EXPLOITING CROSS-CHANNEL INFORMATION FOR PERSONALIZED RECOMMENDATION

XIANG WANG
(Bachelor of Computer Science and Engineering, Beihang University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2019

Supervisor:
Professor Tat-Seng Chua

Examiners:
Professor Janice Lee Mong Li
Associate Professor Roland Yap Hock Chuan
Professor Maarten de Rijke, University of Amsterdam

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

XIANG WANG

1 January 2019

ACKNOWLEDGEMENTS

First and foremost, I want to thank my supervisor, Prof Tat-Seng Chua for his support and guidance all the way. He has promoted my critical thinking and analysis, and taught me how to do research professionally. I appreciate all his contributions of ideas, time, and consideration throughout all four years. It has been my honor to be his PhD student.

I would like to thank my thesis committee members: Prof Lee Mong Li, Prof Roland Yap, and Prof Maarten de Rijke for their time and valuable comments on my work.

The members of the NExT center has been a great source of precious friendship and collaboration. I am grateful to my mentors: Dr Liqiang Nie and Dr Xiangnan He for their useful assistance and guidance. I have benefited immensely from the collaboration with them and appreciated their enthusiasm for research. I also thank brilliant group members, including Fuli Feng, Choon Meng Seah, Jingyuan Chen, Jingjing Chen, Lizi Liao, Yunshan Ma, Na Zhao, Xindi Shang, Gelli Francesco, Xiaoyu Du, Meng Liu, Jun He, Dr Xun Yang, and Dr Yixin Cao, and countless others. It is my honor to be friends of them and collaborate with so brilliant researchers. My time at NUS is colorful and enjoyable due to the accompany of my friends.

Lastly, I would like to sincerely thank my family, especially my parents Junzhen Gao and Zhanhong Wang, and my girlfriend An Zhang, for all their love, support, patience and encouragement. Thank you.

CONTENTS

ABSTRACT

Personalized recommendation is ubiquitous, having been applied to many online services such as E-commerce and advertise. It facilitates users to discover a small set of relevant items, which meet their personalized interests, from a large number of choices. Generally, the modeling of user-item interactions is at the heart of personalized recommendation. Such modeling aims to estimate the relevance score of a user for a given item based on historical user-item interactions, accounting for the user's preferences towards the target item. Among the diverse strategies, collaborative filtering has achieved great success, which uses the IDs of a user and an item solely, without utilizing any domain knowledge or side information of users and items. Despite its simplicity and effectiveness, considering only IDs without utilizing the rich side information severely limits the ability of the model to capture user-item relationships precisely and offer the necessary reasoning.

Nowadays, diverse kinds of auxiliary information on users and items become increasingly available in online platforms, such as user demographics, social relations, and item knowledge. More recent evidence suggests that incorporating such auxiliary data with collaborative filtering can better capture the underlying and complex user-item relationships, and further achieve higher recommendation quality. Furthermore, such auxiliary data, such as social relations and background knowledge on items, enable us to uncover valuable evidence as well as reasons on why a recommendation is made.

In this thesis, we explore auxiliary information across different channels to enrich the representations of users and items in performing better recommendation. As defined in cross-channel marketing, a channel refers to the platform that a business can use to reach potential customers, including online platforms and offline retail stores. Different channels emphasize different aspects or behaviors of users. From the viewpoint of recommendation, we define the channel as one domain or source which can indicate users' preferences towards an item. Consequently, cross-channel means incorporating multiple sources of information in establishing the whole view of user-item interactions.

First, we specifically focused on leveraging user behaviors across online and offline channels for event recommendation, utilizing connections between information- and social-oriented channels for cross-domain recommendation, and exploring cross features incorporating the user-centric and item-centric channels. Particularly, for event recommendation, we unified users' online and offline behaviors into user representations, to better depict user's preferences for target events.

Second, we investigated a novel task of cross-channel social recommendation, which aims to recommend relevant items of information channel to potential users within the social channels. We integrated user-item interactions from an information platform and user social relations from a social platform. Wherein, bridge users are used to unify the signals from these two heterogeneous channels

to detect potential customers in the social platform.

While integrating such side information is able to offer strong predictive performance, important questions are being raised on why the recommended items are suitable for the user. Towards this end, we worked on the explainable recommendation task, where top cross features ranked by attention weights can be treated as concrete reasons for a recommendation. Here cross features incorporate the basic features derived from user-centric and item-centric channels, representing the general decision rules, which unveil possible user intents behind a behavior.

To date, incorporating knowledge-aware channels, especially knowledge graphs, into recommender systems is attracting increasing interests, since it can provide deep factual knowledge and rich semantics on items. The usage of such knowledge endows recommender systems with strong representation ability and explainbility. We contribute a new recurrent network to exploit knowledge graph for recommendation, which can generate path representations by composing the semantics of both entities and relations. As such, it is capable of making more accurate recommendations results with knowledge-aware explanations, exhibiting the process of information propagation.

In this thesis, we investigate the role of cross-channel auxiliary information in various personalized recommendation scenarios. One key finding is that information from different channels, such as online and offline behaviors, contains rich cues about users' preferences, endowing recommender systems with strong expressiveness. Meanwhile, the features across different channels can provide valuable reasons behind a recommendation. Extensive experiments are performed on real-world datasets to demonstrate our proposed methods.

LIST OF TABLES

LIST OF FIGURES

LIST OF PUBLICATIONS

The work in this thesis has been published in the following venues:

- Xiang Wang, Liqiang Nie, Xuemeng Song, Dongxiang Zhang, and Tat-Seng Chua. *Unifying Virtual and Physical Worlds: Learning Toward Local and Global Consistency.* ACM Trans. Inf. Syst. (TOIS'17), volume 36 pages 4:1–4:26.
- Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. *Item Silk Road: Recommending Items from Information Domains to Social Users.* In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17) pages 185–194, Shinjuku, Tokyo, Japan, August 7-11, 2017
- Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. *TEM: Tree-enhanced Embedding Model for Explainable Recommendation.* In Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW'18), pages 1543–1552, Lyon, France, April 23-27, 2018
- Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. *Explainable Reasoning over Knowledge Graphs for Recommendation.* In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), Hilton Hawaiian Village, Honolulu, Hawaii, USA, January 27 February 1, 2019

Note that I am the lead author of all of these works.

Chapter 1

Introduction

1.1 Background

With the explosive growth of information, users are overwhelmed by a large number of choices available to them[1]. Recommender system (RS) serves as the generic solution to overcome such information overload, aiming to find a small set of relevant items from the overwhelming online content and services (*e.g.,* movies, news, and products) to meet users' personal interests. As such, it has been gaining widespread adoption across industry, driving customer-oriented services in various online platforms like Amazon, YouTube, and Facebook. Regardless of platform, personalized recommendation plays a pivotal role to boost business and facilitate decision-making process. For instance, Netflix has millions of users worldwide and over 80% of TV shows these users watch are discovered through the platform's recommendation systems[2]; and Amazon, which has a total of over 398 million products, utilizes its recommendation system as a targeted marketing took, generating 35% percent of its revenue[3].

