) of the Proposed Method and Baselines. The ↓ represents negative transfer compared with SMF.

Model	Domain	NDCG	
		@10	@20
SMF	APP-Ins	32.56±0.43	32.53±0.89
	APP-Use	57.39±1.46	60.81±1.72
	Article	21.59±1.30	21.93±1.06
	Video-S	3.83±0.13	4.84±0.25
	Video-L	20.37±1.19	22.91±1.2
CMF	APP-Ins	32.39±0.29$^{\downarrow}$	32.45±0.27$^{\downarrow}$
	APP-Use	43.99±0.78$^{\downarrow}$	47.89±0.74$^{\downarrow}$
	Article	11.24±0.34$^{\downarrow}$	12.07±0.31$^{\downarrow}$
	Video-S	4.00±0.14	5.04±0.18
	Video-L	12.67±0.07$^{\downarrow}$	15.14±0.05$^{\downarrow}$
MPF	APP-Ins	36.95±5.56	36.53±4.02
	APP-Use	55.67±2.71$^{\downarrow}$	59.14±2.52$^{\downarrow}$
	Article	20.96±0.63$^{\downarrow}$	21.29±0.52$^{\downarrow}$
	Video-S	3.85±0.40	4.91±0.11
	Video-L	8.03±3.65$^{\downarrow}$	10.01±3.79$^{\downarrow}$
GA-MTCDR	APP-Ins	17.81±0.08$^{\downarrow}$	16.01±0.03$^{\downarrow}$
	APP-Use	32.35±0.13$^{\downarrow}$	40.16±0.1$^{\downarrow}$
	Article	6.22±0.18$^{\downarrow}$	6.36±0.13$^{\downarrow}$
	Video-S	4.22±0.05	4.69±0.04
	Video-L	10.46±0.63$^{\downarrow}$	12.23±0.49$^{\downarrow}$
HeroGRAPH-L	APP-Ins	40.5±1.91	38.12±1.51
	APP-Use	59.51±1.08	62.74±0.98
	Article	18.19±0.16$^{\downarrow}$	18.86±0.23$^{\downarrow}$
	Video-S	**5.31**±0.18	**6.2**±0.23
	Video-L	18.71±1.21$^{\downarrow}$	21.35±1.24$^{\downarrow}$
CAT-ART	APP-Ins	**43.47**±1.23	**41.55**±0.94
	APP-Use	**59.98**±0.86	**63.27**±1.02
	Article	**25.97**±0.61	**25.79**±0.58
	Video-S	3.93±0.14	5.05±0.24
	Video-L	**21.03**±0.38	**23.54**±0.86

tend to be overwhelmed by other domains causing biased cross-domain representation and negative transfer, e.g., in domains "APP-Use" and "Video-L". In addition, domains with simple and monotonous data are prone to be affected by too much information in other domains, e.g., for domains "App-Ins", "APP-Use", and "Arti-

cles". Apparently, according to the experimental results, none of the previous work can handle these situations(**RQ2**).

4.3.5 Ablation Study and Analysis

We conduct ablation studies to show the effectiveness of each proposed module and to demonstrate how negative transfer is addressed through model design (**RQ3**). We incrementally accommodate different modules into the single-domain matrix factorization model (SMF), until we incorporate all the proposed sub-modules and features. Specifically, the following models are evaluated:

- **SMF**: The single-domain Matrix Factorization (MF) model.
- **+Autoencoder**: We add the original autoencoder to extract global representations for CDR.
- **+Contrastive**: We further add the contrastive loss for the training of the autoencoder, i.e., the CAT module.
- **+ART**: The ART module is further incorporated to integrate domain-specific user embedding.

Furthermore, to show the impact of negative transfer when domain-specific embeddings are directly transferred without attention, we further remove the attention layer within the ART module. Specifically, the following model is evaluated:

- **-Attention**: We remove the attention from the ART and only use MLP layers to integrate domain-specific features.

Table 4.4 and 4.5 summarize the results of ablation studies. The values of metrics Precision@10, Recall@10, and NDCG@10 are given. We can see, each time we add a new sub-module or feature incrementally on top of the previous model, we can observe an improvement in the overall recommendation performance, which illustrates the effectiveness of autoencoder, contrastive self-supervised learning, and

Table 4.4. Results (in %) of ablation studies. The ↓ represents negative transfer compared with the SMF model.

Domain	Metric	SMF	+Autoencoder	+Contrastive
App-Ins	Precision@10	33.82±0.70	37.64±1.17	37.95±0.45
	Recall@10	21.51±0.39	24.35±0.76	24.58±0.35
	NDCG@10	32.56±0.43	41.34±3.75	42.56±2.02
APP-Use	Precision@10	20.91±0.23	21.00±0.11	21.08±0.23
	Recall@10	65.50±0.89	65.77±0.33	66.01±0.88
	NDCG@10	57.39±1.46	59.09±0.37	58.61±0.40
Article	Precision@10	16.02±0.73	16.54±0.46	16.46±0.34
	Recall@10	16.64±0.43	17.48±1.21	17.19±1.13
	NDCG@10	21.59±1.30	23.98±2.28	23.54±2.75
Video-S	Precision@10	3.89±0.025	3.91±0.08	**3.97**±0.13
	Recall@10	3.59±0.44	3.71±0.40	3.72±0.37
	NDCG@10	3.83±0.13	3.87±0.08	3.91±0.05
Video-L	Precision@10	5.98±0.20	6.04±0.01	6.07±0.04
	Recall@10	26.73±0.87	27.00±0.08	27.17±0.20
	NDCG@10	20.37±1.19	21.00±0.14	21.12±0.21

Table 4.5. Results (in %) of ablation studies. The ↓ represents negative transfer compared with the SMF model.

Domain	Metric	+ART	-Attention
App-Ins	Precision@10	**38.36**±0.58	36.24±0.26
	Recall@10	**24.86**±0.34	23.35±0.23
	NDCG@10	**43.47**±1.23	36.08±1.54
APP-Use	Precision@10	**21.23**±0.18	21.01±0.07
	Recall@10	**66.53**±0.65	65.92±0.41
	NDCG@10	**59.98**±0.86	59.28±0.24
Article	Precision@10	**16.82**±0.21	$15.88 \pm 0.15^{\downarrow}$
	Recall@10	**18.76**±0.56	$15.89 \pm 0.38^{\downarrow}$
	NDCG@10	**25.97**±0.61	22.25±1.71
Video-S	Precision@10	3.93±0.08	$3.82 \pm 0.28^{\downarrow}$
	Recall@10	**3.83**±0.50	$3.46 \pm 0.25^{\downarrow}$
	NDCG@10	**3.93**±0.14	$3.73 \pm 0.18^{\downarrow}$
Video-L	Precision@10	**6.08**±0.09	$5.86 \pm 0.03^{\downarrow}$
	Recall@10	**27.18**±0.39	$26.27 \pm 0.09^{\downarrow}$
	NDCG@10	**21.03**±0.38	$20.26 \pm 0.15^{\downarrow}$

the ART modules. Specifically, with the original autoencoder framework, we already get rid of the negative transfer in all domains of our dataset. This can be attributed to the natural advantage of the autoencoder framework where the encoder is trained to extract the most important information to reconstruct the input, thus, is able to reduce the effect of noisy samples. Furthermore, by adding contrastive training (+*Contrastive*), we improve the performance in most of the domains via a more general user representation. However, the recommendation is not always improved, especially in the "Article" domain. The reasons are twofold. 1) In our work, the goal of contrastive learning is to make the user embedding more robust to noise and less dependent on domain-specific information. 2) The "Article" domain is less related to the rest of the domains, that's why all the baseline methods encounter the negative transfer problem in this domain, shown in Table 4.2. Therefore, a more domain-independent global embedding given by the CAT module has less domain-specific information from the "Article" domain, resulting in slightly worse recommendation performance compared to the user embedding given by the autoencoder. Finally, domain-specific embeddings, which represent users' preferences in individual domains, are incorporated through the *ART* module to further boost recommendation performance while avoiding negative transfer through attention. Note that, due to the special data characteristics in the "Video-S" domain, we can only get a relatively small improvement. Thus, all variants have close performance. However, it is clearly shown in Table 4.4 and 4.5 that we get the best performance when incorporating all the modules, i.e., +*ART*, in the rest of the domains.

Furthermore, without attention (-*Attention*), the performance is greatly reduced in all domains and the negative transfer problem also prevails, which demonstrates the effectiveness of the attention module in avoiding the negative transfer problem.

Figure 4.3 shows the averaged attention scores on the test set, given by the ART module in each individual domain. Each row represents the attention weights assigned to the source domains by the corresponding target domain, therefore the sum of weights in a row equals 1. We can see, the weights between related domains

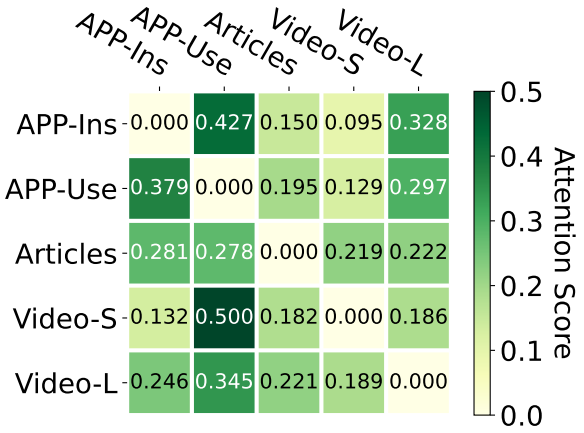


Figure 4.3. Averaged attention scores on the test set. Each row represents the attention scores assigned by the corresponding domain to the other domains.

are high, e.g., domains "APP-Ins" and "APP-Use", while weights between unrelated domains are low, e.g., "APP-Ins" and "Video-L". Furthermore, the asymmetry of the weight matrix shows the inequality in the use of shared domain embedding between two domains. For example, "APP-Use" assigns a weight of 0.13 to the "Video-S" domain, while the "Video-S" gives an attention score of 0.5 to the "APP-Use". This is reasonable, it's helpful for the recommendation of videos if we know what Application a user is more likely to use, but the other way around is much harder. These phenomena further demonstrate the need for a mechanism to select the most important and helpful information among the features provided by multiple domains.

4.4 Conclusion

In this chapter, we focus on the MTCDR problem and propose the CAT-ART model. We build a CAT module to extract robust unbiased global user representation in a self-supervised manner via contrastive learning and an autoencoder framework based on pre-trained domain-specific user embeddings. Then, the ART module is built in each domain, which transfers domain-specific user embeddings from other domains with the attention mechanism. Combining these two modules, CAT-ART

boosts recommendation in all participating domains and avoids negative transfer at the same time. We believe CAT-ART has made a valuable contribution to exploring the MTCDR and the negative transfer issue, approaching the objectives of One for All and All for One.

Chapter 5

Representation Learning on Temporal Heterogeneous Graph

5.1 Introduction

Representation learning on graphs is gaining popularity due to the widespread presence of graph-structured data in real-world scenarios, e.g., citation networks and social networks. Most of these real-world graphs exhibit heterogeneity and complex dynamics that evolve continuously in time. For example, a citation network may have the "Author" and "Paper" nodes, while an author can write multiple papers at different points in time. Although an increasing amount of efforts have been made to study heterogeneous information networks (HINs) and temporal networks, most recent studies focus on either static HINs [103], [147] or temporal monographs [109], [111].

A few recent studies attempt to handle the dynamics and heterogeneity of temporal HINs simultaneously. THINE [122] leverages the attention mechanism and meta-path to handle the heterogeneity and use the Hawkes process to model the evolution of temporal networks. HPGE [123] also leverages the Hawkes process and time-importance sampling techniques to model the dynamics of the temporal graph. However, they produce the all-time general node embeddings, i.e., each node is assigned with a single embedding for all timestamps [148], and thus can not generate

dynamic node embeddings at any given time. In addition, they are all transductive models that are not inductive and scalable to new nodes. Therefore, there still exists a gap in the literature to develop deep inductive models for temporal HINs that can generate time-varying node embeddings in continuous time.

There are two significant challenges to obtaining inductive temporal HIN embeddings. The first challenge lies in better modelling the dynamic impact between heterogeneous nodes. TREND [110] integrates a single Hawkes process to model the temporal influence between nodes. However, it can not handle the node heterogeneity in HINs. THINE [122] and HPGE [123] create a per-node Hawkes process to model the exciting effects between heterogeneous nodes. However, they are not inductive to new nodes and also fail to incorporate the heterogeneity of edges between heterogeneous nodes.

The other challenge is how to effectively capture the evolution of network structures of temporal HINs. Capturing the network structure evolution is crucial to a better understanding of temporal HINs. Most existing works model the evolution of temporal networks as a link formation process where each temporal link/edge is treated as the basic evolution unit in the graph. Thus, they often adopt the temporal link prediction task to train the model, e.g., TGAT [109], CAW [111], and TREND [110]. However, the evolution of real-world temporal HINs often involves the formation of events with complicated high-order subgraph structures. For example, the basic evolution unit in a citation network should be a *publication of a paper*, which is a subgraph event including the authors of the paper, the venue the paper is published in, and the cited papers, where all the edges within the event are formed at the same time. Capturing the formation process of such subgraph events with high-order local structures is critical to understanding the evolution of temporal HINs.

To address the aforementioned challenges, in this chapter, we propose CTRL, a novel Continue-Time Representation Learning model to capture the temporal influence between heterogeneous nodes and the high-order structural evolution of temporal HINs. The CTRL model incorporates temporal and dynamic structural

information of temporal HINs into the Transformer framework [129]. Specifically, temporal information which is modelled by an edge-based Hawkes process and dynamic structural information which is reflected by the dynamic centrality of graphs are incorporated into the aggregation stage of a CTRL layer. We train the model by predicting the event formation in temporal HINs. In proposing CTRL, we make the following contributions:

First, apart from the *semantic correlation* between nodes which are measured by attention scores, we further consider two other key factors in the aggregation process of a CTRL layer. They are 1) *temporal influence* between the target and neighbouring nodes and 2) *dynamic centrality*, which measures the importance of a node over time. To be specific, we propose an edge-based Hawkes process where a neural network is used to extract edge-specific decay rates to capture temporal influence between heterogeneous nodes. We further use the dynamic degrees of neighbours to weigh the importance of each neighbour to incorporate the dynamic centrality in the aggregation procedure. In addition, the messages from heterogeneous temporal neighbours are transformed through node-type- and edge-type-dependent modules before the aggregation to handle graph heterogeneity.

Secondly, we propose to train CTRL through the prediction of temporal events to capture the evolution of high-order structures in temporal HINs. Specifically, two MLP sub-modules are built to predict the event occurrence probability and the probabilities of occurrence of all the edges within the event to preserve the dynamics of both the high-order and first-order local structure. By minimizing the prediction loss events and edges, the CTRL captures the evolution of the high-order local structure of temporal HINs.

We conducted extensive experiments on three public datasets to demonstrate the superiority of CTRL over a range of state-of-the-art approaches and baselines on the inductive temporal link prediction task. Moreover, we demonstrate the effectiveness of the model design choices through ablation studies.

5.2 Preliminaries

In this section, we recap the preliminaries in temporal HINs, graph neural network and Hawkes process.

5.2.1 Temporal HIN

Temporal HIN is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}; \phi, \varphi)$ where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ indicates the set of edges and $\mathcal{T}$ represents the set of timestamps. $\phi : \mathcal{V} \mapsto \mathcal{A}$ and $\varphi : \mathcal{E} \mapsto \mathcal{R}$ are two mapping functions that map nodes and edges to their corresponding types. $\mathcal{A}, \mathcal{R}$ are the sets of types for nodes and edges, respectively. For HINs, $|\mathcal{A}| + |\mathcal{R}| > 2$.

5.2.2 Continuous-time Temporal HIN Embedding

Given a temporal HIN $\mathcal{G}$, the goal of continuous-time representation learning on temporal HIN is to learn an embedding function $f : \mathcal{V} \times \mathcal{T} \mapsto \mathbb{R}^d \times \mathbb{T}$ where d is the dimensionality of node embedding. Thus, we can get the low-dimensional representation of a node at any given time $t \in \mathbb{T}$.

5.2.3 Graph Neural Network

The general architecture of a graph neural network (GNN) layer consists of two parts. They are *Message passing*, which extracts messages from source nodes and passes them to the target node; and *Aggregation*, which aggregates the messages from source nodes to update the representation of the target node.

In this work, we handle the network heterogeneity through node type- and edge-type-dependent modules in the message-passing stage and capture the node feature and temporal topological of a network in the aggregation stage.

5.2.4 Hawkes Process

Hawkes process [121] is a self-exciting point process in which the probability of an event occurring at a particular time depends on the history of events up to that point. Typically, its behaviour is modelled by a conditional intensity function $\lambda(t)$. A common formulation of the intensity function is defined as:

$$\lambda(t) = \mu(t) + \sum_{t_h<t} \kappa(t - t_h; \delta), \tag{5.1}$$

where $\mu(t)$ is the base intensity of the event at time t, $\kappa(t-t_h, \delta) = exp(-\delta(t-t_h))$ denotes the time decay effect of historical event and δ is the decay rate.

In temporal graphs, the Hawks process is used to model the temporal impact between nodes. Specifically, we apply the time decay effect to determine the temporal importance of neighbour nodes in the aggregation process of a GNN layer which is equivalent to the conditional intensity in (5.1), as demonstrated in TREND [110].

5.3 Methodology

In this section, we present the proposed CTRL for temporal HINs. First, we elaborate on the key designs on how we capture the temporal evolution of node features and network structure. Then, we introduce the future event prediction task for the optimization of the proposed GNN model.

5.3.1 Model Overview

Figure 5.1 shows the overall structure of a single layer of the CTRL model. An example is given on how a CTRL layer extracts the temporal representation of node P_1 at time t_3. To be specific, we first sample its historical neighbors (A_1, P_2) which are connected with P_1 through edges t_1, e_1 and t_2, e_3, respectively. Then, the encoded dynamic centrality, indicating the dynamic importance of a node, is added to the node features which are the input of the first CTRL layer, i.e., $h_{P_1}^{t,0}$, $h_{A_1}^{t,0}$, and $h_{P_2}^{t,0}$.

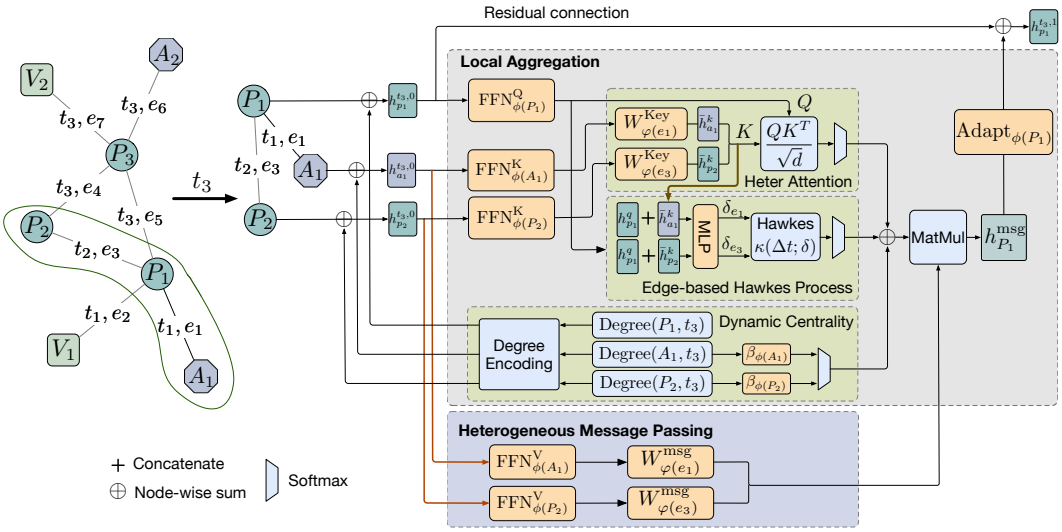


Figure 5.1. The architecture of a layer of the CTRL model.

In message passing, the representations of neighbours are passed to the target node through node type- and edge-type dependent modules, i.e., $\text{FFN}^{V}_{\phi(\cdot)}$ and $W^{\text{msg}}_{\varphi(\cdot)}$. In addition, we consider three key factors to weigh the importance of neighbour nodes during the local aggregation process. They are 1) *semantic correlation* which is calculated via the heterogeneous attention module, 2) *temporal influence* which is measured by the proposed edge-based Hawkes process, 3) and *dynamic centrality* measured by the dynamic degree centrality, which is defined as the number of links incident upon a node before a given timestamp. A node type-dependent adapt module, i.e., $\text{Adapt}_{\phi(P_1)}$, is built to transform the aggregated message, $\boldsymbol{h}_m$, into the feature distribution of the target node. Finally, the adapted message is added to the representation of the target node from the last layer to get its embedding in the current layer, i.e., residual connection.

5.3.2 Heterogeneous Message Passing

To handle the graph heterogeneity caused by different types of nodes and connections/edges, we adopt a heterogeneous message-passing mechanism where node type-dependent modules are used to map feature distributions of different types of nodes into the same latent. Then edge type-dependent modules are used to handle

the edge heterogeneity. Specifically, the messages of passed from neighbors $\{v_i\}_{i=1}^{N}$ to the target node v in the l-th layer is formulated as

$$\boldsymbol{h}_{v,v_i}^{\text{msg}} = \text{FFN}_{\phi(v_i)}^{V}(\boldsymbol{h}_{v_i}^{t_i,l-1})W_{\varphi(e_i)}^{\text{msg}}, i \in [1, N], \tag{5.2}$$

where $\boldsymbol{h}_{v,v_i}^{\text{msg}}$ represents the transformed message passed from v_i to v, N is the number of neighbours, $\text{FFN}_{\phi(\cdot)}^{V}$ and $W_{\varphi(\cdot)}^{\text{msg}}$ are the node type- and edge type-independent transformation modules. $\boldsymbol{h}_{v_i}^{t_i,l-1})$ denotes the representation of v_i from the $l-1$-th layer ($l \geq 1$) at time t_i. Moreover, e_i denotes the edge between node v and v_i, $\phi(v)$ and $\varphi(e)$ denote the types of node v and edge e, respectively. Note that we use bold font to denote vector variables.

5.3.3 Local Aggregation on Temporal HIN

As mentioned in Sec. 5.3.1, we incorporate three key factors in the aggregation process of a CTRL layer which are handled by the following sub-modules.

Heterogeneous Attention The attention scores, calculated as the normalized scaled-dot product values between the representations of the target node and its neighbours, measure the semantic correlation between nodes and are widely used in the aggregation process of GNN, e.g., GAT and SAGE. However, in HINs the target node and its neighbours may have different feature distributions. Moreover, there may exist different types of edges between a pair of node types. Therefore, edge information between two nodes should be considered when calculating their semantic correlation.

Specifically, for an edge, we first map the representations of heterogeneous nodes into the same latent space via node type-dependent mapping functions to handle the node heterogeneity. Then, the mapped representations of the source nodes are further transformed through an edge type-dependent matrix to further incorporate the edge heterogeneity. Finally, the attention score is computed by the scale-dot product between the transformed representations of the source and target nodes. Therefore, given a target node v at timestamp t with its neighbor set

$\{(v_i)\}_{i=1}^{N}$, the attention score is calculated by

$$
\begin{aligned}
Q &= \text{FFN}^{Q}_{\phi(v)}(\boldsymbol{h}_v^{t,l-1}), \\
K &= \{\text{FFN}^{K}_{\phi(v_i)}(\boldsymbol{h}_{v_i}^{t_i,l-1})W^{Key}_{\varphi(e_i)}\}_{i=1}^{N}, \\
\text{Attn}^{t,v} &= \{\text{attn}_{v_i}^{t,v}\}_{i=1}^{N} = \text{softmax}(\frac{QK^T}{\sqrt{d}}),
\end{aligned}
\tag{5.3}
$$

where d is the dimension of the hidden representations, $\text{FFN}^{Q}_{\phi(\cdot)}$ and $\text{FFN}^{K}_{\phi(\cdot)}$ are the node type-dependent feedforward network for the *Query* and *Key* terms, respectively. $W^{Key}_{\varphi(\cdot)}$ is the edge type-dependent matrix which further transfers the representations of neighbour nodes based on their connection types between the target node.

Edge-based Hawkes Process. Intuitively, the impact of historical nodes decays over time. Previous methods often leverage the Hawkes process to model this temporal impact between nodes. However, in a temporal HIN, different types of historical nodes may have different dynamics, i.e. different decay rates. Furthermore, the temporal decay of impact may vary even for the same type of node. For example, in a social network, the impacts of different celebrities may decay differently for different followers depending on their connection(edge) types, e.g., mutual follow or unilateral follow.

To this end, we propose an edge-based Hawkes process to measure the influence of neighbour nodes according to their connection information with the target node. To be specific, all the available information of an edge, including types and representations of the target and source nodes and also the type and feature (if available) of the edge, are taken into account to generate the edge-specific decay rate for the corresponding edge-based Hawkes process. Formally, the influence intensities for a set of historical neighbour nodes $\{v_i\}_{i=1}^{N}$ of a target node v at time t are calculated

by

$$
\begin{aligned}
\boldsymbol{h}_v^q &= \text{FFN}_{\phi(v)}^Q(\boldsymbol{h}_v^{t,l-1}), \\
\bar{\boldsymbol{h}}_{v_i}^k &= \text{FFN}_{\phi(v_i)}^K(\boldsymbol{h}_{v_i}^{t_i,l-1}) W_{\varphi(e_i)}^{Key}, \\
\delta_{e_i} &= \sigma(\text{MLP}(\boldsymbol{h}_v^q \frown \bar{\boldsymbol{h}}_{v_i}^k)), \\
\lambda^{v,t} &= \{\lambda_{v_i}^{v,t}\}_{i=1}^N = \text{softmax}(\{\kappa(t - t_i, \delta_{e_i})\}_{i=1}^N),
\end{aligned}
\tag{5.4}
$$

where e_i represents the edge between the target node v and neighbor v_i, $t_i (< t)$ denotes the time when the edge e_i between node v and v_i is established, and $\frown$ denotes the concatenation operation between two vectors. Moreover, introduced in (5.3), $\text{FFN}_{\phi(\cdot)}^Q$ and $\text{FFN}_{\phi(\cdot)}^K$ are node type-dependent modules to handle the node heterogeneity and $W_{\varphi(\cdot)}^{Key}$ is edge type-dependent matrix to deal with the heterogeneous connections. Furthermore, the MLP module takes the transformed latent representations of the target and source nodes, i.e., $\boldsymbol{h}_v^q$ and $\bar{\boldsymbol{h}}_{v_i}^k$, as input and outputs the decay rate, δ_{e_i}, of the Hawkes process that models the dynamic impact of neighbour v_i to v. Introduced in (5.1), $\kappa(\Delta t, \delta) = exp(-\delta \Delta t)$ calculates the dynamic influence intensity. Note that, $\sigma(\cdot)$ is the ReLU [149] activation function. Finally, the relative temporal influence of all historical neighbour nodes is captured with the softmax function.

Dynamic Centrality As shown in Figure 5.1, we first get the dynamic degree centrality of a node at a given timestamp, e.g., the dynamic degree of node P_1 at t_3 is denoted as Degree(P_1, t_3). The dynamic degree measures the node importance at the current time which can is incorporated into the aggregation procedure of the CTRL layer. However, the degrees of different types of nodes are often not comparable. For example, in a citation network, the dynamic degree of a venue (or conference) is much larger than the degree of an author. Therefore, we adopt trainable node-type dependent variables, i.e., $\beta_{\phi(\cdot)}$, to scale the degrees of different types of nodes before feeding into the softmax function. To be specific, given a target node v and its neighbors $\{v_i\}_{i=1}^N$ at time t, we have

$$
\omega^{v,t} = \{\omega_{v_i}^{v,t}\}_{i=1}^N = \text{softmax}(\{\beta_{\phi(v_i)} * D_{v_i}^t\}_{i=1}^N), \tag{5.5}
$$

where $D_{v_i}^t$ denotes the dynamic degree centrality of node v_i at t.

Finally, the hidden representation of node v at time t in the l-the CTRL layer is formulated as

$$\begin{aligned} \boldsymbol{h}_v^{\text{msg}} &= \sum_{i=1}^{N} (\alpha_1 \text{attn}_{v_i}^{v,t} + \alpha_2 \lambda_{v_i}^{v,t} + \alpha_3 \omega_{v_i}^{v,t}) \boldsymbol{h}_{v,v_i}^{\text{msg}}, \\ \boldsymbol{h}_v^{t,l} &= \boldsymbol{h}_v^{t,l-1} + \text{Adapt}_{\phi(v)}(\boldsymbol{h}_v^{\text{msg}}), \end{aligned} \tag{5.6}$$

where $\boldsymbol{h}_{v,v_i}^{\text{msg}}$, defined in (5.2), is the transformed message passed from v_i to v, $\boldsymbol{h}_v^{\text{msg}}$ is the aggregated message received by node v from all its neighbor nodes. $\alpha_1 + \alpha_2 + \alpha_3 = 1$ are trainable weights for the semantic importance, temporal influence, and the dynamic degree centrality of each neighbour. $\text{Adapt}_{\phi(v)}$ is the node type-dependent adapt module, which consists of a single linear layer, that maps the received message vector, $\boldsymbol{h}_v^{\text{msg}}$, to the latent feature space of the target node v. Note that we only consider undirected graphs here, however, the degree of centrality can be easily extended to directed graphs.

Furthermore, the dynamic degree is also an important dynamic feature of nodes. Thus, we develop a degree encoding module which assigns real-valued vectors to the dynamic degree of each node. The degree embedding is further added to the node feature as the input of the first CTRL layer. That is,

$$\boldsymbol{h}_{v_i}^{t,0} = \boldsymbol{x}_{v_i} + \boldsymbol{z}_{v_i,t}, \tag{5.7}$$

where $\boldsymbol{x}_{v_i}$ is the node feature of node v_i, $\boldsymbol{z}_{v_i,t}$ is the embedding of the dynamic node degree of node v_i at time t. For a directed graph, $\boldsymbol{z}_{v_i,t}$ can be divided into $\boldsymbol{z}_{v_i,t}^{+}$ and $\boldsymbol{z}_{v_i,t}^{-}$ which stand for the embeddings of the dynamic indegree and outdegree.