Technically speaking, the recommendation problem is usually tackled as a matching problem, which aims to estimate the relevance score of a user for a given item based on the historical user-item interactions, as well as the profiles for users and items. The user-item interactions are usually available in the form of explicit feedback like reviews and ratings, along with various kinds of implicit feedback like views and clicks [11]. Regardless of the application domain, a

[1] `https://www.economist.com/business/2011/06/30/too-much-information`.
[2] `https://goo.gl/ToJ4qt`.
[3] `https://goo.gl/s9BaUK`.

user's profile usually consists of an ID (to identify which specific user) and some side information such as age, gender, and hometown. Similarly, an item's profile typically contains an ID and some attributes like categories, tags, and price [165]. Collaborative modeling of user-item interactions and their profiles, hence, is at the heart of personalized recommendation, accounting for the user preferences on items.

Towards this end, several groups of recommendation strategies are proposed: collaborative filtering (CF) methods [43, 81, 67, 165], content-based methods [10, 79], and hybrid methods [87, 39]. Among various strategies, CF has achieved great success owing to its excellence in user-item interaction modeling and simplicity. It explores user preferences towards items by simply assuming that similar users would have similar preferences towards certain items [138, 95, 165]. Technical systems use only the ID of a user and an item in matching, while other side information are overlooked. Particularly, matrix factorization (MF) [81, 69], one of the simplest yet effective CF model, characterizes a user ID or an item ID with a latent vector, and models a user-item interaction as the inner product of their latent vectors. Regardless of the recommendation scenarios, CF serves as a generic solution for recommendation without requiring any domain knowledge. However, we note that MF's expressiveness can be limited by the use of only ID information to model a user-item interaction. Particularly, the extremely large item set, such as millions of products in Amazon and Taobao, easily leads to sparse user-item interactions. Consequently, the sparsity issue greatly hinders the expressiveness of CF [178]. In addition, CF methods usually suffer from the cold-start issues, namely, they fail to recommend new items (*cf.* cold-start items) or new users (*cf.* cold-start users) since no interactions are used to learn their latent vectors.

In addition to IDs, users (or items) are always associated with abundant side information, such as user demographics and item attributes, which may offer relevance signals of user preferences on items. Therefore, many CF variants have been proposed to incorporate auxiliary information for tackling these issues, such as factorization machine (FM) [130], Neural FM (NFM) [65], Wide&Deep [31], and Deep Crossing [139]. Various techniques, such as the bilinear interaction pooling in NFM, are employed on different types of auxiliary data to uncover the complex user-item relationships. Take FM as an

example, it feeds the categorical side information with IDs as the input features, and utilizes addition and inner product operations to capture the linear and pairwise interactions between features. The way of combining ID and auxiliary information enriches the representations of users and items, and enhances their interactions. Moreover, recent deep learning based recommender systems are gaining significant attention by overcoming the expressiveness limitation of conventional models and capturing the non-linear and higher-order user-item interactions within the auxiliary data [181].

1.2 Motivation and Challenge

Nowadays, diverse kinds of auxiliary information on users and items are increasing available in online platforms, offering valuable information and accounting for predicting the user preferences on items. In general, each platforms is carefully designed for one main purpose, such as E-commerce, social networking, or social content sharing. Hence, the information within a platform can reflect one specific aspect of users or items. For instance, most E-commerce websites like Amazon allow users to post comments to express their opinions on the items, which contain textual words that reveal items' specific aspects with users' personal sentiments (*e.g.,* "I am loving iPhone X due to the new screen and face recognition sensors"). Apparently, the information involved in one platform consists of multiple types of entities (*e.g.,* users and items) and links (*e.g.,* user-user social relations and item taxonomy). Such heterogeneous data can be used by recommender systems to construct more accurate and comprehensive profiles for a user and an item.

We denote such a data source as a **channel**, which indicates user preferences on a target item. As we can see, different channels usually highlight specific aspects of items or behaviors of users. Here we focus on **cross-channel recommendation** to integrate information from different sources, reveal latent relationships among users and items, and further capture user preferences on items. We classify the heterogeneous information into various types of channels in different recommendation scenarios, including online and offline channels, and user-centric and item centric channels.

Event-based social networks offer a motivating example for categorizing user

behaviors into **online and offline channels**, where a user finds like-minded friends and forms interest groups online, and regularly attends offline events; this facilitates the exploration of the co-existence and relations of online and offline behaviors. E-commerce is an instance of categorizing auxiliary information into **user-centric and item-centric channels**, where users are usually associated with demographics (*e.g.,* age, gender, hometown, and income level), while items are assigned with rich attributes (*e.g.,* categories, tags, and price), reflecting some statistical patterns or rules (*i.e.,* feature crossing) to explain why a user tends to choose the target item. We will elaborate such cross-channel scenarios in Chapter 3 and Chapter 5, respectively.

As a promising direction, aggregating users' footprints on multiple platforms, covering more diverse knowledge on users and items, has been attracting increasing attention to industry and academia. Taking the word-of-mouth marketing as an example, it is not unusual for a user to share her travel experiences in TripAdvisor, and if she also holds a Facebook account, her experience in TripAdvisor can be recommended to her friends in Facebook. Here the users, who are simultaneously involved in both social network sites and information platforms, act as a bridge to propagate the heterogeneous knowledge across platforms. Regardless of application domains, we can roughly divide the information from different platforms into two groups: **information-oriented and social-oriented channels** [165], as Figure 1.1 shows. The former typically refers to forums or E-commerce platforms that present items, such as the point-of-interests in TripAdvisor, movies in IMDb, and products in Amazon etc. In information channel, ample user-item interactions are available in the form of explicit feedback (*e.g.,* reviews and ratings) and implicit feedback (*e.g.,* views and clicks) [11]. On the other hand, the social-oriented channel is mainly social networking services, which emphasize the social connections among users (*e.g.,* friendships, following, and follower). As we can see, the information channel essentially highlights item-centric data which is widely used in conventional recommendation scenarios, whereas the social channel emphasizes the user-centric data.

More recent evidence illustrates that incorporating auxiliary data with CF is driving a remarkable revolution in recommender systems. In this thesis, we will investigate the significance of cross-channel information in different

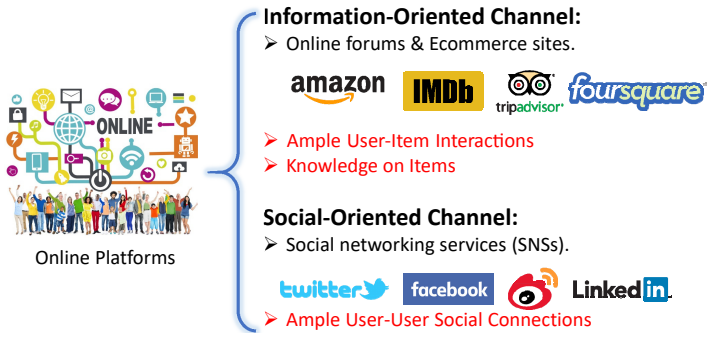


Figure 1.1: Illustration of the information-oriented and social-oriented channels.

real-world recommendation scenarios. Another noticeable opportunity lies in the **reasoning for recommendation** that explains why the target item is suitable for a user. Most state-of-the-art CF methods, however, work as a black box, where the decision process of a recommendation cannot be explicitly presented. We note that the rich factual and knowledge-aware information, including the social relations and background knowledge on items, can provide valuable evidence on why a recommendation is made to a user, bringing new opportunities in explainable recommendation. Providing concrete explanations or reasons allows users to make more accurate decisions, thereby improving their satisfaction. Towards this end, we will also explore the explainable recommendation task.