5.3.4 Event-based Training

Previous temporal network methods are often trained by modelling the temporal link formation process which ignores the evolution of high-order network structures. Therefore, we propose to train the CTRL by modelling the evolution of the

events in a given temporal HIN.

Different networks have different basic events, for example, the event of a citation network should be the *publication of a paper* which is a subgraph including the authors, venue, and the cited papers. Therefore, we first identify the basic event for a given temporal HIN and model the evolution of the predefined basic even from two perspectives.

First, we predict the probability of an event, as a whole, occurring. To be specific, for an event, which is subgraph, $\mathcal{G}_s^t = \{\mathcal{V}_s^t, \mathcal{E}_s^t\}$ at time t. We get the event embedding by taking averages of the node representations in the event, that is,

$$\boldsymbol{h}_{g_s^t} = \frac{1}{|\mathcal{V}_s^t|} \sum_{v \in \mathcal{V}_s^t} \boldsymbol{h}_v^{t,L}, \tag{5.8}$$

where $\boldsymbol{h}_v^{t,L}$ is the latent node representation from the last CTRL layer. Then, we build an MLP module, $\text{MLP}_{\text{event}}$, which takes the event representation as input and predicts the probability of an event occurring.

$$p_{g_s^t} = \text{MLP}_{\text{event}}(\boldsymbol{h}_{g_s^t}). \tag{5.9}$$

Secondly, to preserve the topological information within each event, we further predict the occurrence probability of each edge in an event. Specifically, we further build another MLP module, MLP_{edge}, that takes the representations of the source and target node of an edge as input and outputs the probability of the edge occurring. Formally, for an given $e_j^t = \{v_i, v_j, t\} \in \mathcal{E}_s^t$, the edge probability is calculated as:

$$p_{e_k^t} = \text{MLP}_{\text{edge}}(\boldsymbol{h}_{v_i}^{t,L} || \boldsymbol{h}_{v_j}^{t,L}). \tag{5.10}$$

To explore the evolution of the whole network, for each event $\mathcal{G}_s^t$, we apply negative sampling on the temporal HIN to create a negative event $\mathcal{G}_n^t = \{\mathcal{V}_n^t, \mathcal{E}_s^t\}$ by randomly sample a set of negative nodes $\mathcal{V}_n^t$. Note that, there are still overlapping between the node sets of the $\mathcal{G}_s^t$ and $\mathcal{G}_n^t$, i.e., $\mathcal{V}_s^t \cap \mathcal{V}_n^t \neq \emptyset$. Then, the event occurrence

loss of $\mathcal{G}_s^t$ is calculated as:

$$\mathcal{L}_{\text{occur}}^{g_s^t} = -log(p_{g_s^t}) - log(1 - p_{g_n^t}), \tag{5.11}$$

where $p_{g_n^t}$ is the occurrence probability of the negative event $\mathcal{G}_n^t$ defined in (5.9). Furthermore, for each edge $e_k^t = \{v_i, v_j, t\} \in \mathcal{E}_s^t$, we create a negative edge $\hat{e}_k^t = \{v_i, v_j', t\}$ by randomly sample a negative target node v_j'. Then the structural loss of event $\mathcal{G}_s^t$ is computed by:

$$\mathcal{L}_{\text{topo}}^{g_s^t} = -\frac{1}{|\mathcal{E}_s^t|} \sum_{e_k^t \in \mathcal{E}_s^t} (log(p_{e_k^t}) + log(1 - p_{\hat{e}_k^t})), \tag{5.12}$$

where $p_{\hat{e}_k^t}$ is the probability of negative edge $\hat{e}_k^t$ defined in (5.10) and $\mathcal{E}_s^t$ is the edge set of event $\mathcal{G}_s^t$.

Finally, we optimize model parameters θ through the following objective function

$$\arg\min_{\theta} \sum_{\mathcal{G}_s^t \in \mathcal{G}} \mathcal{L}_{\text{occur}}^{g_s^t} + \mathcal{L}_{\text{topo}}^{g_s^t}, \tag{5.13}$$

where $\mathcal{G}$ is a temporal HIN. We optimize the objective with gradient descent on a batch of predefined events.

5.4 Experiments

We conduct extensive experiments and compare our approach with state-of-the-art methods to demonstrate the effectiveness of the proposed model.

5.4.1 Experimental Setup

Datasets. We evaluate the proposed approach and all the baseline methods on three real-world datasets, i.e., ACM[1], DBLP, and IMDB[2]. The ACM and DBLP are two paper citation networks where the ACM dataset contains three types of nodes ("pa-

[1]https://www.aminer.cn/citation#b541
[2]https://www.imdb.com/interfaces/

Datasets	#N/E-Types	#Nodes	#Edges	Time Range
ACM	3; 3	87,926	204,436	2000-2016
DBLP	4; 4	147,138	612,673	2010-2020
IMDB	2; 6	96,175	217,358	2000-2020

Table 5.1. Data statistics. #N/E-Types denotes the number of node and edge types.

per", "author", and "venue") and the DBLP dataset involves four types of nodes ("paper", "author", "venue", and "field"). The IMDB dataset is a network with two types of nodes ("movie" and "people") and six types of edges ("editor", "actress", "actor", "director", "writer", and "producer"). [3]The statistics of these datasets are shown in Table 5.1.

For citation networks, i.e., ACM and DBLP datasets, we use Doc2vec [150] model to convert the title and abstract of each paper into a real-valued vector with a dimension size of 128, which is treated as the raw features of the paper node. Moreover, in the DBLP dataset, we leverage the word2vec method to convert each "field" into a vector of length 128. Then, we use all-zero vectors as the raw features of other node types, i.e., "author" and "venue" nodes. For the IMDB dataset, we encode the raw features of a movie, including its title, genre, and release region, into a vector of size 246 and also use all-zero vectors as the raw features of "people" nodes.

In addition, the basic event for a citation network is defined as *the publication of a paper* which includes the authors of the paper, the venue where the paper is published, the cited papers, and the field of study (available for the DBLP dataset). For the IMDB dataset, the basic event is defined as *the release of a movie* consisting of the movie node and its corresponding crews including director, producer, actor, editor, etc.

Experimental setup. We split a temporal HIN into three parts for training, validation, and testing, according to the timestamps to make sure that all methods are trained and evaluated under the same conditions. Specifically, we split up a

[3]Dataset details can be found in supplementary

Models	Heterogeneity	Temporal	Centrality
GCN [99]	✘	Sample	✘
SAGE [22]	✘	Sample	✘
GAT [100]	✘	Sample	✘
RGCN [151]	✘	Sample	✘
TGAT [109]	✘	Encode	✘
HGT [105]	✔	Encode	✘
TGSRec [24]	✔	Encode	✘
CAW [111]	✘	Encode	✔
TREND [110]	✘	Hawkes	✔

Table 5.2. Details of baseline methods. The "Temporal" column indicates how each baseline utilizes time information where "Sample" for temporal sampling, "Embed" for time encoding, and "Hawkes" for Hawkes process.

temporal network according to the occurrence time of the predefined basic event. Therefore, all the events in the validation and testing sets are not involved in the training process and all the edges in the validation and testing sets are new edges where at least one node of an edge is not involved in model training. So that we can evaluate all methods with the inductive link prediction task. We evaluate all methods with four metrics including Accuracy, Averaged Precision (AP), F1 score, and AUC. We repeat all experiments at least three times and give out the average and standard deviation of all metrics.

Baselines. We compare the proposed method with a range of state-of-the-art network embedding methods. Specifically, we have compared with the baseline methods in Table 5.2. For static methods, i.e., GAT, SAGE, GCN, and RGCN, we use temporal neighbourhood sampling to adapt to the temporal graph embedding setting.

5.4.2 Implementation Details

We evaluate the proposed method and baselines on a server with AMD EPYC 7K62 48-Core Processor and NVIDIA Tesla P40 GPUs. The experimental environment is Linux 3.10 with CUDA 10.1. To have fair comparisons, the dimension of node

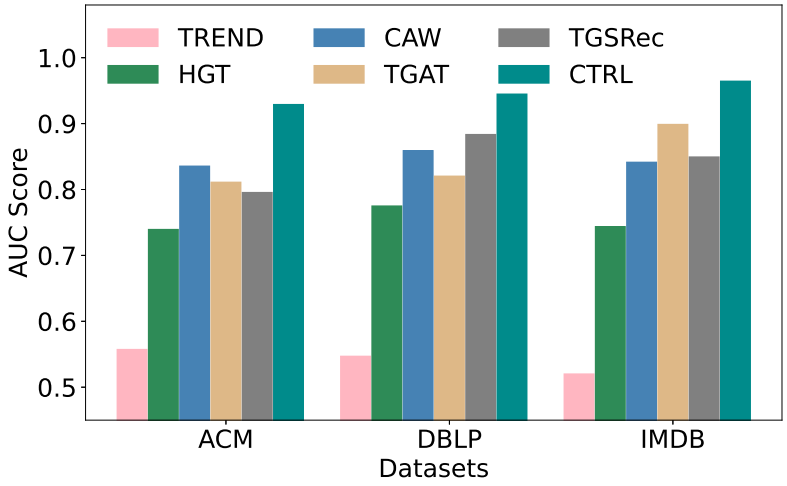


Figure 5.2. AUC of the proposed method and temporal network embedding baselines on the three datasets.

embedding is set to 128 for all methods. We use the Adam optimizer with a learning rate of 0.001 and adopt a batch size of 1024. Moreover, we set the number of neighbours $N = 10$ and randomly select one negative sample for a positive edge in both training and evaluation. We apply two GNN layers for both the CTRL model and all baseline methods. The number of heads of the scaled-dot product module is set to 2. In the training stage of the CTRL, we sampled (with repeat) 5 authors, 10 cited papers, and 10 fields of study for each paper to form the "publication of a paper" event in the ACM (without the "field" node) and DBLP datasets. Similarly, 10 "people" nodes with different roles, e.g, director, actor, etc., are sampled for each movie to form the "movie release" event in the IMDB dataset. Furthermore, we use the default settings for other parameters of the baselines.

5.4.3 Main Results

Table 5.3, 5.4, and 5.5 summarize the experimental results of all baselines and the proposed method on the three real-world temporal HINs. The proposed model significantly outperforms all other baselines with a relative performance improvement on accuracy, average precision (AP), and F1 scores of 12.65%, 8.53%, and 9.53% on the ACM dataset. Similarly, on the DBLP and IMDB datasets, we achieved a

Methods	Accuracy(%)	AP(%)	F1(%)
GCN	58.79±0.3	67.29±1.02	65.72±0.16
SAGE	60.7±1.21	63.9±1.18	66.21±0.95
GAT	61.35±0.74	63.96±1.42	63.3±1.08
RGCN	63.37±0.35	65.35±0.42	63.94±0.62
TGAT	72.77±1.47	79.95±1.13	71.43±1.5
HGT	68.01±0.36	71.78±1.24	68.46±0.43
TGSRec	75.02±1.29	73.93±3.95	78.08±1.07
CAW	73.87±0.09	84.64±0.14	76.64±0.09
TREND	55.72±1.53	53.15±0.94	59.54±3.16
CTRL	**84.51±0.12**	**91.86±0.57**	**85.52±0.06**
Imp.(%)	12.65%	8.53%	9.53%

Table 5.3. Results of the inductive temporal link prediction task on the ACM dataset. Imp.% indicates the relative performance improvement of our methods compared to the best results given by all the baselines. All improvements are significant with a t-test p-value less than 0.05.

Methods	Accuracy(%)	AP(%)	F1(%)
GCN	66.59±0.18	72.62±0.24	67.81±0.16
SAGE	67.47±0.16	73.57±0.14	73.54±0.13
GAT	68.84±0.47	75.91±0.65	67.69±0.65
RGCN	68.8±0.32	73.98±0.43	68.31±0.48
TGAT	75.09±2.11	82.81±2.35	72.93±2.95
HGT	69.65±0.42	77.49±0.19	67.84±0.93
TGSRec	79.76±0.28	86.5±0.08	80.33±0.63
CAW	74.95±0.16	86.68±0.11	70.21±0.56
TREND	54.69±2.44	52.49±8.84	64.11±9.1
CTRL	**86.46±0.08**	**94.15±0.03**	**86.22±0.08**
Imp.(%)	8.4%	8.62%	7.33%

Table 5.4. Results of the inductive temporal link prediction task on the DBLP dataset. All improvements are significant with a t-test p-value less than 0.05.

Methods	Accuracy(%)	AP(%)	F1(%)
GCN	59.26±0.22	62.38±0.41	59.62±0.73
SAGE	61.35±0.56	67.59±1	61.86±1.86
GAT	61.28±0.12	65.6±0.4	62.05±1.48
RGCN	67.33±0.46	71.64±0.7	67.12±1.21
TGAT	82.33±0.3	88.63±0.45	83.14±0.25
HGT	67.35±0.23	72.77±0.23	68.48±0.27
TGSRec	75±0.73	83.47±1.67	77.66±0.91
CAW	74.66±0.03	85.24±0.02	73.45±0.17
TREND	52±0.24	51.04±2.23	55.09±2.43
CTRL	**90.91±0.25**	**95.69±0.39**	**91.19±0.12**
Imp.(%)	10.42%	7.97%	9.68%

Table 5.5. Results of the inductive temporal link prediction task on the IMDB dataset. All improvements are significant with a t-test p-value less than 0.05.

relative improvement of 8.4%, 8.62%, 7.33% and 10.42%, 7.97%, 9.68%. In addition, Figure 5.2 shows the AUC scores of our model and other temporal graph embedding baselines. We observe that the proposed model achieved AUC scores over 0.9 in all datasets, which significantly outperforms the baselines.

Generally, the temporal methods are able to outperform all the static methods except for the TREND model. In the training stage, TREND tries to predict the dynamic node degree which is problematic in some networks, e.g., in citation networks, the dynamic degree of the "venue" node is on the order of thousands, while the dynamic degree of "author" node is less than 10 for most authors. Moreover, in the movie network, the dynamic degree of a movie is also 0. These may cause the network to focus more on the node degree and fail to preserve the network structure. In our work, the dynamic degree is encoded into node features and is scaled with trainable type-dependent variables. Therefore, the proposed model can capture dynamic centrality without such a problem.

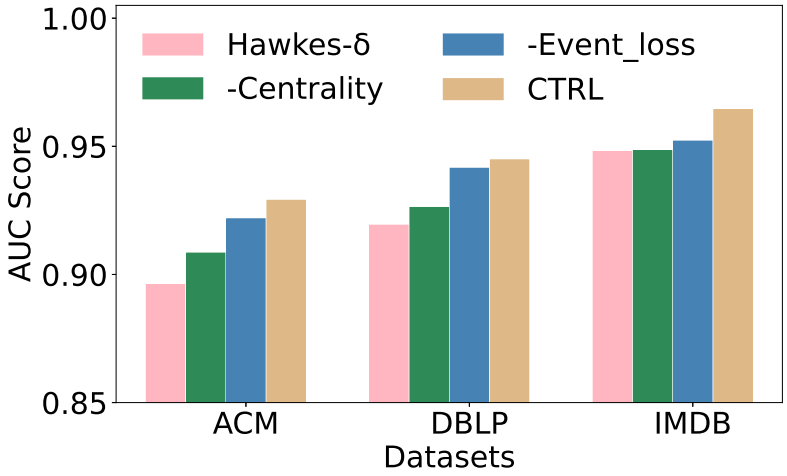


Figure 5.3. AUC scores of ablation studies.