1.3 Contributions of the Thesis

This thesis explores several real-world applications which leverage different cross-channel informations to enhance the recommendation performance. They are summarized as follows.

1.3.1 Event Recommendation across Online and Offline Channels

Event recommendation has a wide range of applications within event-based services, such as event attendance prediction and event organization. Such event-based platforms (*e.g.,* Meetup and Facebook Event) not only help users find like-minded event attendees online, but also facilitate them to regularly attend offline events in some physical locations. Precise event recommendation can offer tremendous help to event organizers to predict who will attend the event, and how many users will be present eventually. To solve this task,

the primary strategy of prior efforts is to cluster users into different groups based on their online and offline interactions separately, and then conduct the recommendation [98, 54]. However, such simple clustering fails to fully explore the co-existence and relations of online and offline behaviors. In fact, user behaviors online and offline are inseparable and mutually reinforced [62, 114]. Hence, how to appropriately model the online and offline information, while preserving the structure coherence of users' social circles, is at the core of event recommendation.

In this work, we categorize user behaviors of one event-based platform into online and offline channels, which typically represent the online interest groups and offline attendance records, respectively. To explore the intrinsic relations across channels, we propose a dual-clustering method to group potential users into groups based on online and offline behaviors, where the global and local consistency constraints on users' social circles are adopted to encourage information propagation across channels. Having established the user representations over different groups, we use a simple model to integrate the group and personalized recommendation factors for predicting event attendance.

1.3.2 Social Recommendation across Information and Social Channels

As mentioned in Section 1.2, existing online platforms can be roughly categorized into information-oriented and social-oriented channels. When deciding to adopt an item, besides directly consulting the information sites (*e.g.,* TripAdvisor and Amazon) to obtain the factual knowledge of target items (*e.g.,* battery and price), a user usually gathers more detailed information from her experienced friends. This is referred to as the word-of-mouth marketing, which is widely recognized as an effective strategy for obtaining recommendation [165]. Hence, it is crucial to encourage the word-of-mouth marketing in the social channel. However, such channel is designed mainly for users to rebuild social connections, instead of seeking opinions regarding items. Though some item cues implying users' purchase intents many be mentioned in the channel, they typically contain only item names, with little detail about their comments or ratings. The sparse and weak user-item connections greatly degrade the ability to promote sales

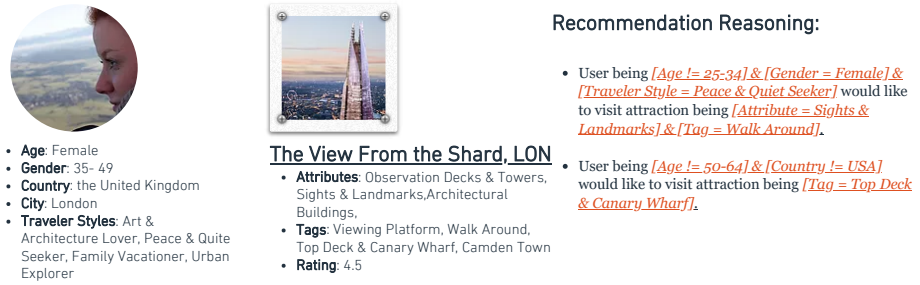


Figure 1.2: Illustration of the explainable recommendation.

and services in the social channel. We hence resort to information channel for borrowing the user-item interactions, where some users simultaneously involved in both channels can act as a bridge to propagate the desired knowledge. Nevertheless, it is highly challenging to leverage the history and behavior of a small number of bridge users to provide quality recommendation for non-bridge users in social channel.

To tackle these issues, we propose a novel solution named neural social collaborative ranking. For the modeling of information channel, we develop an attribute-aware recommender system based on neural CF framework, which leverages the attributes of users and items to enrich their representations, including the bridge users. For the modeling of social channel, the representations of bridge users are utilized to guide the representation learning of social users, where the smoothness and fitting constraints are employed. As such, we can estimate the relevance score of a non-bridge user from social channel for an item from information channel.

1.3.3 Explainable Recommendation using User and Item Channels

We note that the rich side information affiliated with user-item interactions can provide valuable evidence on why a recommendation is suitable for a user, as shown in Figure 1.2. However, such reasoning over auxiliary information has not been fully explored in offering explanations. On the technical side, the trade-off of accuracy and explainability is one fundamental problem in recommender systems. Most embedding-based methods, such as Wide&Deep and NFM, achieve the state-of-the-art performance, taking advantages of hidden and non-linear layers. However, they are considered as black box models, since their complex architecture obfuscates their internal decision processes. At the

other end of the spectrum, tree-based methods like decision trees are considered as explainable models, since they predict by inferring decision rules from data. Such decision rules, composed of feature combinations or feature crossing, are easy to comprehend and explain.

In this work, we propose a tree-enhanced embedding method that combines the strengths of the embedding-based and tree-based methods. We first employ a tree-based model to learn explicit decision rules that are the feature combinations across the user-centric and item-centric channels. Thereafter, we feed the cross features into an embedding-based model, incorporating a carefully designed attention network. The cross features with tailored attention weights are treated as the concrete reasons for a recommendation.

1.3.4 Knowledge-aware Recommendation

Recently, leveraging external knowledge like knowledge graph, has obtained increasing interests. This is because it can connect different topic domains with the shared items, boost recommendation performance, and introduce other benefits like reasoning and explainability. A knowledge graph is a type of heterogeneous graph, in which nodes correspond to entities and edges represent relations, showing how the content associated with users and items are interlinked to each other. Technically, embedding-based methods typically use knowledge graph embedding methods to support the representation learning of items, rather than characterizing user-item relations. On the other hand, meta path-based methods infer the user preferences via explicit meta-paths in the graph. However, it relies heavily on manually crafted meta-paths, requiring extensive domain knowledge.

Towards achieving knowledge-aware recommendation, we contribute a recurrent network to exploit knowledge graph for recommendation, which can generate path representations by composing the semantics of both entities and relations. By leveraging the sequential dependencies within a path, we allow effective reasoning on paths to infer the underlying rationale of a user-item interaction. Moreover, we design a new weighted pooling operation to discriminate the strengths of different paths in connecting a user with an item, endowing our model with a certain level of explainability. As such, we can provide more accurate recommendation with knowledge-aware explanations,

thus better analyzing the user behaviors and assisting users in making decisions.

In this thesis, we explore how to incorporate information from different channels into recommender systems and conduct extensive experiments to verify the effectiveness of the proposed methods. Most previous efforts only measure similarity based on auxiliary data, but neither specifies the characteristics of different channels nor considers the relations between channels. To bridge the research gap, we identify the importance of each channel, integrate their useful cues to infer users' preference, and then leverage cross features with attentive weights as the reasons behind a recommendation.