ACM Dataset			
Variants	Accuracy(%)	AP(%)	F1(%)
CTRL	**84.51±0.12**	**91.86±0.57**	**85.52±0.06**
-Event_loss	83.56±0.88	90.91±0.76	84.79±0.7
-Centrality	82.47±0.29	89.64±0.25	83.95±0.26
Hawkes-δ	81.01±0.37	88.97±0.16	81.56±0.37
DBLP Dataset			
Variants	Accuracy(%)	AP(%)	F1(%)
CTRL	**86.46±0.08**	**94.15±0.03**	**86.22±0.08**
-Event_loss	86.07±0.31	93.74±0.24	85.87±0.32
-Centrality	84.63±0.13	91.81±0.16	84.79±0.32
Hawkes-δ	83.49±0.27	91.41±0.21	83.37±0.31
IMDB Dataset			
Variants	Accuracy(%)	AP(%)	F1(%)
CTRL	**90.91±0.25**	**95.69±0.39**	**91.19±0.12**
-Event_loss	89.3±0.14	93.91±0.05	89.81±0.08
-Centrality	88.4±0.02	94.05±0.26	88.54±0.04
Hawkes-δ	88.19±0.18	94.03±0.1	88.37±0.14

Table 5.6. Results of ablation studies on the three datasets.

5.4.4 Ablation Study

We further conduct ablation studies where variants of the proposed method are evaluated to demonstrate the effectiveness of each proposed module and feature. We gradually remove different sub-modules or features from the CTRL model and show the impact of each module. Specifically, apart from the proposed method, i.e., **CTRL**, the following variants are evaluated:

- **-Event_loss.** We remove the event loss and train the model with edge loss, the same as previous works, TGAT, CAW, etc.
- **-Centrality.** We further remove the dynamic centrality sub-module.
- **Hawkes-δ.** We further replace the edge-based Hawkes process with the original Hawkes process where a single trainable decay rate, i.e., δ, is used like TREND.

Table 5.6 summarizes the results of the accuracy, average precision, and F1 score of ablation studies on the ACM, DBLP, and IMDB datasets, respectively. The AUC score is plotted in Figure 5.3. We observe performance degradation each time a sub-module or feature is removed from the previous model. Specifically, the comparison between the **CTRL** and the **-Event_loss** model demonstrates the effectiveness of event-based training for capturing the high-order evolution of temporal HINs. Moreover, the better performance achieved by the **-Event_loss** compared to the **-Centrality** indicates the necessity of integrating the dynamic centrality in the temporal graph. Finally, comparison results between the **-Centrality** and the **Hawkes-δ** model show the advancement of the proposed edge-based Hawkes process over the original Hawkes process on the temporal HINs.

5.5 Conclusion

In this chapter, we propose the CTRL model for continuous-time representation learning on temporal HINs. In the message passing stage of a CTRL layer, node

type- and edge-type-dependent parameters are used to handle the graph heterogeneity. Moreover, in the aggregation steps, we consider the semantic correlation, temporal influence, and dynamic node centrality to determine the importance of neighbour nodes. Finally, we train CTRL with a future event prediction task to capture the evolution of high-order network structure. Extensive experiments on three real-world datasets demonstrate the superiority of CTRL and the effectiveness of the model design.

Chapter 6

Unbiased Sequential Recommendation

6.1 Introduction

In this chapter, we present the DGT and PS-DGT model to address the issues and bridge the research gap introduced in Chapter 1.3. We first construct a temporal heterogeneous network to capture the sequential interactions between users and items, from which two separate representations of each user are extracted for obtaining disentangled embeddings of their preference and disfavour features. Subsequently, we decompose the propensity score of each interaction and leverage prior knowledge of user rating distributions and item rating distributions for a more accurate estimation. Finally, the estimated propensity scores are used both to reweigh samples for unbiased optimization objectives and further calibrate the interaction history in modelling user/item representation. Specifically, we have made the following contributions:

- We focus on the unbiased sequential recommendation in explicit user feedback settings, an area which has not been previously studied and (in addition to the MNAR issue) presents new challenges due to the biases in users' behaviour sequence. As Figure 6.1 illustrates, users tend to express preferences with high

ratings, but rarely give low ones. This leads to a great disparity in the amount of data/interactions between preferences and disfavors. Despite its prevalence, little attention has been paid to this issue for sequential recommendation tasks.

- We propose a novel **Dis**entangled **G**raph **T**ransformer network **DGT** which alleviates the selection in user and item history interaction records, providing unbiased user and item representations. Specifically, user preference and disfavour embeddings are modelled by splitting the interaction records based on their rating scores: interactions with a rating larger than 3 are used to generate the user's preference embedding, while interactions with a rating less than or equal to 3 are used to create the users' disfavour representation. This allows us to learn a disentangled user representation that effectively avoids unbalanced training samples caused by selection bias. The same procedure is applied for each item for creating its disentangled item representation.

- To achieve an accurate estimation of the propensity score and handle other biases, we make a mild assumption that the interaction propensity can be factorized into user rating propensity, item rating propensity, and user-item correlation propensity. An interaction is defined as a user-item-timestamp-rating tuple (u, v, t, r), indicating that user u rated item v with rating r at time t. The user and item rating propensities can be represented by their rating distributions on the training set. The user-item correlation propensity is the probability of user u to rate item v with r at time t, which can be modelled as a classification problem. With the PS estimation module, we build the Propensity Score enhanced DGT, i.e, **PS-DGT**, by applying the estimated PS as weights in loss functions for training an unbiased sequential recommender on observational data (addressing the MNAR issue); additionally, it serves as a weight for calibrating history records of users and items in their representation learning process to further mitigate bias from history.

Through extensive experiments on five bench-marking datasets, we show that our proposed methods have achieved significant improvements on the biased observational test set in terms of both the traditional metrics and unbiased metrics.

6.2 Problem Formulation

In this section, we introduce sequence recommendation problems and temporal heterogeneous graphs. Then, we discuss bias and debiasing in the sequence recommendation from a causality perspective of view under the potential outcome framework.

6.2.1 Sequential Recommendation

We first formulate the sequential recommendation problem with explicit user feedback. Let $U = \{u_1, \cdots, u_N\}$ and $V = \{v_1, \cdots, v_M\}$ denote the sets of users and items where N and M are the total number of users and items. For a user $u \in U$ and an item $v, \in V$, a tuple (u, v, t, r) denotes that user u interacted with item v at timestamp t with an user feedback (rating score) $r \in \{1, 2, 3, 4, 5\}$. Formally, in the explicit feedback scenario, a sequential recommendation model predicts a user's rating for an item at a given timestamp based on the current features of the item and the user, that is,

$$\hat{r}^t_{u,v} = SRec(\text{feat}^t_u, \text{feat}^t_v), \tag{6.1}$$

where $\hat{r}^t_{u,v}$ is the predicted rating by the sequential recommendation model $SRec$. feat^t_u and feat^t_v are the features of the user and the item at timestamp t which may include context information, e.g., user profile and item attributes, and history interaction records. We adopt the mean square error as the loss function to optimize the model parameters,

$$\mathcal{L} = \frac{1}{|S|} \sum_{(u,v,t,r)\in S} ||r^t_{u,v} - \hat{r}^t_{u,v}||_2, \tag{6.2}$$

where S is the training set and $|S|$ denotes the number of sample in S.

6.2.2 Temporal Heterogeneous Graph

In this work, we model the sequential recommendation as a link prediction problem on a temporal heterogeneous graph (THG). A THG is defined as a graph $\mathcal{G} =$

$(\mathcal{V}, \mathcal{E}, \mathcal{T}; \phi, \varphi)$ where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ indicates the set of edges and $\mathcal{T}$ represents the set of timestamps. $\phi : \mathcal{V} \mapsto \mathcal{A}$ and $\varphi : \mathcal{E} \mapsto \mathcal{R}$ are two mapping functions that map nodes and edges to their corresponding types. $\mathcal{A}, \mathcal{R}$ are the sets of types for nodes and edges, respectively. For HINs, $|\mathcal{A}| + |\mathcal{R}| > 2$. In explicit feedback-based sequential recommendations, we have two types of nodes, i.e., user and item, and a single edge type, i.e., user-item interaction. Our goal is to predict the rating score $r \in \{1, 2, 3, 4, 5\}$ for a given edge.

6.2.3 Bias and Debias in Sequential Recommendation

6.2.3.1 Bias in Sequential Recommendation.

Inspired by the notions in previous literature [27], [83], we introduce three variables: the interaction variable $I_{u,v,t,r}$, the relevance variable $R_{u,v,t} = r \in \{1, 2, 3, 4, 5\}$ and the observation variable $O_{u,v,t,r}$. $I_{u,v,t,r} = 1$ represents an interaction happens between user u and item v at time t with a user feedback of r, and 0 otherwise. $R_{u,v,t} = r$ denotes the user's feedback on the item at the given time. $O_{u,v,t,r} = 1$ means the interaction (u, v, t, r) is observed the item at t, and 0 otherwise. At timestamp t, an interaction (u, v, t, r) happens when user u observes and rates the item v with rating r at t. Thus, the probability of observing a tuple (u, v, t, r) can be formulated as:

$$P(I_{u,v,t,r} = 1) = P(O_{u,v,t,r} = 1, R_{u,v,t} = r) \tag{6.3}$$

In the ideal distribution P^*, all the users are exposed to the entire item set at any given time and users are always willing to share their true feedback. This means all interactions are observable, i.e., $P^*(O_{u,v,t,r} = 1) = 1$ and $P(I_{u,v,t,r} = 1) = P(R_{u,v,t} = r)$ which indicates the interactions reflect users' true behaviour patterns. Therefore, we can extract users' true preferences by minimizing the loss function defined in (6.2) on a training set S^* collected from the ideal distribution P^*.

Missing Not At Random. However, in real scenarios, each user is only exposed to a small subset of the item dataset due to its vast size. Moreover, various biases, e.g., selection bias [74], exposure bias [81], and conformity bias [84], occur

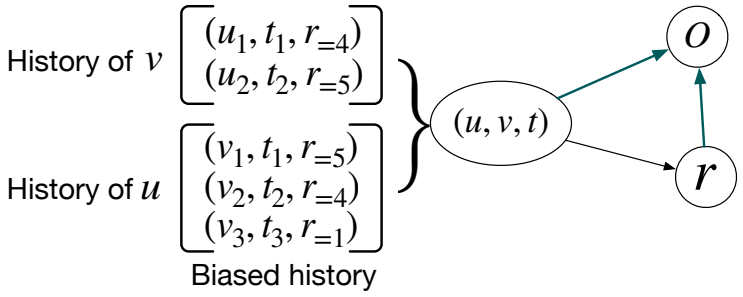


Figure 6.1. Causal graph in the sequential recommendation with explicit feedback. User u tends to express its preferences rather than dislikes leading to a biased history.

in the data generation procedure (exposure and data collection) of real-world recommendation systems [85] making the unobserved samples missing not at random (MNAR) and resulting in the observed data distribution P a skewed version of the ideal unbiased distribution P^*. This means the observed training set S is also biased leading to biased representations of users and items as well as poor recommendation performance.

Biased History Records. In sequential recommendation, the interaction histories of users and items are often encoded as their features (or latent representations) for recommendation at the current time. However, the historical interactions are also full of biases. For example, as shown in Figure 6.1, with selection bias, some users may tend to share their positive feedback (high rating scores) rather than negative ones (low rating scores) resulting in unbalanced history records for both users and items. This unbalanced history records would lead to biased user representation where users' disfavour features are overwhelmed by the preference features resulting in poor accuracy when predicting users' rating on their disliked items. This is a common issue in sequence recommendation, but it has been neglected by previous unbiased sequential models such, e.g., USR [27] and DEPS [28].

6.2.3.2 Potential Outcome Framework.

The Potential Outcome Framework (POF) [152] is a powerful and effective tool to uncover the underlying causality from MNAR observational datasets. Specifically,

we leverage the POF for unbiased sequential recommendation by configuring the following key components of the POF, they are:

- *Unit*. It is the basic studied objective. Here, we define a user-item-time triple (u, t, v) as the study unit.
- *Treatment*. It denotes an action imposed on the unit which is defined as the observational variable $O \in \{0, 1\}$.
- *Potential outcome*. This is the result of applying the treatment on the unit which is the rating score r in our problem setting.
- *Covariates*. They are random variables that the *treatment* is known to have no effect on them. Generally, they are attributes or pre-treatment variables of the unit or environment. In sequential recommendations, the *Covariates* are user and item features including their IDs and historical interaction records.

Under the POF framework, the unbiased sequence recommendation problem is transformed into estimating the potential outcome of each unit when it is assigned with the treatment ($O = 1$). That is, $Y_{(u,v,t)}(O = 1|\text{feat}_u^t, \text{feat}_v^t)$ where feat_v^t is the feature/covariates of item v at timestamp t.

As is shown in the causal graph in Figure 6.1, we note that in the sequential recommendation scenario, the observational variable O (treatment) is dependent both on the user-item features (exposure bias) and the rating score (selection bias).

6.2.3.3 Propensity Score

The ideal randomized experiments, as illustrated by the straight line in Figure 6.2, assume that the observational probability is independent of the covariates. Nonetheless, in real scenarios, the observational probability of an interaction is often dependent on its covariates, leading to the issue of missing not at random. To address this problem, a commonly used method is to approximate the randomized experiment by re-weighting the observed distribution via the propensity score.

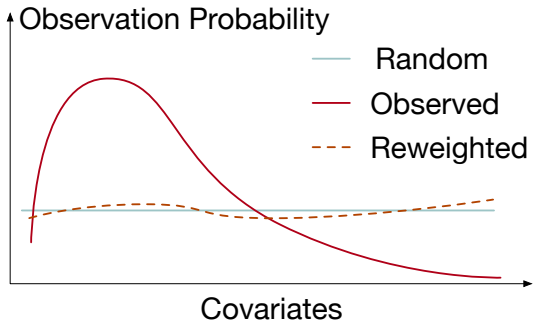


Figure 6.2. Example on how to achieve unbiased estimation from biased observed data via PS re-weighting.

The propensity score is defined to be a function of the covariates as

$$P(u, v, t, r) := P(O = 1|u, v, t, r), \tag{6.4}$$

That is, $p(u, v, t, r)$ denotes the probability unit (u, v, t) will be associated with $O = 1$ (be observed) conditioned on the given covariates (u, v, t, r). With the propensity score, we formulate the unbiased loss function for an unbiased estimation of the potential outcome from observational data.

$$\begin{aligned} \mathcal{L}^{unbias} &= \frac{1}{|S|} \sum_{O=1} \frac{P^*(O = 1; u, v, t, r)}{P(O = 1; u, v, t, r)} ||r - \hat{r}||_2, \\ &\propto \frac{1}{|S|} \sum_{O=1} \frac{1}{p(u, v, t, r)} ||r - \hat{r}||_2, \end{aligned} \tag{6.5}$$

where S is the observed dataset.

Re-weighting the observed distribution to approximate the ideal unbiased distribution (distribution of randomized experiment) through IPS [86], [87] has been a widely used and efficient method for tackling various biases in traditional recommendations. USR [27] and DEPS [28] extend this IPS method into sequential recommendation scenarios, using user behaviour histories to predict the IPS. However, they train the IPS module based on biased observed data, making it unreliable. Furthermore, they ignore that user history is already generated in a biased environment; thus it is difficult to extract unbiased user preferences from biased input

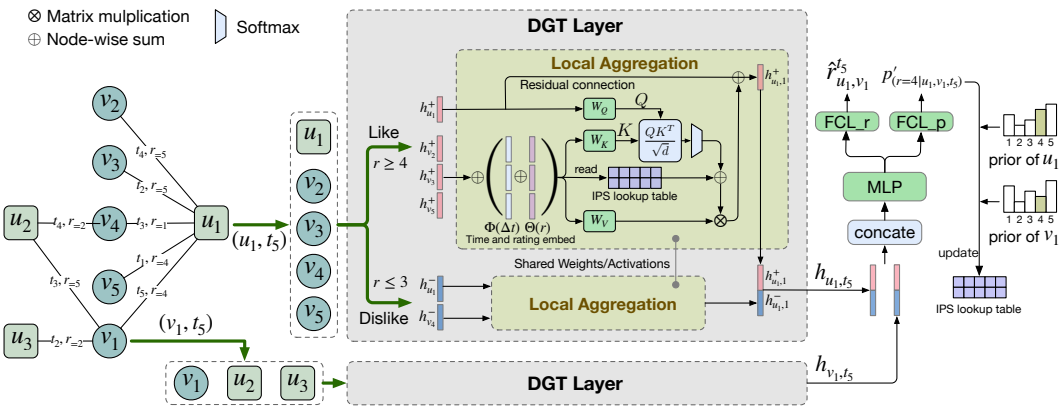


Figure 6.3. The architecture of a k-layer ($k = 1$) PS-DGT model. An example of predicting the rating of user u_1 on item v_1 at timestamp t_5 is illustrated.

features (interaction histories).

6.3 Methodology

In this section, we first introduce the model architecture of the proposed unbiased sequential recommendation model PS-DGT. Then, we will discuss the estimation of the propensity score and how it can be used to help reduce various biases.

6.3.1 Disentangled Graph Transformer

Shown in Figure 6.3, the interactions between users and items are represented as a temporal heterogeneous graph where an edge $e = (u, i, t, r)$ denotes that user u rates item i with r at timestamp t. An example is given on how we extract the disentangled latent representations of a user-item pair to predict the user's rating on the item at a given timestamp. To be specific, given user u_1 and item v_1 at timestamp t_5, we first extract the disentangled temporal representations for both u_1 and v_1 through a temporal graph neural network. Two latent embeddings are extracted to represent the user's preference and disfavour patterns which indicate what the user likes and dislikes. For example, historical neighbours with rating score $r \geq 4$, $\{(v_2; t_4, r = 5), (v_3; t_2, r = 5), (v_4; t_3, r = 1), (v_5; t_1, r = 4)\}$ for u_1, are selected to model the preference of a user and neighbours with a rating score

$r \leq 3$, $\{(v_4; t_3, r = 1)\}$ for u_1, are selected to model the disfavour embedding of a user. Then, the two embeddings of the user are concatenated to form the final disentangled representation. Representations of a user-item pair are concatenated and fed into an MLP module to predict the user's rating on the item at the given timestamp.

6.3.1.1 Learning Preference Embedding

For user $u \in U$ with its historical neighbors with positive feedback $\{v_i, t_i, r_i\}^{t_i<t}, r_i \geq 3$ at timestamp t. We create two trainable embedding matrices $\mathbf{V}^+ \subset \mathbb{R}^{\mathbf{N}\times\mathbf{d}}$ and $\mathbf{U}^+ \subset \mathbb{R}^{\mathbf{M}\times\mathbf{d}}$ to represent the preference features of users and items. Note that, d represents the number of dimensions in the latent space and N and M are the number of users and items, respectively. Both the user and its neighbours (items) are converted into latent embeddings by looking up the embedding tables $\mathbf{U}^+$ and $\mathbf{V}^+$,

$$\begin{aligned} \boldsymbol{h}^+_{u,t,0} = \boldsymbol{h}^+_{u,0} &= \mathbf{U}(\mathbf{u}), \\ \{\boldsymbol{h}^+_{v_i,t,0}\}^{t_i<t} &= \{\mathbf{V}(\mathbf{v_i})\}^{\mathbf{t_i}<\mathbf{t}}, \end{aligned} \tag{6.6}$$

where $\boldsymbol{h}^+_{u,t,0}$ and $\boldsymbol{h}^+_{v_i,r,0}$ denote the preference features of user u and item v_i at timestamp t. Note that we use bold font to represent vector variables.

Heterogeneous Attention We adopt the dot-product attention layer to weigh the contribution of the features from the neighbours in the local aggregation of the GNN, the attention score is calculated as:

$$\begin{aligned} \mathbf{q} &= W^Q \boldsymbol{h}^+_{u,t,0}, \\ \mathbf{k} &= \{W^K \boldsymbol{h}^+_{v_i,t,0} \oplus \Phi(t - t_i) \oplus \Theta(r_i)\}^{t_i<t}, \\ \mathbf{v} &= \{W^V \boldsymbol{h}^+_{v_i,t,0} \oplus \Phi(t - t_i) \oplus \Theta(r_i)\}^{t_i<t}, \\ \text{attn} &= \text{softmax}(\frac{\mathbf{q}\mathbf{k}^\mathbf{T}}{\sqrt{d}}), \end{aligned} \tag{6.7}$$

where W^Q, W^K, and W^V are the transformation parameters for the *Query*, *Key*, and *Value* terms of the dot-product attention. d is the dimension of the hidden rep-

resentations. $\Phi(\cdot)$ is the time encode function which maps time intervals, $t - t_i$, into real-valued vectors of the size of d and $\Theta(\cdot)$ is the rating embedding function that maps the rating values into vectors. $\oplus$ denotes the element-wise addition operation.

History calibration via Inverse Propensity Weight (IPW). As introduced in Sec 6.2.3, the historical neighbours of the node in the temporal graph are unbalanced due to selection bias. Therefore, we further leverage the inverse propensity score to reweigh each neighbour for an unbiased node representation. To be specific,

$$IPW = \text{softmax}(\{\frac{1}{p(u, v_i, t_i, r_i)}\}^{t_i<t}), \tag{6.8}$$

where $p(u, v_i, t_i, r_i)$ is the propensity score defined in (6.4). The estimation module of the propensity score is introduced in the next subsection.

Therefore, the preference embedding of user u as a single PS-DGT layer is computed as:

$$\boldsymbol{h}^{+}_{u,t,1} = ((1-\alpha) * \text{attn} + \alpha * IPW) * \mathbf{v} + \mathbf{h}^{+}_{\mathbf{u},\mathbf{t},\mathbf{0}}, \tag{6.9}$$

where $\alpha \in (0, 1)$ is a hyper-parameter that controls the ratio of the context-based attention score attn and IPW. $\boldsymbol{h}^{+}_{u,t,1}$ is the output of the first PS-DGT layer. Generally, we can stack multiple (k) PS-DGT layers and the output of the last layer is denoted as $\boldsymbol{h}^{+}_{u,t}$ which is the final preference embeddings of user u at timestamp t.

6.3.1.2 Learning disfavor Embedding

Similarly, we create another two trainable embedding matrices $\mathbf{V}^{-} \subset \mathbb{R}^{\mathbf{N}\times\mathbf{d}}$ and $\mathbf{U}^{-} \subset \mathbb{R}^{\mathbf{M}\times\mathbf{d}}$ to represent the disfavour features of users and items. Following the same procedure, the disfavour embedding of user u is extracted as $\boldsymbol{h}^{-}_{u,t}$ after k PS-DGT layers. Note that, all parameters and functions are shared when learning preference and disfavour embeddings, except for the embedding tables.

The final user representation is then the concatenation of its preference embedding and disfavour embedding. That is, $\boldsymbol{h}_{u,t} = \boldsymbol{h}^{+}_{u,t} \frown \boldsymbol{h}^{-}_{u,t}$ where $\frown$ is the

concatenation operation.

The same procedure is conducted for items to get the disentangled item representation $h_{v,t}$. Finally, we build an MLP module for the rating prediction task based on the extracted representations of the user and item. Specifically, we have

$$\hat{r}^t_{u,v} = \text{FCL}_r(\text{MLP}(\boldsymbol{h}_{u,t} \frown \boldsymbol{h}_{v,t})), \tag{6.10}$$

where $\hat{r}^t_{u,v}$ is the predicted rating for the triple (u, v, t).

As depicted in Figure 6.3, we employ the same graph neural network (with shared weights/activations) when extracting preference and disfavour embeddings. This is because the PS-DiGAN layer is used to extract patterns from temporal neighbours of nodes, thus making extracting preference and disfavour embeddings two similar tasks. Furthermore, if we are to train separate layers for disfavored embeddings, we will be confronted with the challenge of inadequate training samples caused by selection bias; therefore, we use the same layers for modelling both embeddings.

Through the disentangled representations, we separately model users' preferences and disfavour features, effectively preventing the disfavour features from being overwhelmed by the preference features due to lack of low score ratings, which in turn greatly mitigate the selection bias introduced in Sec. 6.2.3. Furthermore, with the help of the IPS, we are able to reweigh the neighbour's importance in the local aggregation of GNN for unbiased representation.

6.3.2 Estimation of Propensity Score

As introduced in Sec 6.2.3, with the propensity score, we can have an unbiased estimation of the potential outcome from the biased observational dataset. However, the propensity score is a complex mixture of different biases such as the exposure mechanism and selection bias.

To this end, for an interaction (u, v, t, r), we first factorize the interaction probability, introduced in (6.3), into user rating propensity $P(O_{u,r})$, item rating propen-

sity $P(O_{v,r})$, and user item correlation $P(R_{u,v,t} = r)$:

$$\begin{aligned} P(u, v, t, r) &= P(O_{u,v,t,r} = 1) = P(O = 1 | R_{u,v,t} = r) \\ &= P(O_{u,r} = 1) P(O_{v,r} = 1) P(R_{u,v,t} = r) \\ &\propto N_{u,r} N_{v,r} P(R_{u,t,i} = r), \end{aligned} \tag{6.11}$$

where $P(O_{u,r})$ and $P(O_{u,r})$ are the probabilities of user u giving the rating r and item v be rated with r. $P(R_{u,v,t} = r)$ is the probability of u giving a rating r to item v at time t. $N_{u,r}$ denotes the number of observed samples where u gives a rating r, similarly, $N_{v,r}$ is the number of times when v is given a rating r.

$N_{u,r}$ and $N_{v,r}$ are easily acquired prior knowledge of users and items. Therefore, we focus on estimating the rating probability of a user on an item at a given time. Specifically, we build another fully connected layer FCL_p which takes the output of the MLP module, introduced in (6.10), as input to predict the rating probability. That is,

$$P'(R_{u,v,t} = r) = \text{FCL}_p(\text{MLP}(\boldsymbol{h}_{u,t} \frown \boldsymbol{h}_{v,t})), \tag{6.12}$$

Then, the loss of the probability prediction is formulated as:

$$\mathcal{L}_{\text{PS}} = - \sum_{r \in \{0,1,2,3,4,5\}} y^{(r)} log(P'(R_{u,v,t} = r)), \tag{6.13}$$

where $y^{(r)} \in 0, 1, \sum_r y^{(r)} = 1$ is the true rating. $r = 0$ indicates that the corresponding sample is unobserved, i.e., the negative sampled training unit.

Finally, we incorporate the priors of the users and items and apply softmax to get the final propensity score.

$$p = \text{softmax}(\{N_{u,r} N_{v,r} P'(R_{u,t,i} = r)\}_{(u,v,r,t) \in S}), \tag{6.14}$$

where S is the observed set. Moreover, to avoid high variance caused by small PS, we further apply a clip operation on the PS score smaller than a given threshold

Algorithm 3: Training of the PS-DGT

Input: User set U, and item sets: V;
Interaction set: $(u, v, t, r) \in S$;
Latent embedding size: $d > 0$.
Stage 1:
Optimize the DGT model with the original loss (6.2).
Stage 2:
Fix the parameters of the DGT and MLP modules.
Optimize the parameters of the FCL_p layer with the cross-entropy loss (6.13).
Update the PS lookup table with the clipped PS score (6.15).
Stage 3:
Fine-tune the PS-DGT loss with the unbiased loss (6.5).

$\beta \in (0, 1)$, i.e.,

$$p^{clip}(u, v, t, r) = max(p(u, v, t, r), \beta). \tag{6.15}$$

6.3.3 Training Strategy

We propose a three-phase training algorithm shown in Algorithm 3 to optimize the parameters of DGT and the PS module.

In the first phase, for stable training, we pre-train the DGT model without the PS module through the mean square error loss (6.2). Through the pre-training process, DGT produces a well-embedded history of users and items as their features, producing a boost-start for the following PS module. In the second phase, we fix the parameters of the DGT as well as the MLP module and optimize the FCL_p to accurately predict the rating probability. Along with the prior knowledge of users and items, we update the PS lookup table for the unbias sequential recommendation in the next step. Finally, we incorporate the propensity score in the local aggregation procedure of the DGT to form the final PD-DGT model. Then, the PS-DGT is tuned on the observed set with unbias loss defined in (6.5). Note that, in the fine-tuning, the PS lookup table is updated per epoch.

Table 6.1. Statistics of the Datasets.

Dataset	#Users	#Items	#Interactions	Sparsity(%)	Avg.Int
CD	26,193	63,592	774,591	99.95	29.57
Music	7471	32,905	151,858	99.94	20.33
Beauty	5925	5,398	48,859	99.85	8.246
Movie	40,593	49,848	1,168,233	99.94	28.78
Sport	93,233	195,398	1,521,104	99.99	16.32

6.4 Experiments

We performed in-depth experiments on five publicly available datasets and compared the performance of our approaches to state-of-the-art, unbiased sequential recommendation methods.

We adopt five publicly available Amazon datasets [133] to evaluate the proposed model and all the baseline methods. Table 6.1 summarizes the detailed statistics of the five datasets where "Avg.Int" denotes the average number of interactions of users. Following common practices in sequential recommendation [16], [34], users and items with less than five interactions are filtered out. Moreover, for a given user, the second to last item in the behaviour sequence is selected as the validation item, and the last item is used for testing, while the remaining items are used for training. Figure 6.4 illustrates the sample distributions of our experimental datasets, revealing a notable discrepancy in the amount of data obtained from different rating levels across all datasets. This selection bias, caused by users' tendency to express their positive feedback rather than dislikes, could lead to a decrease in performance when predicting ratings.