Chapter 2

Literature Review

In this chapter, we review prior efforts on personalized recommendation, which has been attracting attention in recent decades. We can roughly divide existing recommender systems into three groups: collaborative filtering (CF) methods, content-based methods, and hybrid methods. After introducing these recommendation paradigms, we illustrate the approaches exploiting different auxiliary information, which are most related with our work.

2.1 Personalized Recommendation

Personalized recommendation estimates users' preferences on items based on the historical user-item interactions, as well as the profiles of users and items. There generally exists two types of recommendation tasks based on the form of outputs: rating prediction and top-N recommendation. Rating prediction predicts the explicit feedback from users, such as one to five stars, which is usually tackled as a regression problem; whereas, top-N recommendation produces a ranked list with n items to each user [181], which casts as a ranking problem.

From the view of matching, the core of recommender systems is to estimate the relevance score of a user for a target item. The recommendation strategies are usually classified into three categories [3]: collaborative filtering (CF), content-based, and hybrid methods. As the focus of this thesis is on the CF paradigm, we emphasize CF approaches first and review prior efforts on the other paradigms briefly in this chapter.

2.1.1 Collaborative Filtering Methods

CF explores user preferences on target items by learning from the user-item historical interactions, either explicit feedback (*e.g.,* reviews and ratings) or implicit feedback (*e.g.,* browsing and click). The underlying assumption of CF is that similar users would have similar preferences towards items. Generally, CF approaches are commonly classified into memory-based and model-based categories.

Memory-based CF approaches [95, 3, 45] drive the user and item similarity matrix directly from the user-item interaction records, and consequently produce an estimation between a user and the target item. Obviously, similarity estimation is at the core of these methods, where many types of similarity measures are adopted such as cosine similarity and Pearson correlation coefficient. While easy to comprehend, memory-based methods unrealistically measure the similarities when the size of items is extremely large and the interaction data is sparse. The heuristic measurements lack tailored optimization for recommendation, leaving the higher-order and non-linear relations within interaction data untouched. Moreover, the obtained similarity is purely driven from the available user-item interactions, yet fail to generalize to the unseen items (*e.g.,* cold-start issues). Such limitations adversely hinder the generalization ability and expressiveness of these methods.

Model-based CF approaches usually tackle the shortcomings of memory-based methods and endow model strong representation and generalization ability, for which the holistic goal is to build:

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

where f denotes the underlying model with parameters Θ, and $\widehat{y}_{ui}$ denotes the predicted score for the user-item interaction y_{ui} between user u and item i. Along with this direction, latent factor models have garnered much attention and are becoming increasingly prevalent. Matrix factorization (MF) is one of the simplest yet effective latent factor models. It characterizes a user ID or an item ID with a latent vector (*aka.* embedding), modeling the user explicit

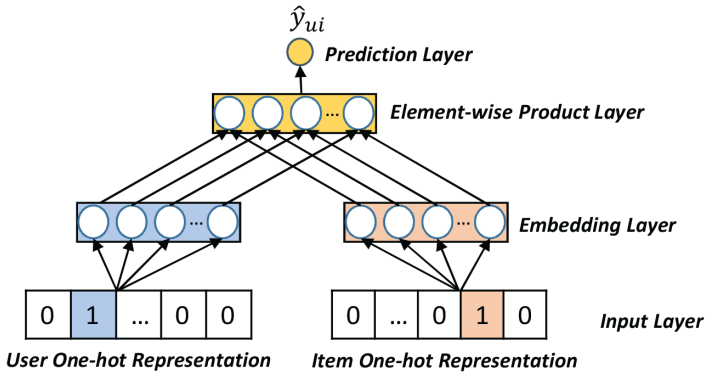


Figure 2.1: MF as a shallow neural network model.

feedback like ratings as the inner product of their latent vectors:

$$f_{MF}(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^\top \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}, \tag{2.2}$$

where $\mathbf{p}_u \in \mathbb{R}^K$ and $\mathbf{q}_i \in \mathbb{R}^K$ are model parameters denoting the latent vector (*aka.* representation) for user u and item i, respectively. Then its optimization is to, for each user-item pair, minimize the error between the observed rating and the model prediction, casting recommendation as the regression task.

Despite its effectiveness, we note that MF's expressiveness can be limited by the use of the inner product operation to model a user-item interaction. To illustrate this, we present a neural network view of the MF model. As shown in Figure 2.1, we feed the one-hot representation of user/item ID into the architecture, and project them with a fully connected embedding layer. By feeding the user/item embedding vectors into the element-wise product layer, we obtain a hidden vector $\mathbf{h} = \{p_{uk} q_{ik}\}$. If we directly project $\mathbf{h}$ into the output score, we can exactly recover the MF model. As such, MF can be deemed as a shallow neural network with one hidden layer only. Based on this connection, we argue that there are two key limitations of MF-based approaches for cross-domain social recommendation:

- First, MF only considers the simple two-way interaction between a user and an item, by assuming that their cross latent factors (*i.e.,* $\mathbf{p}_u$ and $\mathbf{q}_i$) are independent of each other. However, such an independence assumption can be insufficient to model real-world data, which usually have complex and non-linear underlying structures [67, 93].

- The case can be even worse if we take the attributes into account. A typical

way to extend MF with side attributes is SVDfeature, *i.e.,* by summing attribute embedding vectors with user/item embedding vector. As a result, the rich correlations among users, items, and attributes are ignored.

Inspired by the simplicity of MF, many variants have been proposed, such as SVD++ [80], Factorization Machine (FM) [130], Localized MF [184], Social MF [187]. FM has achieved great success as it can model the feature interactions between arbitrary number of entities, whereas MF only models the interactions between a user and an item. Each feature is associated with an embedding vector, and then the second-order correlation between any two features are captured by their pairwise inner products, as well as the higher-order feature interactions. Since most of these information are categorical variables, they are usually converted to real-valued feature vector via one-hot encoding [130, 65]. Let $\mathbf{x}_u$ and $\mathbf{x}_i$ denote the feature vector for user u and item i, respectively. To predict y_{ui}, a typical solution is to concatenate $\mathbf{x}_u$ and $\mathbf{x}_i$, *i.e.,* $\mathbf{x} = [\mathbf{x}_u, \mathbf{x}_i] \in \mathbb{R}^n$, which is then fed into a predictive model. FM [130, 11] is a representative of such predictive models, which is formulated as:

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{t=1}^{n} w_t x_t + \sum_{t=1}^{n} \sum_{j=t+1}^{n} \mathbf{v}_t^\top \mathbf{v}_j \cdot x_t x_j, \tag{2.3}$$

where w_0 and w_t are bias terms, $\mathbf{v}_t \in \mathbb{R}^k$ and $\mathbf{v}_j \in \mathbb{R}^k$ denote the embedding for feature t and j, respectively. We can see that FM associates each feature with an embedding, modeling the interaction of every two (nonzero) features via the inner product of their embeddings. If only user ID and item ID are used as the features of $\mathbf{x}$, FM can exactly recover the MF model; by feeding IDs and side features together into $\mathbf{x}$, FM models all pairwise (*i.e.,* second-order) interactions among IDs and side features. In the light of this, FM is a natural tool to integrate the auxiliary information with user-item interactions. However, owing to such simple linear operations like inner products, these MF variants only capture linear relationships, failing to uncover the nonlinear and complex relationships within the data.

Recently, many researchers resort to deep learning for enhancing representation ability and further recommendation quality. This is because that deep learning can extract more complex abstractions as data representations

and effectively capture the nonlinear user-item relations. Various types of deep learning techniques are involved in prior efforts, spanning from multi-layer perceptron (MLP) and autoencoder (AE) [92] to convolutional neural network (CNN) and recurrent neural network (RNN). We classify the existing studies based on the usage of deep learning techniques: enrich the representations of users and items from the auxiliary data like images and texts [178, 162], and enhance the feature interaction functions [31, 67]. Wide&Deep has been proposed in [31], where the wide component learns some explicit feature dependencies and the deep component employs a MLP on the concatenation of feature embedding vectors for uncovering the implicit feature interactions. Deep Crossing [139] stacks a MLP and multiple residual units over the embeddings of individual features to automatically generate the combinatorial features. Neural CF (NCF) [67] utilizes the feed-forward neural networks to replace the inner products of MF and parameterize the user-item interaction function, which serves as the generic deep recommendation solution. Recently, Neural FM (NFM) [65], as an extension of FM, proposes a bilinear interaction pooling operation to adopt element-wise products over the embedding vectors of users and items, and subsequently stacks MLP to capture the non-linear relationship between users and items. Similar to the spirit in Wide&Deep, DeepFM [61] seamlessly integrates FM and MLP components which capture linear and non-linear feature interactions, respectively.

Despite the excellence in modeling user-item interactions, these CF approaches have several limitations. First, conventional CF methods only take ID information of users and items into consideration, and forgoes other auxiliary data to characterize the users and items, limiting their expressiveness. Second, the sparsity issue adversely hinders the effectiveness of CF methods, especially when the size of item set is extremely large, the sparse interactions data is insufficient to model the parameters accurately. Moreover, CF methods usually suffer from the cold-start issues, namely, they fail to recommend new items (*cf.* cold-start items) or new users (*cf.* cold-start users) since no historical interactions are used to learn their latent vectors.

2.1.2 Content-based Methods

Content-based recommendation is mainly to recommend items similar to those a given user interacted with before. The basic process performed is based on the comparison between the content features of a user profile and the target item [100]. Whereby, for each user, it leverages the contents of the historical items that she liked before to build her profile, which are assumed to reflect the user's content-based interests. A various range of auxiliary data, such as texts, keywords, and images, can be utilized to construct the content feature for an item. Typically, the texts of items is one of the most widely used content features, where each keyword is weighted in a specific way. Early works use the frequency or importance of keywords to profile an item and a user. For instance, Fab system, proposed to recommend Web pages to users, represents each page content with the top 100 important words [8]. Except for the keyword frequency, other complex functions like TF-IDF and BM25 are adopted to measure the importance of keywords. For example, the work in [22] makes use of TF-IDF and BM25 to weight the social tags and accordingly describes the user and item profiles. The work in [167] considers content-based recommendation of TV-broadcasts where meta-data and synopses are used to select the top keywords for representing items.

Having established the user and item profiles, many techniques can be conducted to classify whether the target item is relevant to the given user. Many machine learning models, such as decision trees, k-nearest neighborhood, and naive Bayes classifier, are involved to estimate the relevance score of a user for an item. For example, the work in [100] adopts the naive Bayes method to estimate the probability that an item is relevant based on the prior probability distribution available for each user. The similarity measurements, like cosine similarity [22], are also widely used to calculate the relevance of users' and items' feature vectors.

As content-based recommendation mainly relies on the content features of user and item instead the historical user-item interactions, it can handle cold-start items, for which few or no user interactions exist. Moreover, the content features with the similarity measurements can be treated as the reasons why a recommendation is suitable for the user, endowing the recommender systems

with explainability. However, there exist several major shortcomings in these approaches. First, they lack collaborative filtering effect, that similar users tend to have similar preferences on items, assuming individual users are independent of each other. Such independence assumption limits the diversity and novelty of the recommended results, since the recommended items need to match the historical items, rather than uncovering the potential relevant items. Second, the recommendation performance heavily depends on the quality of content features. Namely, when insufficient information differentiating items in within the content features, the recommendation methods easily fail to provide high quality results. Lastly, while tackling the cold-start item issue, these methods fail to construct the profile for a cold-start user owing to no historical interactions available.

2.1.3 Hybrid Methods

Hybrid systems combine the CF and content-based methods to utilize the strengths of both methods [6] and remedy the shortcomings. Depending on the hybridization approach, existing methods can be categorized into two groups: early fusion and late fusion. First, early fusion refers to combining both explicit contents and latent factors, followed by some CF methods to boost the prediction accuracy. For example, collaborative knowledge base embedding (CKE) method [178] combines the explicit contents, including as visual, textual, and knowledge-aware features, with the latent vector from CF to represent the representation for each item. Similarly, the work in [156] proposes a 3D CNN model for session-based recommendation, where the content features of items and session clicks are considered to boost the prediction accuracy. On the other hand, late fusion build separate recommender systems that are specialized to each kind of information, and then combine the predictions of these systems. For instance, various types of combinations can be used, such as a weighted approach [40], voting mechanism [124], and boosting algorithm [105, 123]. To alleviate the cold-start issue, the work in [123] attempts to generate new synthetic ratings based on content information, and then employ CF on the new user-item rating matrix. It has been classified that hybrid recommender systems, empirically, outperforms the pure CF or content-based methods, especially for solving the cold-start user and item issues.

2.2 Exploiting Side Information for Recommendation

Despite the success of CF, it solely depends on the historical user-item interactions, but ignoring the side information, which limits the expressiveness and capacity of the recommender system. In real-world scenarios, rich heterogeneous side information is available. For example, in Amazon, products are associated with many attributes like categories and brands, as well as the comments provided by users; in TripAdvisor, users can construct their social networks by following other likeminded users and build personal profiles using demographics, where point-of-interests are affiliated with rich descriptions like price and location. Such abundant side information can help to capture users' preferences and item properties better, ranging from user and item metadata [4, 148, 122, 134, 119], contextual comments, social relations, multimedia to knowledge graphs.

User and Item Metadata: Indeed, there exists relevance signal of user preferences in the user metadata (*e.g.,* demographic information about user like age, job, and gender) and item metadata (*e.g.,* categories, price, and properties). Since most of these information are categorical variables, existing methods first convert them into real-valued feature vectors via one-hot encoding and then adopt hidden factor models to incorporate with CF. For example, to solve cold-start situations, Stern *et al.* [148] proposed a latent factor model to project the categorical user metadata (*e.g.,* age, job, and gender) into a latent space. Park *et al.* [122] proposed feature-based regression models that use metadata of users and items with the same spirit. Rendle *et al.* [134] applied FM to model such kind of contextual information and the second-order interactions between two features. Moreover, the work in [157] attempts to explain a recommendation by considering the matching between the relevant tags of an item and the preferred tags of the user.