6.4.1 Experimental Setup

Our experiments aim at answering the following research questions:

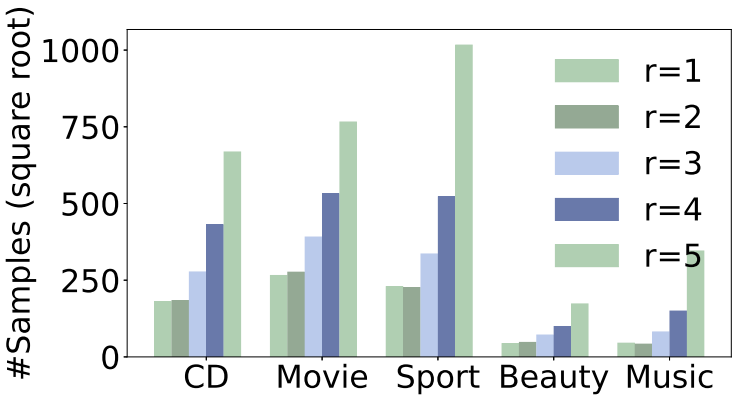


Figure 6.4. Sample distributions of the five Amazon datasets.

- **RQ1.** How does our method perform compare to state-of-the-art baselines in terms of both traditional and debias metrics?
- **RQ2.** How do the sub-modules help the model succeed in alleviating the bias issue in the sequential recommendation?
- **RQ3.** How do hyperparameters affect model performance?

Similar to existing rating prediction work, we use mean squared error (MSE) and mean absolute error (MAE) as standard evaluation metrics on the biased observed set. To assess a model's performance on an unbiased set, prior studies usually re-sample the test data; however, this step is unnecessary in the explicit feedback scenario. Specifically, we utilize macro-MSE and macro-MAE as bias-free evaluation metrics, which are defined as follows:

$$\text{Macro-Metric} := \frac{1}{|R|} \sum_{r \in R} \frac{1}{S_r} \sum_{(u,v,t,r) \in S_r} \text{Metric}(\hat{r}, r), \tag{6.16}$$

where Metric represents MSE or MAE, $R = \{1, 2, 3, 4, 5\}$ and S_r is the test set where all interactions are with the rating r.

To showcase the efficacy of the proposed approach, we conducted comparative studies with a variety of baseline methods, including traditional models, sequential recommenders and unbiased sequential methods. Specifically, the following methods were evaluated and compared:

- **MF**: The classical matrix factorization method for explicit feedback recommendation.
- **NMF** [12]: The neural matrix factorization model, which adopts multiple fully connected layers to predict the ratings based on user and item embeddings.
- **SAS** [16]: It adopts unidirectional self-attention to model user preference from its interaction history.
- **SAS_rate** [16]: We integrate rating embeddings into the original SAS model for better performance in our setting.
- **LightGCN** [23]: It simplifies the Graph Convolution Network (GCN) model for better performance in the recommendation task.
- **USR** [27]: The first unbiased sequential recommendation method.
- **DEPS** [28]: The state-of-the-art model for unbiased sequential recommendation.

To ensure reliable results, all experiments were conducted three times and the mean values of all metrics were reported.

6.4.2 Implementation Details

Environment. We implement our method using PyTorch with python 3.6 and train the model on Tesla P40 GPU with a memory size of 22.38 GiB and a 1.53 GHz memory clock rate.

Model implementation. For a fair comparison, we use a single PS-DGT layer in our model, and a maximum number of 50 temporal neighbours are sampled for message passing and local aggregation. We use 2 attention heads for all the attention layers in the PS-DGT layer. The dimensionality of the final user or item embedding is set to $d = 64$, which means the size of the preference embedding and disfavour embedding is $32 = d/2$. To avoid over-fitting, we adopt a drop rate of 0.1 in the

aggregation of each GNN layer. Moreover, we use early-stop to select the best model. Specifically, we stop training when the validation loss does not decrease for 5 consecutive epochs and the maximum training epochs is 50. Additionally, we use grid search to find the optimal clip rate β, introduced in (6.15). The search range and results are given in Sec. 6.4.5. The IPS weighting parameter α is set to 0.1 on the "Music" and "CD" datasets and is set to 0.02 on the rest of the datasets after searching in the range of $[0.01, 0.5]$. Adam optimizer with a learning rate of 1×10^{-3} is used in the first two training phases introduced in Algorithm 3. In the final fine-tuning step, the learning rate is set to 1×10^{-4}. We adopt a trainable embedding table for all the five rating scores $\Theta(\cdot)$. a function that maps a scalar to a real-valued vector is used for time encoding the $\Phi(\cdot)$. To reduce computational complexity, as opposed to applying the *softmax* operation to the entire dataset (as specified in equation (6.14)), we opted to apply the softmax to 1024 training samples per training batch.

The MLP module in Figure 6.3 consists of hidden layers with the size of $[2 * d, 64, 32]$, where d is the dimension of the user and item embedding. For a fair comparison, all evaluated methods, except for the MF model, are with the same MLP module as the output layers for the rating prediction task. Moreover, they all use the same dimensionality (64) for the latent user and item representations. All models were trained through the mean square error with the Adam optimizer. For sequential recommendation baselines, the maximum sequence length is also set to 50.

6.4.3 Main Results

Table 2 summarizes the results of the proposed models (DGT and PS-DGT) and all the baselines on the five datasets. The proposed DGT method significantly outperforms all baselines including the state-of-the-art unbiased sequential recommendation methods, i.e., USR and DEPS, across all five datasets with both the biased metrics and unbiased metrics which is remarkable (**RQ1.**). Generally speaking, in order to remove bias and achieve good performance in unbiased evaluation, a model

Table 6.2. Performance comparisons on the five amazon datasets between the proposed models (DGT and PS-DGT) and the baseline methods. All numbers are in percentile (%). "m-MSE/MAE" denotes "Macro-MSE/MAE".

Datasets	Metric	Traditional Baselines				
		MF	NMF	SAS	SAS_rate	LightGCN
Sport	MSE	6.36	4.57	5.07	4.38	5.51
	MAE	13.73	14.41	16.29	14.26	14.71
	m-MSE	22.47	14.6	16.25	14.1	15.17
	m-MAE	38.45	30.53	33.04	29.95	30.67
Movie	MSE	8.63	4.64	5.91	4.86	5.01
	MAE	17.27	15.34	18.15	15.77	15.29
	m-MSE	20.66	9.04	12.18	9.96	9.04
	m-MAE	36.52	23.26	28.03	24.69	22.84
CD	MSE	6.6	4.13	5.03	4.35	4.43
	MAE	14.26	14.27	16.46	14.54	14.35
	m-MSE	20.49	10.81	13.55	11.91	10.12
	m-MAE	36.43	25.82	29.78	27.27	24.41
Music	MSE	4.61	1.49	2.08	1.4	1.74
	MAE	12.38	6.4	9.68	6.35	6.55
	m-MSE	14.11	12.74	18.77	12.51	12.53
	m-MAE	29.54	27.18	35.65	27.07	27.09
Beauty	MSE	15.81	4.79	5.2	4.09	5.0
	MAE	27.3	14.21	17.25	14.01	13.96
	m-MSE	15.59	11.74	14.89	10.59	12.32
	m-MAE	30.34	26.06	31.43	25.03	26.83

usually sacrifices part of its performance on the observed biased test set. That is, there is a trade-off between the performance of the traditional metric and the unbias metrics. This is why the DEPS model would outperform traditional baselines on the unbiased evaluation metrics Macro-MSE/MAE yet fails to outperform traditional methods on normal metrics, MAE and MSE. Moreover, without explicitly taking the selection bias into account, USR and DEPS are outperformed by the modified SAS_rate model. Nevertheless, our DGT can achieve the best performance under both the traditional metrics and the unbiased metrics, which is outstanding. The results of DGT demonstrate its effectiveness in mitigating selection bias, which can be attributed to its disentangled representation. The success of DGT is significant and remarkable, making it a promising approach for dealing with selection bias in

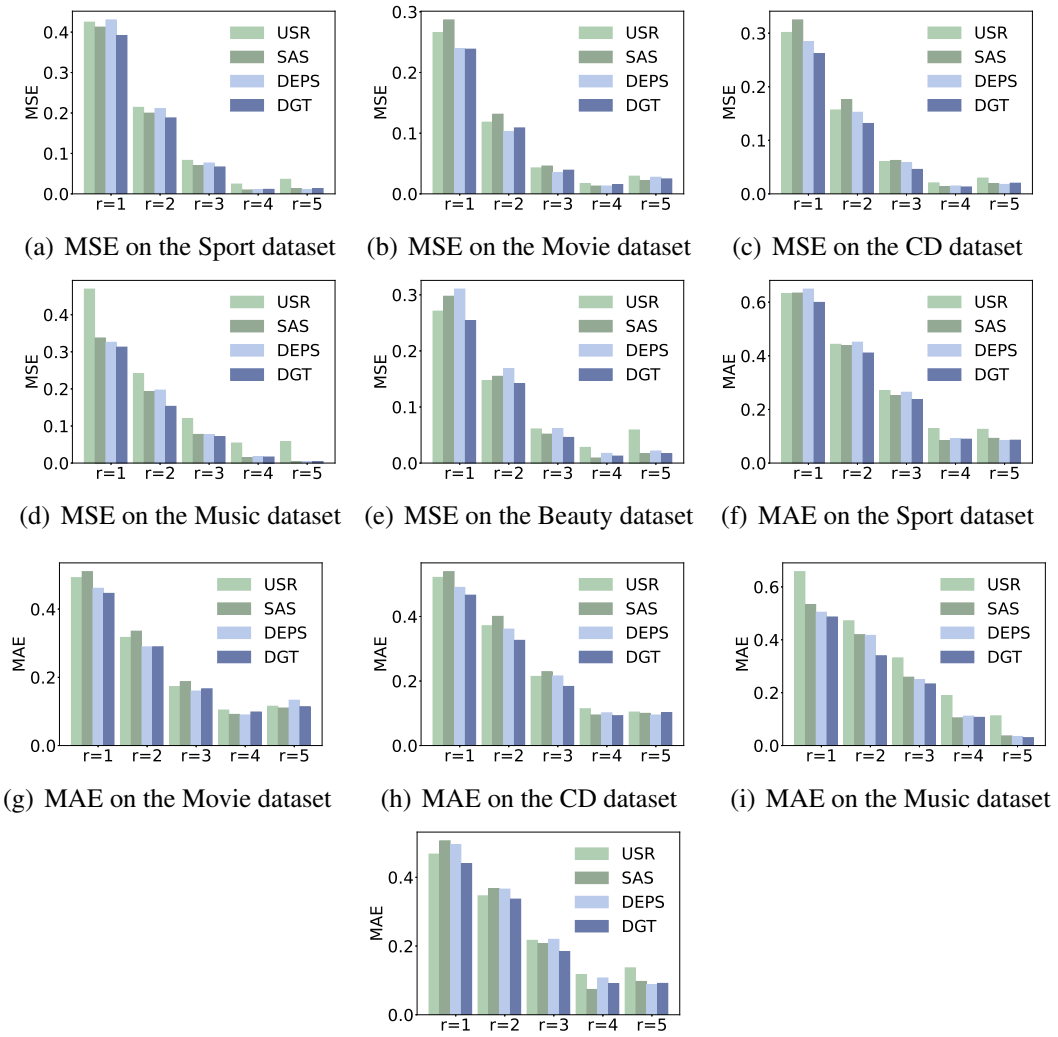
Table 6.3. Performance comparisons on the five amazon datasets between the proposed models (DGT and PS-DGT) and the baseline methods. All numbers are in percentile (%). "m-MSE/MAE" denotes "Macro-MSE/MAE".

Datasets	Metric	Unbias Baselines		Proposed	
		USR	DEPS	**DGT**	**PS-DGT**
Sport	MSE	6.37	4.39	**4.22**	4.38
	MAE	17.39	14.01	**13.51**	15.16
	m-MSE	15.63	14.78	*13.4*	**11.52**
	m-MAE	31.99	30.77	*28.42*	**26.32**
Movie	MSE	5.06	4.56	**4.51**	4.67
	MAE	15.85	16.09	**15.16**	16.42
	m-MSE	9.45	8.34	8.52	**7.58**
	m-MAE	24.04	22.66	*22.29*	**21.19**
CD	MSE	4.92	4.01	**3.76**	3.88
	MAE	14.8	13.9	**13.54**	14.27
	m-MSE	11.37	10.53	*9.44*	**8.41**
	m-MAE	26.5	25.26	*23.42*	**21.98**
Music	MSE	6.71	1.45	**1.3**	1.33
	MAE	14.14	6.25	**5.61**	6.39
	m-MSE	18.85	12.39	*11.13*	**10.18**
	m-MAE	35.27	26.33	*23.92*	**22.73**
Beauty	MSE	6.98	4.72	**3.8**	3.74
	MAE	16.99	13.99	**13.19**	13.78
	m-MSE	11.31	11.57	*9.4*	**8.32**
	m-MAE	25.68	25.41	*22.85*	**21.41**

recommendation systems.

Furthermore, we incorporate the PS module into the DGT model to further improve the performance of the performance under the unbiased evaluation metrics with a little sacrifice of performance on the normal metrics which is reasonable due to the trade-off as mentioned above. Specifically, with the help of the PS module in both reweighing the observed samples and calibrating interaction history, the PS-DGT further boost the performance on unbiased metrics. Moreover, PS-DGT still achieved competitive results and outperformed most of the baselines on traditional metrics.

The plot in Figure 6.5 provides an in-depth view of how the model performs under different ratings. The prediction error decreases as the rating score increase



(a) MSE on the Sport dataset (b) MSE on the Movie dataset (c) MSE on the CD dataset

(d) MSE on the Music dataset (e) MSE on the Beauty dataset (f) MAE on the Sport dataset

(g) MAE on the Movie dataset (h) MAE on the CD dataset (i) MAE on the Music dataset

(j) MAE on the Beauty dataset

Figure 6.5. Model performance on test sets with different rating scores.

which is reasonable and correspond to the data distribution in Figure 6.4. Moreover, we can see that our DGT method achieved the best performance the most of time in different rating levels and metrics.

6.4.4 Ablation Study

In answering **RQ2.**, we further conduct ablation studies where variants of the proposed method are evaluated on the five datasets to demonstrate the effectiveness of each proposed module and feature. We gradually remove different sub-modules

or features from the PS-DGT model and show how the results are affected. To be specific, the following variants are evaluated:

- **PS-DGT**: The full proposed unbias sequential recommendation model.
- **-PS-loss**: In which we use the simple MSE loss (6.2) to train our model instead of the unbias loss function (6.5).
- **-PS-attn**: We further drop out the propensity score in the local aggregation of a GNN layer. This is the DGT model.
- **-Disentangle**: In this variant, we further remove the disentangled representation module in DGT and extract a single user or item embedding from all its temporal neighbours (with a maximum number of 50).

We have observed a significant improvement in the performance of the **-PS-attn** compared to the **-Disentangle** model when evaluated on unbiased metrics, without compromising on the performance of standard metrics. This demonstrates the effectiveness of the disentangled representation in both alleviating selection bias and for accurate overall user representation. Additionally, inducing the PS module in either the DGT layer (**-PS-loss**) or the optimization objective (**PS-DGT**) further boost the performance in terms of the unbiased metrics while leading to a small degradation in normal metrics due to the trade-off between them as mentioned before. Nevertheless, our final model is able to achieve decent performance in terms of standard metrics and significantly outperforms all baselines as shown in Table 6.2 and 6.3.