However, prior efforts mostly treat the user and item metadata holistically, rarely differentiating heterogeneous data or hardly capturing high-order cross feature effects. Treating information across heterogeneous channels equally may lead to information loss and make negative transfer. Furthermore, feature crossing can effectively reflect the underlying feature dependencies within the data and endow the model with high explainability. Hence, most existing

methods have the limited capacity and expressiveness.

Contextual Comments: User comments (*aka.* reviews), reflecting the fine-grained sentiments w.r.t. specific aspects of items, have been utilized to assist recommender systems. Various domains contain the available contextual comments, spanning from movies [47], hotels [107], restaurants [60] to E-commerce [103]. Regardless of application domain, prior efforts mostly extract words, aspects, or sentiments from the contextual comments, and then factorize them with a latent vector to enable the collaborative filtering effect. For instance, Bao *et al.* [9] adopted a topic modeling technique (*i.e.,* non-negative matrix factorization) to model the latent topics in reviews and simultaneously used MF for rating prediction. Pero *et al.* [125] constructed a user-item opinion matrix, each entry of which is the sentiment score of a review, and then applied CF on the opinion matrix. Similarly, Diao *et al.* [47] jointly modeled the sentiment, aspects, ratings in a graphical model for movie recommendation. Recently, Zhang *et al.* [184] developed an explicit factor model, which incorporated user sentiment w.r.t. item aspects as well as the user-item ratings, to facilitate generating aspect-based explanations. Due to its rich semantics, the contextual content is usually used to conduct explainable recommendation. For example, Ren *et al.* [129] involved the viewpoints, a tuple of user sentiment and item aspect, and trusted social relations in a latent factor model and presented personalized viewpoints as explanations.

Social Relations: Prior CF methods mostly assume that individual users are independent of each other, ignoring the social connections among them. However, when adopting an item, we usually turn to friends for suggestions. Several social recommendation approaches [74, 102, 77, 101, 140] hence have been proposed to utilize social networks, demonstrating the effectiveness especially when the user-item interactions are sparse. Intuitively, the structure of social network can facilitate recommender systems to better narrow the user latent space. For instance, Ma *et al.* [102] injected a social regularization term into the MF framework to ensure connected users have similar latent factors. Jamali *et al.* [74] proposed a trust-based approach which leverages the trust network among users and then produce recommendation based on the trusted users' ratings. Moreover, additional contents, such as the retweets between two friends in Twitter, are used to enhance the trust between two users. For example, Jiang

et al. [77] proposed a social contextual recommendation framework, where the user-user influence matrix is integrated with user and item feature matrices, and MF is employed to incorporate individual preference and interpersonal influence. Moreover, social relations are beneficial for explainable recommendation. The work in [140] proposed a generative model and utilized the social connects to explain user preferences.

It is worth noting that the aforementioned studies are all based on social network relations of an information domain. While in this work, we focus on how to distill useful signal from an external social network (*e.g.,* Facebook and Twitter), so as to improve the recommendation service of any information domain.

2.3 Cross-domain Recommendation

Distinct from the traditional recommendation methods that focus on data within a single domain, cross-domain recommendation concerns data from multiple domains. A common setting is leveraging the user-item interaction of a related auxiliary domain to improve the recommendation of the target domain. However, existing cross-domain recommendation work has an underlying assumption that the target and auxiliary domains are homogeneous. Depending on [50, 76, 51], they can be divided into two directions. One is assuming that different domains share overlapped user or item sets. The work [137] augments ratings of movies and books for the shared users and accordingly conducts CF. Based on the shared users' latent space, the authors in [29] leveraged cluster-level tensor sharing as a social regularization to bridge the domains. One more step, the authors in [72] formulated a generalized triadic user-item-domain relation over the common users and accordingly to capture domain-specific user factors and item factors. More recently, the authors [50] proposed a multi-view deep learning recommendation system by using auxiliary rich features to represent users from different domains. Without aligned user or item, the other direction is on homogeneous data with the same rating scale. Codebook Transfer [86] represents cluster-level rating patterns between two rating matrices in two related domains. [151] introduces a topic model to recommend authors to collaborate from different research fields.

Despite the compelling success achieved by previous work, little attention has been paid to recommendation across heterogeneous domains. In our settings, the source domain is a social network with user-user relations only, while the target domain is an information domain with user-item interactions. Hence, the auxiliary information is the social friendship, rather than the conventional interaction data. As a result, existing approaches can be hardly applied to this new research problem.

2.4 Knowledge-aware Recommendation

The recent work on recommendation using knowledge resources can be roughly categorized into the embedding and path fasions.

Embedding-based Methods. Work [178, 120, 13, 160, 108] maps the items to relevant entities in the knowledge graph and leverages the structural knowledge embedding techniques to enrich the representations of items. Based on the anchor entities, they collaborate with the collaborative filtering techniques to make recommendations. For example, Zhang *et al.* [178] integrated matrix factorization and TrnasR into a unified framework to learn the latent representations in collaborative filtering as well as item's semantic representations from the knowledge base. Through this way, their proposed method can capture the implicit relationship between users and items, and further boost the recommendation performance. Bellini *et al.* [13] exploited the structure of knowledge graph to draw a topology of the underlying neural networks and assigned each neuron in the networks with an item to autoencode the ratings of the users. More recently, Wang *et al.* [160] aligned the words in the news content with the relevant entities in the additional knowledge base to generated a knowledge-aware embedding for each news and integrated the knowledge graph representation with content-based filtering methods to make news recommendation. While implicitly encoding the semantics of knowledge into the representations of items, the embedding-based methods only use knowledge graph to perform feature learning and cannot seamlessly fuse the knowledge into the recommendation process. Only employing the translational principles on triples, none of this work considers information propagation and preference inference along with the paths in the knowledge graph, especially in

the recommendation scenarios. As a result, some natural properties of knowledge graph, such as reasoning and explainability, are not fully explored.

Path-based Methods. In the literature of the path-based methods, it has been studied that extracts the paths in the knowledge graph to infer users' preferences on the items explicitly. The prior work [24] treats items as the relevant entities in a knowledge graph and ranks items using Personalized PageRank. In a similar line, Catherine *et al.* [23] proposed probabilistic logic programming models for recommendation on knowledge graph. On the other hand, Yu *et al.* [174] proposed a meta-path based method to infer users' preferences on items in heterogeneous information networks. Upon that, several methods [141, 185, 48, 121] have been proposed for considering the semantic on relations. Shi *et al.* [141] considered weighting meta-path by distinguishing the semantic of relations within the heterogenous information network, obtaining the personalized weighted preferences on paths. More recently, the authors in [185] utilized the meta-path and meta-graph to compute the user-item similarity matrix and employed the factorization models upon these similarity matrices to make recommendation. While able to identify the personalized preferences on items based on the paths, such methods fail to generate the knowledge-aware representations for entities and hence hardly generalize to the unseen path, suffering from the suboptimal performance.

In this thesis, we focus on exploiting side information, especially cross-channel data, for improving the recommendation performance in different scenarios.

Chapter 3

Event Recommendation across Online and Offline Channels

This chapter explores the first issue in event recommendation across online and offline channels, which typically refer to online interactions and offline activities in event-based social platforms like Meetup.

3.1 Introduction

The past decades have witnessed the proliferation of online social networking services (SNSs), such as Facebook and Twitter, which break the barrier of physical distance and allow interactions among friends in a virtual world. As a new branch of SNS, event-based social networks (EBSNs), are taking root and exerting a tremendous fascination on people. Distinct from conventional online SNSs, which usually connect acquaintance, EBSNs not only help individuals find like-minded event attendees and further form the lasting, influential, and local community groups, but also facilitate them to regularly meet face-to-face in various physical locations. In other words, EBSNs emphasize more on organizing "regional" physical events, such as "1 hour of bubble soccer" and "bubble bump sports at Kovan Sports Centre Pte Ltd" among local communities. To date, the unique and interesting services of EBSNs have attracted a sheer volume of users. The most representative examples are Meetup[1] and Facebook Event[2]. Take Meetup as an example, as of August 2015, it claimed to have 27.7 million

[1] `http://www.meetup.com/`.
[2] `http://https://www.facebook.com/events/upcoming/`.

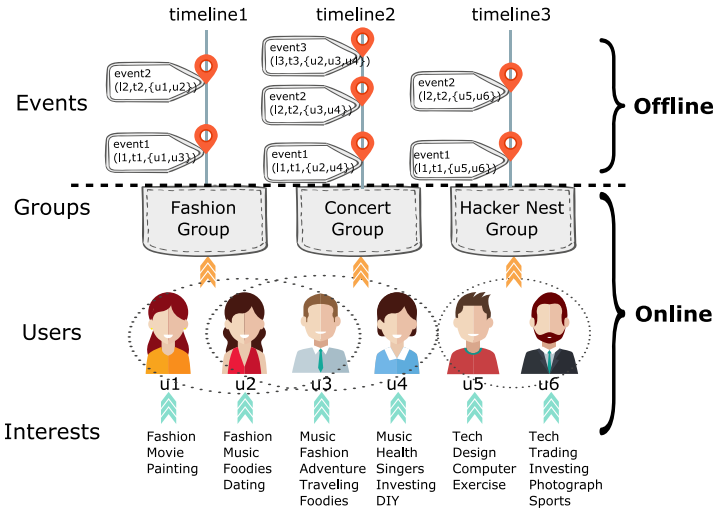


Figure 3.1: Demonstration of EBSNs, where users u can be involved in multiple online groups represented by dash circles and attend the specific physical event at the specific location l and timestamp t.

members in 180 countries and 210,240 groups[98, 127].

As demonstrated in Figure 3.1, user behaviors within EBSNs can be classified into online and offline channels. In the online channel, users are associated with several interest tags such as fashion, music, and technique [88]. They can be concurrently involved in multiple online groups according to their personal interests [158, 146]. Within the same groups, users can interact with each other online by exchanging thoughts and sharing experiences on the topics of common interests [88]. On the other hand, in the offline channel, EBSNs support and record user interactions in the physical world, where users can create and invite others to join social events, such as dining out and jogging, at specific places and times [177]. As we can see, each network may contain some knowledge that the other does not have; therefore, they can be jointly employed to comprehensively and accurately describe the user behaviors. In fact, user behaviors across online and offline networks are mutually reinforced [152, 89, 114, 62]. On one hand, online group users sharing common interests hold a better-than-average chance in attending the same offline events. On the other hand, co-participating the offine events can, in turn, strengthen their social ties and further propel the engagement of online interactions [54, 114]. In summary, online and offline user behaviors are usually compatible and complementary to each other rather than independent. We thus argue that, as compared to pure online or offline channel, appropriate aggregation of them can provide us a better way to comprehensively understand user behaviors and their underlying organizational principles.

In this chapter, we consider the task of event recommendation (*i.e.,* event attendance prediction). To achieve this, we first explore the intrinsic relations across online and offline channels to group users into different communities. Typical community detection aims at grouping persons who are connected to each other by relatively durable social relations to form a tight and cohesive social entity due to the presence of a "unity of will" or "sharing of common values" [111, 75, 114]. Community detection by jointly considering online and offline networks will offer opportunities to gain new insights into user behaviors, their organizational structures, and further support new services and applications. However, this task is non-trivial and poses a set of tough challenges:

- **User Contextualization.** Due to a variety of complex reasons, some users may be active in online interactions, but rarely attend physical events, or vice versa. This leads to the problem of inconsistent and unbalanced interactive behaviors of the same users across online and offline networks. Moreover, the same users probably own different close neighborhoods, namely social circles, in these different networks. Therefore, how to design a suitable approach to characterize the invariant essence of users across heterogeneous networks remains largely untapped.

- **Structure Coherence.** From an overall perspective, the online and offline networks are composed of the same set of users; meanwhile, they interpret their behaviors from different but coherent angles. Accordingly, we believe that the inherent and hidden community structures embedded in these two networks should be similar. How to properly model the consensus structures to reinforce the performance of community detection is, however, a challenging issue.

- **Effectiveness Evaluation.** How to quantitatively validate the result of community detection and verify its applicability in real-world problems is another challenge we face.

Community detection across online and offline networks can be treated as a particular task of either multi-view clustering or community detection in multi-layer graphs. These two clustering technologies both exploit and fuse information from multiple views [85, 111, 153, 90]. It is worthwhile mentioning

that a wide range of approaches have been proposed, with strong theoretical underpinnings and great practical success. In general, prior efforts can be roughly summarized into two categories. One is to concatenate a set of features extracted from multiple views before applying any conventional off-the-shelf clustering algorithms on these features [15, 25]. The other is to derive clustering structures from individual views or individual layer graphs and reconcile them based on the principle of consensus [19, 57, 82, 83, 98, 188, 16, 175]. However, the approaches in the first category usually ignore the relatedness among inter-networks, and frequently suffer from the curse of dimensionality and incompatible feature space. The second category is to derive clustering structures from individual views or layer graphs and reconcile them based on the principle of consensus. The approaches in this category usually model the global consensus across views or layers, and consider more comprehensive representations of users compared with that of the first category. However, rare efforts are dedicated to involving the local consensus, which is more beneficial to user contextualization [171]. In addition, few of them have been applied to solve the community detection over EBSNs.

To tackle the challenges above, we propose a novel dual-**CL**ustering model for community d**E**tection o**V**er dual-n**E**two**R**ks, CLEVER for short. Rather than learning from online and offline separately, it is capable of reinforcing community detection over online and offline networks by co-regularizing the local and global consistency in a unified model. To be more specific, local consistency is to encourage the invariants of the same users across different networks, whereby users are contextualized by their surrounding neighbors. This is inspired by the proverb "a leopard never changes his spots in different environments" and the principle of "neighborhood-preserving" [135, 78, 114]. Comparatively speaking, global consistency aims to maximize the agreement on community detection results in separate networks, which intelligently ensures the similar underlying structures over the same set of users. We co-regularize the local and global consistency in our proposed CLEVER model at the same time. We verify our model on a publicly accessible dataset and apply it to a real-world application: event attendance prediction. By conducting experiments on a representative dataset crawled from Meetup, we demonstrate that our proposed model yields significant gains in EBSNs community detection, and achieves fairly satisfactory

results for the task of event attendance prediction.