6.4.5 Influence of the Hyper-parameters

In this subsection, we conduct experiments to study the influence of the hyper-parameters introduced in the model design (**RQ3.**).

Clip rate β. We know that a small propensity score would lead to high variance in the model training. The clip rate is used to control the trade-off between the bias and variance in the model training. To study its effect on model performance, we tune it in the range of $\{0.02, 0.04, 0.06, 0.08, 0.1\}$. The results are given in Figure

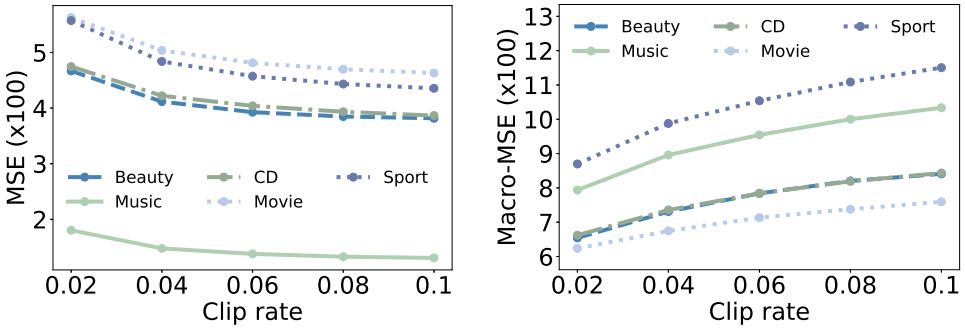
Table 6.4. Results (in %) of ablation studies on the five datasets.

Models	Metric	Sport	Movie	CD	Music	Beauty
PS-DGT	MSE	4.38	4.67	3.88	1.33	3.74
	MAE	15.16	16.42	14.27	6.39	13.78
	Macro-MSE	11.52	7.58	8.41	10.18	8.32
	Macro-MAE	26.32	21.19	21.98	22.73	21.41
-PS-loss	MSE	4.28	4.55	3.79	1.3	3.81
	MAE	13.59	15.08	13.44	5.54	13.07
	Macro-MSE	13.17	8.58	9.39	11.07	9.52
	Macro-MAE	28.04	22.27	23.16	23.81	22.93
-PS-attn	MSE	4.22	4.51	3.76	1.3	3.8
	MAE	13.51	15.16	13.54	5.61	13.19
	Macro-MSE	13.4	8.52	9.44	11.13	9.4
	Macro-MAE	28.42	22.29	23.41	23.92	22.85
-Disentangle	MSE	4.16	4.42	3.8	1.29	3.99
	MAE	13.89	15.43	13.85	5.52	13.52
	Macro-MSE	13.5	8.6	9.72	11.61	10.13
	Macro-MAE	28.95	22.58	24.21	24.52	23.89

6.6. We can see that a smaller clip rate indicates a greater degree of debiasing resulting in a smaller macro-MSE and increased MSE, which is reasonable and in line with previous methods. In our primary experiments, we set the clip rate of the proposed DGT and PS-DGT models to 0.06 across all five datasets for optimal performance with respect to both traditional and unbiased metrics.

Embedding size d. We then explored the effect of embedding size on model performance, varying it between $\{16, 32, 64, 128\}$ in the DGT model. The results are displayed in Figure 6.7; as expected from previous studies, larger embeddings generally produced better results (lower error values), though with a relatively small improvement. To prevent unnecessarily high model complexity and ensure good performance, we opted to use an embedding size of 64 for all evaluated models.

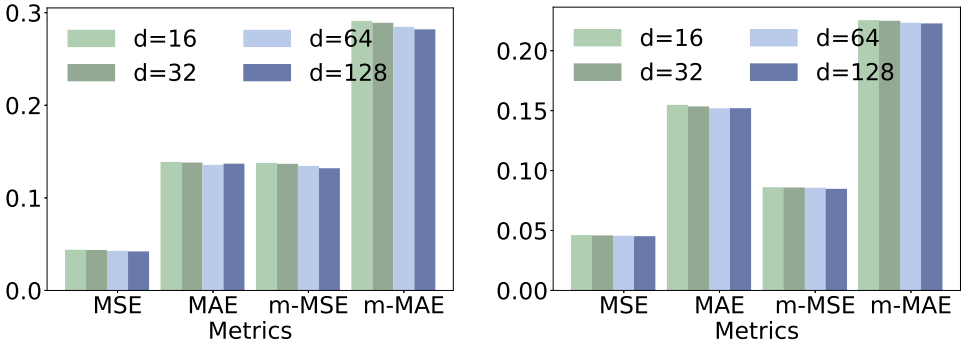
Number of GNN layers. Finally, we analyze the influence of the number of GNN layers in our DGT model. In GNN-based recommender systems, it is typically seen that the number of GNN layers used is $k \in \{1, 2\}$. From Table 6.5, we observe that with a deeper GNN architecture, DGT enhances performance on both



(a) MSE results on the five sets

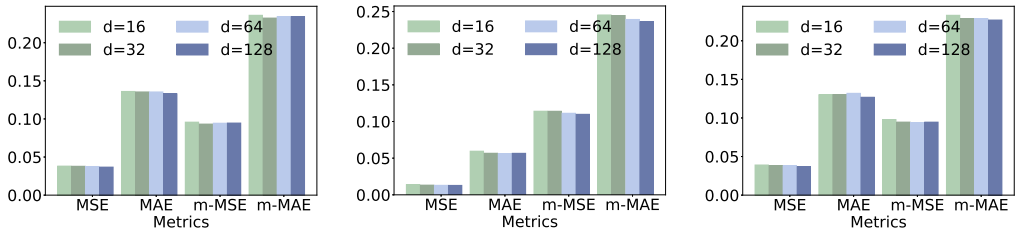
(b) Macro-MSE results on the five sets

Figure 6.6. Tune on the clip rate β.



(a) Results on the Sport dataset

(b) Results on the Movie dataset



(c) Results on the CD dataset

(d) Results on the Music dataset

(e) Results on the Beauty dataset

Figure 6.7. The influence of dimensionality d of the user/item representation on the DGT model. "m-MSE/MAE" denotes the bias-free metrics "Macro-MSE/MAE".

standard and unbiased metrics (i.e., MSE/MAE and Macro-MSE/MAE). However, for fairness in comparison to traditional sequential recommendation approaches which only have access to first-order neighbours of users/items, we utilize only one GNN layer in the proposed DGT model. Despite this limitation; we still achieve

Table 6.5. Results (in %) of ablation studies on the five datasets.

Models	Metric	Sport	Movie	CD	Music	Beauty
DGT-L1	MSE	4.22	4.51	3.76	1.3	3.8
	MAE	13.51	15.16	13.54	5.61	13.19
	Macro-MSE	13.4	8.52	9.44	11.13	9.4
	Macro-MAE	28.42	22.29	23.42	23.92	22.85
DGT-L2	MSE	**3.97**	**4.34**	**3.56**	**1.23**	**3.47**
	MAE	**13.24**	**14.93**	**13.04**	**5.35**	**12.11**
	Macro-MSE	**12.6**	**8.19**	**9.39**	**10.94**	**8.99**
	Macro-MAE	**27.34**	**21.87**	**23.23**	**23.35**	**21.85**

the highest performance among all state-of-the-art methods.

6.5 Conclusion

In this chapter, we propose the DGT and PS-DGT models for unbiased explicit feedback-based sequential recommendations. We extract disentangled embeddings of users and items from a temporal graph neural network to represent their preferences and dislikes. Such disentangled representation effectively avoids the dislike feature being overwhelmed by the preference features in their interaction records due to selection bias. Moreover, we decompose the propensity score into a user-rating propensity, item-rating propensity and user-item correlation and incorporate prior knowledge of user- and item-rating distribution for accurate estimation of the propensity score (PS). The PS is employed as weights in the loss functions and prior knowledge in sequential modelling to combat the bias in the user (and item) history. Extensive experiments on five real-world datasets demonstrate the superiority of and the effectiveness of the proposed methods.

Chapter 7

Conclusions and Future Directions

This chapter summarizes the main contributions of this dissertation and discusses possible future directions.

7.1 Conclusion

Firstly, we discussed a recently popular method for mitigating data sparsity issues in effective recommendation systems in Chapters 3 and 4, namely cross-domain recommendation. Specifically, In Chapter 3, we presented RecGURU, an adversarial learning-based cross-domain sequence recommendation model. Our approach is different from traditional methods which transfer knowledge or data from source to target domains. Instead, it extracts user representations from the target domain and fuses knowledge (user representation distribution) from the source domain through adversarial learning in order to generate global user representations. The global user representations are then applied to the target domain for boosted recommendations. This way, our model is applicable to non-overlapping users for cross-domain recommendations and increases its applicability in real-world systems where there are usually few overlapping users. We performed extensive experiments on public datasets and Tencent business data to validate the reliability and effectiveness of our model. In Chapter 4, we propose the CAT-ART model to extend the dual-target cross-domain recommendation (DTCDR) to multi-target cross-domain recommen-

dation (MTCDR) scenarios and achieve both One for All (OFA) and All for One (AFO) objectives. In the One for All objective, All domain-specific representations of a user are used for extracting One single global user representation. To do this, we propose a contrastive autoencoder (CAT) module which takes the user's pre-trained representations in all participant domains as input and outputs global user representations. With the contrastive autoencoder, we can extract unbiased representations of users without sharing original data. Additionally, in the All for One objective, all available features are used for recommendations in one target domain. For this purpose, an attention-based representation transfer (ART) unit is built into each target domain to incorporate domain-specific user representation from all source domains, thus avoiding negative transfer problems. We collected data from 5 domains in Tencent's real business scenarios including Article, App installation, App Usage, Long video watching and short video watching; experiments have been conducted to verify the effectiveness of the model design.

We further investigate the potential of deploying Graph Neural Networks (GNNs) for improved recommendations. In particular, sequential recommendation problems can be represented as an edge prediction task in a dynamic heterogeneous graph. Thus, Chapter 5 proposes CTRL, a dynamic heterogeneous graph representation learning model. CTRL employs an edge-based Hawks process to consider temporal influences of historical events on current node representations and take node centrality and semantic correlations into account in the local aggregation of each GNN layer. During training, apart from edge prediction, event (subgraph) prediction tasks are included to capture evolutionary patterns of higher-order topology structures. Experiments on three public datasets validate the effectiveness of CTRL.

In Chapter 6, we discussed the bias problem and presented debiasing techniques for recommendation systems. We proposed a Disentangled Graph Transformer (DGT) to tackle this issue in sequence recommendations based on explicit user feedback. Our model modelled user preferences and dislikes separately, avoiding data imbalance problems between the two. Moreover, we decomposed the interaction propensity score into three components, i.e., user-rating propensity, item-rating

propensity, and user-item correlation propensity, and applied prior knowledge of user/item rating distributions to estimate the score accurately. Additionally, instead of reweighing the loss for each training sample, the PS was also incorporated into GNN layers to calibrate biased historical interaction records for unbiased user representations. Finally, experiments on five public datasets revealed our method's superiority over other state-of-the-art solutions in traditional and unbiased evaluation metrics.

7.2 Future Directions

This thesis mainly focuses on collaborative filtering-based methods for various recommendation scenarios. However, side information about the users and items is also helpful for accurate user modelling and recommendations. Therefore, one promising direction is to further integrate side information into the current model for handling the data sparsity issue in a single domain. Moreover, side information is also semantically interoperable among different domains, therefore it can also be used as a bridge to fill the gap in various domains for cross-domain recommendations on non-overlapped users.

Unbiased evaluation of recommendation systems is crucial for mitigating various biases in recommendation systems. Current evaluation methods either rely on a significant amount of unbiased data, which negatively affects user experience, or resample biased data, which is often unreliable. New evaluators using large-scale biased data and small-size unbiased data are worth exploring. Additionally, more theoretical studies are needed to analyze the proposed evaluator's expectations, bounds, and confidence.

Furthermore, explainable recommendations are becoming increasingly important as they improve transparency, persuasiveness, effectiveness, trustworthiness, and satisfaction. Explainable recommendation and debiasing are related since they both address why an algorithm recommends certain items. A causal graph can be promising in addressing bias and providing explanations from strong causal paths.

To design a better causal graph capable of reasoning, debiasing, and explanation, we need to take the next step. We believe that a causal model will take recommendation research to a new frontier.

Bibliography

[1] J. J. Castro-Schez, R. Miguel, D. Vallejo, and L. M. López-López, "A highly adaptive recommender system based on fuzzy logic for b2c e-commerce portals," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2441–2454, 2011.

[2] Z. Huang, D. Zeng, and H. Chen, "A comparison of collaborative-filtering recommendation algorithms for e-commerce," *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 68–78, 2007.

[3] D. L. Chao, J. Balthrop, and S. Forrest, "Adaptive radio: achieving consensus using negative preferences," in *Proceedings of the 2005 ACM International Conference on Supporting Group Work*, 2005, pp. 120–123.

[4] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez, "Social knowledge-based recommender system. application to the movies domain," *Expert Systems with applications*, vol. 39, no. 12, pp. 10 990–11 000, 2012.

[5] P. Winoto and T. Y. Tang, "The role of user mood in movie recommendations," *Expert Systems with Applications*, vol. 37, no. 8, pp. 6086–6092, 2010.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[7] 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.

[8] M. D. Ekstrand, J. T. Riedl, J. A. Konstan *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[9] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," *The adaptive web: methods and strategies of web personalization*, pp. 325–341, 2007.

[10] 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.

[11] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining*. IEEE, 2010, pp. 995–1000.

[12] 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.

[13] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[14] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1161–1170.

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

[16] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 197–206.

[17] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, and Y. Li, "Sequential recommendation with graph neural networks," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 378–387.

[18] T. Man, H. Shen, X. Jin, and X. Cheng, "Cross-domain recommendation: An embedding and mapping approach." in *IJCAI*, vol. 17, 2017, pp. 2464–2470.

[19] F. Zhu, Y. Wang, C. Chen, J. Zhou, L. Li, and G. Liu, "Cross-domain recommendation: challenges, progress, and prospects," *arXiv preprint arXiv:2103.01696*, 2021.

[20] Q. Cui, T. Wei, Y. Zhang, and Q. Zhang, "Herograph: A heterogeneous graph framework for multi-target cross-domain recommendation." in *ORSUM@ RecSys*, 2020.

[21] H. Yan, C. Yang, D. Yu, Y. Li, D. Jin, and D. M. Chiu, "Multi-site user behavior modeling and its application in video recommendation," *IEEE Transactions on Knowledge & Data Engineering*, vol. 33, no. 01, pp. 180–193, 2021.

[22] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[23] 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.

[24] Z. Fan, Z. Liu, J. Zhang, Y. Xiong, L. Zheng, and P. S. Yu, "Continuous-time sequential recommendation with temporal graph collaborative transformer," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. ACM, 2021.

[25] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *Proceedings of the eleventh ACM conference on recommender systems*, 2017, pp. 161–169.

[26] J. Chen, H. Dong, Y. Qiu, X. He, X. Xin, L. Chen, G. Lin, and K. Yang, "Autodebias: Learning to debias for recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 21–30.

[27] Z. Wang, S. Shen, Z. Wang, B. Chen, X. Chen, and J.-R. Wen, "Unbiased sequential recommendation with latent confounders," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2195–2204.

[28] C. Xu, J. Xu, X. Chen, Z. Dong, and J.-R. Wen, "Dually enhanced propensity score estimation in sequential recommendation," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 2260–2269.

[29] C. Li, M. Zhao, H. Zhang, C. Yu, L. Cheng, G. Shu, B. Kong, and D. Niu, "Recguru: Adversarial learning of generalized user representations for cross-domain recommendation," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 571–581.

[30] C. Li, Y. Xie, C. Yu, B. Hu, G. Shu, X. Qie, D. Niu *et al.*, "One for all, all for one: Learning and transferring user embeddings for cross-domain recommendation," *arXiv preprint arXiv:2211.11964*, 2022.

[31] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, vol. 17. Melbourne, Australia, 2017, pp. 3203–3209.

[32] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 811–820.

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

[34] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1441–1450.

[35] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention network," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[36] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, "Filter-enhanced mlp is all you need for sequential recommendation," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2388–2399.

[37] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, "Next-item recommendation with sequential hypergraphs," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1101–1110.

[38] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 346–353.

[39] S. Zhang, D. Yao, Z. Zhao, T.-S. Chua, and F. Wu, "Causerec: Counterfactual user sequence synthesis for sequential recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 367–377.

[40] H. Chen, Y. Lin, M. Pan, L. Wang, C.-C. M. Yeh, X. Li, Y. Zheng, F. Wang, and H. Yang, "Denoising self-attentive sequential recommendation," in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 92–101.

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

[42] G. Hu, Y. Zhang, and Q. Yang, "Conet: Collaborative cross networks for cross-domain recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 667–676.

[43] C. Gao, X. Chen, F. Feng, K. Zhao, X. He, Y. Li, and D. Jin, "Cross-domain recommendation without sharing user-relevant data," in *The World Wide Web Conference*, 2019, pp. 491–502.

[44] C. Zhao, C. Li, R. Xiao, H. Deng, and A. Sun, "Catn: Cross-domain recommendation for cold-start users via aspect transfer network," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 229–238.

[45] A. Krishnan, M. Das, M. Bendre, H. Yang, and H. Sundaram, "Transfer learning via contextual invariants for one-to-many cross-domain recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1081–1090.

[46] F. Zhu, Y. Wang, C. Chen, G. Liu, M. Orgun, and J. Wu, "A deep framework for cross-domain and cross-system recommendations," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[47] S. Kang, J. Hwang, D. Lee, and H. Yu, "Semi-supervised learning for cross-domain recommendation to cold-start users," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1563–1572.

[48] P. Li and A. Tuzhilin, "Ddtcdr: Deep dual transfer cross domain recommendation," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 331–339.

[49] ——, "Dual metric learning for effective and efficient cross-domain recommendations," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[50] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[51] D. Perera and R. Zimmermann, "Cngan: Generative adversarial networks for cross-network user preference generation for non-overlapped users," in *The World Wide Web Conference*, 2019, pp. 3144–3150.

[52] C. Wang, M. Niepert, and H. Li, "Recsys-dan: discriminative adversarial networks for cross-domain recommender systems," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 2731–2740, 2019.

[53] H. Yan, P. Zhao, F. Zhuang, D. Wang, Y. Liu, and V. S. Sheng, "Cross-domain recommendation with adversarial examples," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 573–589.

[54] Y. Li, J.-J. Xu, P.-P. Zhao, J.-H. Fang, W. Chen, and L. Zhao, "Atlrec: An attentional adversarial transfer learning network for cross-domain recommendation," *Journal of Computer Science and Technology*, vol. 35, no. 4, pp. 794–808, 2020.

[55] M. Ma, P. Ren, Y. Lin, Z. Chen, J. Ma, and M. d. Rijke, "π-net: A parallel information-sharing network for shared-account cross-domain sequential recommendations," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 685–694.

[56] F.-Z. Zhuang, Y.-M. Zhou, H.-C. Ying, F.-Z. Zhang, X. Ao, X. Xie, Q. He, and H. Xiong, "Sequential recommendation via cross-domain novelty seeking trait mining," *Journal of Computer Science and Technology*, vol. 35, pp. 305–319, 2020.

[57] F. Yuan, X. He, A. Karatzoglou, and L. Zhang, "Parameter-efficient transfer from sequential behaviors for user modeling and recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1469–1478.

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

[59] H. Kanagawa, H. Kobayashi, N. Shimizu, Y. Tagami, and T. Suzuki, "Cross-domain recommendation via deep domain adaptation," in *European Conference on Information Retrieval*. Springer, 2019, pp. 20–29.

[60] F. Yuan, L. Yao, and B. Benatallah, "Darec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns," *arXiv preprint arXiv:1905.10760*, 2019.

[61] Y. Wang, C. Feng, C. Guo, Y. Chu, and J.-N. Hwang, "Solving the sparsity problem in recommendations via cross-domain item embedding based on co-clustering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 717–725.

[62] F. Zhu, C. Chen, Y. Wang, G. Liu, and X. Zheng, "Dtcdr: A framework for dual-target cross-domain recommendation," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1533–1542.

[63] M. Liu, J. Li, G. Li, and P. Pan, "Cross domain recommendation via bi-directional transfer graph collaborative filtering networks," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 885–894.

[64] F. Zhu, Y. Wang, C. Chen, G. Liu, and X. Zheng, "A graphical and attentional framework for dual-target cross-domain recommendation." in *IJCAI*, 2020, pp. 3001–3008.

[65] C. Li, M. Zhao, H. Zhang, C. Yu, L. Cheng, G. Shu, B. Kong, and D. Niu, "Recguru: Adversarial learning of generalized user representations for cross-domain recommendation," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 571–581. [Online]. Available: https://doi.org/10.1145/3488560.3498388

[66] Y. Zhu, Z. Tang, Y. Liu, F. Zhuang, R. Xie, X. Zhang, L. Lin, and Q. He, "Personalized transfer of user preferences for cross-domain recommendation," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 1507–1515.

[67] F. Zhuang, Y. Zhou, F. Zhang, X. Ao, X. Xie, and Q. He, "Cross-domain novelty seeking trait mining for sequential recommendation," *arXiv preprint arXiv:1803.01542*, 2018.

[68] J. Liu, P. Zhao, F. Zhuang, Y. Liu, V. S. Sheng, J. Xu, X. Zhou, and H. Xiong, "Exploiting aesthetic preference in deep cross networks for cross-domain recommendation," in *Proceedings of The Web Conference 2020*, 2020, pp. 2768–2774.

[69] A. K. Sahu and P. Dwivedi, "Knowledge transfer by domain-independent user latent factor for cross-domain recommender systems," *Future Generation Computer Systems*, vol. 108, pp. 320–333, 2020.

[70] D. Kim, S. Kim, H. Zhao, S. Li, R. A. Rossi, and E. Koh, "Domain switch-aware holistic recurrent neural network for modeling multi-domain user behavior," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 663–671.

[71] F. Yuan, G. Zhang, A. Karatzoglou, J. Jose, B. Kong, and Y. Li, "One person, one model, one world: Learning continual user representation without forgetting," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 696–705.

[72] X. Zhao, N. Yang, and P. S. Yu, "Multi-sparse-domain collaborative recommendation via enhanced comprehensive aspect preference learning," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 1452–1460.

[73] F. Zhu, Y. Wang, J. Zhou, C. Chen, L. Li, and G. Liu, "A unified framework for cross-domain and cross-system recommendations," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[74] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," *arXiv preprint arXiv:1206.5267*, 2012.

[75] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *International Conference on Machine Learning*. PMLR, 2014, pp. 1512–1520.

[76] H. Steck, "Evaluation of recommendations: rating-prediction and ranking," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 213–220.

[77] A. Collins, D. Tkaczyk, A. Aizawa, and J. Beel, "A study of position bias in digital library recommender systems," *arXiv preprint arXiv:1802.06565*, 2018.

[78] Y. Zheng, C. Gao, X. Li, X. He, Y. Li, and D. Jin, "Disentangling user interest and popularity bias for recommendation with causal embedding," *arXiv preprint arXiv:2006.11011*, p. 64, 2020.

[79] H. Abdollahpouri and M. Mansoury, "Multi-sided exposure bias in recommendation," *arXiv preprint arXiv:2006.15772*, 2020.

[80] J. Chen, C. Wang, S. Zhou, Q. Shi, J. Chen, Y. Feng, and C. Chen, "Fast adaptively weighted matrix factorization for recommendation with implicit feedback," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3470–3477.

[81] J. Chen, C. Wang, S. Zhou, Q. Shi, Y. Feng, and C. Chen, "Samwalker: Social recommendation with informative sampling strategy," in *The World Wide Web Conference*, 2019, pp. 228–239.

[82] D. Liu, P. Cheng, Z. Dong, X. He, W. Pan, and Z. Ming, "A general knowledge distillation framework for counterfactual recommendation via uniform data," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 831–840.

[83] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, "Unbiased recommender learning from missing-not-at-random implicit feedback," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 501–509.

[84] G. Lederrey and R. West, "When sheep shop: measuring herding effects in product ratings with natural experiments," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 793–802.

[85] P. Wu, H. Li, Y. Deng, W. Hu, Q. Dai, Z. Dong, J. Sun, R. Zhang, and X.-H. Zhou, "On the opportunity of causal learning in recommendation systems: Foundation, estimation, prediction and challenges," in *Proceedings of*

the International Joint Conference on Artificial Intelligence, Vienna, Austria, 2022, pp. 23–29.

[86] A. Schuth, H. Oosterhuis, S. Whiteson, and M. de Rijke, "Multileave gradient descent for fast online learning to rank," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 457–466.

[87] Z. Zhu, Y. He, Y. Zhang, and J. Caverlee, "Unbiased implicit recommendation and propensity estimation via combinational joint learning," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 551–556.

[88] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims, "Recommendations as treatments: Debiasing learning and evaluation," in *international conference on machine learning*. PMLR, 2016, pp. 1670–1679.

[89] 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*. PMLR, 2019, pp. 6638–6647.

[90] W. Wang, F. Feng, X. He, H. Zhang, and T.-S. Chua, "Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1288–1297.

[91] T. Wei, F. Feng, J. Chen, Z. Wu, J. Yi, and X. He, "Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1791–1800.

[92] Y. Zhang, F. Feng, X. He, T. Wei, C. Song, G. Ling, and Y. Zhang, "Causal intervention for leveraging popularity bias in recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 11–20.

[93] 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*, 2021, pp. 2980–2991.

[94] S. Ding, F. Feng, X. He, J. Jin, W. Wang, Y. Liao, and Y. Zhang, "Interpolative distillation for unifying biased and debiased recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 40–49.

[95] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[96] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

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

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

[99] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[100] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018, accepted as poster. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[101] Y. Sun and J. Han, "Mining heterogeneous information networks: principles and methodologies," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 1–159, 2012.

[102] 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.

[103] 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.

[104] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 793–803.

[105] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 2020, pp. 2704–2710. [Online]. Available: https://doi.org/10.1145/3366423.3380027

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

[107] Z. Liu, C. Huang, Y. Yu, and J. Dong, "Motif-preserving dynamic attributed network embedding," in *Proceedings of the Web Conference 2021*, 2021, pp. 1629–1638.

[108] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5363–5370.

[109] D. Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan, "Inductive representation learning on temporal graphs," in *International Conference on Learning Representations (ICLR)*, 2020.

[110] Z. Wen and Y. Fang, "Trend: Temporal event and node dynamics for graph representation learning," in *Proceedings of the Web Conference 2022*, 2022.

[111] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=KYPz4YsCPj

[112] L. Sun, Z. Zhang, J. Zhang, F. Wang, H. Peng, S. Su, and S. Y. Philip, "Hyperbolic variational graph neural network for modeling dynamic graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4375–4383.

[113] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[114] X. Wang, D. Lyu, M. Li, Y. Xia, Q. Yang, X. Wang, X. Wang, P. Cui, Y. Yang, B. Sun *et al.*, "Apan: Asynchronous propagation attention network for real-time temporal graph embedding," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2628–2638.

[115] H. Zhou, D. Zheng, I. Nisa, V. Ioannidis, X. Song, and G. Karypis, "Tgl: A general framework for temporal gnn training on billion-scale graphs," *arXiv preprint arXiv:2203.14883*, 2022.

[116] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[117] Y. Yin, L.-X. Ji, J.-P. Zhang, and Y.-L. Pei, "Dhne: Network representation learning method for dynamic heterogeneous networks," *IEEE Access*, vol. 7, pp. 134 782–134 792, 2019.

[118] R. Bian, Y. S. Koh, G. Dobbie, and A. Divoli, "Network embedding and change modeling in dynamic heterogeneous networks," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 861–864.

[119] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou, "Dynamic heterogeneous information network embedding with meta-path based proximity," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[120] Q. Li, Y. Shang, X. Qiao, and W. Dai, "Heterogeneous dynamic graph attention network," in *2020 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2020, pp. 404–411.

[121] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[122] H. Huang, R. Shi, W. Zhou, X. Wang, H. Jin, and X. Fu, "Temporal heterogeneous information network embedding." in *IJCAI*, 2021, pp. 1470–1476.

[123] Y. Ji, T. Jia, Y. Fang, and C. Shi, "Dynamic heterogeneous graph embedding via heterogeneous hawkes process," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 388–403.

[124] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2857–2866.

[125] Y. Lu, X. Wang, C. Shi, P. S. Yu, and Y. Ye, "Temporal network embedding with micro-and macro-dynamics," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 469–478.

[126] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning hierarchical representation model for nextbasket recommendation," in *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2015, pp. 403–412.

[127] M. Xu, F. Liu, and W. Xu, "A survey on sequential recommendation," in *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, 2019, pp. 106–111.

[128] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv preprint arXiv:1901.07291*, 2019.

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

[130] A. Makhzani and B. Frey, "K-sparse autoencoders," *arXiv preprint arXiv:1312.5663*, 2013.

[131] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[132] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

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

[134] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[135] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[136] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 650–658.

[137] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[138] W. Zhang, L. Deng, L. Zhang, and D. Wu, "Overcoming negative transfer: A survey," *arXiv preprint arXiv:2009.00909*, 2020.

[139] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[140] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[141] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2021.

[142] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.

[143] 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*, 2016, pp. 855–864.

[144] P. Adam, G. Sam, C. Soumith, and C. Gregory, *Pytorch:Tensors and dynamic neural networks in python with strong gpu acceleration.*, 2017, https://pytorch.org/.

[145] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[146] 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.

[147] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1726–1736.

[148] C. D. Barros, M. R. Mendonça, A. B. Vieira, and A. Ziviani, "A survey on embedding dynamic graphs," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–37, 2021.

[149] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[150] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.

[151] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[152] L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, and A. Zhang, "A survey on causal inference," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 5, pp. 1–46, 2021.