The main contributions are threefold:

- We propose a novel dual-clustering model for community detection over virtual-physical networks in EBSNs. It enhances the community detection performance by harvesting the compatible and complementary information cues with local and global consistency.
- We theoretically optimize our proposed model by the novel and effective Crank-Nicolson-like update scheme, which tackles the quadratic programming optimization problem with orthogonal constraints.
- We validate the effectiveness of our proposed CLEVER model on the application of event attendance prediction. In addition, we have released our data, code, and parameter settings to facilitate other researchers to repeat our experiments and verify their own models[3].

The remainder is organized as follows. Section 5.2 reviews the related work. Sections 3.3 and 3.4 respectively detail our proposed CLEVER model and present experimental results and analyses. Section 3.5 introduces the application, followed by the conclusion and future work in Section 3.6.

3.2 Related Work

Multi-view clustering and multi-layer graphs clustering methods are suitable for dual-clustering over the dual-networks in EBSNs. The basic idea of these approaches is to partition objects into clusters based on multiple representations of the object from different views. In recent years, multi-view and multi-layer graph learning has been receiving increasing attention and existing algorithms can be classified into two categories: early fusion [15, 25] and joint learning [19, 57, 82, 83, 98, 183, 175, 16, 20, 66, 90].

The early fusion methods concatenate all views into a single one and then adapt any conventional off-the-shelf clustering algorithms, such as k-means [15, 25]. For instance, Kamalika et al. [25] constructed a lower-dimensional feature subspace from multiple views of data by Canonical Correlation Analysis

[3] `https://goo.gl/EZPYpy`.

(CCA), and then applied single linkage clustering on the projections. Blaschko et al. [15] utilized kernel-CCA to simultaneously learn linear projections from multiple spaces into a common latent space, and then proposed a generalization of spectral clustering based on the latent space. However, these approaches generally overlook the obvious fact that each view has its own specific statistical property, and ignore the structural relatedness among views. Additionally, these approaches suffer from the over-fitting problem in case of insufficient training samples.

The other paradigm of joint learning aims to derive clustering structures and attributes from individual views and reconcile them based on the principle of consensus. This line of research can be divided into several sub-directions. 1) Normalized cut, generalized from a single view to multiple views, finds a cut which is close to the optimal one on each graph. Zhou et al. [188] generalized the normalized cut approach to multiple views via a random walk formulation. 2) Co-training is to limit clustering result in each view to agree with those in other views. Kumar et al. [82] incorporated a co-training framework with multi-view spectral clustering, where the graph structure of one view is constrained and modified by the eigenvectors of the Laplacian in other views. 3) Joint nonnegative matrix factorization (NMF) for multi-view clustering is to jointly factorize the multiple matrices through co-regularization. He et al. [66] applied the NMF on multi-view data to obtain coefficient matrices derived from factorizations of different views and further regularized them towards a common consensus. Moreover, Ni et al. [113] developed a flexible and robust framework based on NMF that clusters multiple domain-specific networks sharing multiple underlying clustering structures. 4) Co-regularization encourages corresponding data points in each view should have the same cluster membership via co-regularizing the clustering hypotheses. Kumar et al. [83] co-regularized the clustering hypotheses from multiple views and further enforced these hypotheses to be consistent across all the views via a disagreement measurement. Cai et al. [21] proposed an effective multi-modal spectral clustering method via learning a commonly shared graph Laplacian matrix by unifying different views.5) Pattern mining focuses on attribute and structural information of multiple graphs. Boden et al.[16] proposed a multi-layer graph learning method considering both aspects of structural density and attribute similarity of nodes.

Recently, Liu et al. [98] employed an extended Fiedler method to incorporate the heterogeneity between online and offline networks during the community detection process. Although existing approaches focus on the principle of global consensus across views, they ignore the local consistency of nodes' contextualization. In addition, few of them are applied to the situation of community detection over online and offline networks.

3.3 Stepwise Model Demonstration

We first declare some notations. In particular, we use bold capital letters (*e.g.,* $\mathbf{X}$) and bold lowercase letters (*e.g.,* $\mathbf{x}$) to denote matrices and vectors, respectively. We use $\mathbf{I}$ to denote the identity matrix. If not clarified, all vectors are in column forms. We employ non-bold letters (*e.g.,* x) to represent scalars, and Greek letters (*e.g.,* β) as parameters. We denote the norm and the trace of matrix $\mathbf{X}$ as $\|\mathbf{X}\|_F$ and $tr(\mathbf{X})$, respectively. Moreover, let $X(i,j)$, $\mathbf{X}(i,:)$, and $\mathbf{X}(:,j)$ respectively denote the entry in row i and column j, the i-th row and the j-th column of $\mathbf{X}$. We utilize the superscripts v and p to indicate the virtual (online) and physical (offline) networks, respectively, as well as subscript i to represent the specific user i.

3.3.1 Problem Statement

Suppose we have a set of N users. Let us denote their representations in online and offline networks as $\mathbf{X}^v = [\mathbf{x}_1^v, \ldots, \mathbf{x}_N^v] \in \mathbb{R}^{D_v \times N}$ and $\mathbf{X}^p = [\mathbf{x}_1^p, \ldots, \mathbf{x}_N^p] \in \mathbb{R}^{D_p \times N}$ respectively, where D_v and D_p are the corresponding feature dimensions. In particular, we characterize the user online and offline representations (i.e., $\mathbf{X}^v$ and $\mathbf{X}^p$) by using their interest tags and physical event attendance records, respectively. D_v and D_p separately denote the number of all interest tags and event records.

The basic insight of community detection is to encourage the similar users to have the similar community assignments [127]. Towards this end, we need to reveal the pairwise similarities among users and then construct the networks, which explore the implicit topological structure among users. In this work, we employ the Gaussian similarity function, a widely-used method, on user representations to obtain the online and offline social affinity graphs. It is

noted that some off-the-shelf clustering methods, such as K-means, applied on user representations rather than social networks, hardly lead themselves to the explicit topology and compact communities. To present more clearly, we use $c \in \mathcal{C} = \{v, p\}$ to represent the channel indicator, where v and p are the online and offline indicator, respectively. In particular, we construct the online and offline social affinity graphs $\mathbf{A}^v \in \mathbb{R}^{N\times N}$ and $\mathbf{A}^p \in \mathbb{R}^{N\times N}$ by calculating their pairwise similarities with the Gaussian similarity function [116, 114, 118],

$$\begin{cases} A^v(i,j) = \exp\left(-\frac{\left\|\mathbf{x}_i^v - \mathbf{x}_j^v\right\|^2}{(\theta^v)^2}\right) \\ A^p(i,j) = \exp\left(-\frac{\left\|\mathbf{x}_i^p - \mathbf{x}_j^p\right\|^2}{(\theta^p)^2}\right) \end{cases}, \tag{3.1}$$

where the radius parameters θ^v and θ^p are simply set as the median of the Euclidean distances of all user pairs in online and offline networks separately. We leave the further exploration of the similary function design as the future work.

Our research objective is to obtain compact and cohesive partitions $\mathcal{P}^v = \{\mathcal{P}_1^v, \ldots, \mathcal{P}_C^v\}$ and $\mathcal{P}^p = \{\mathcal{P}_1^p, \ldots, \mathcal{P}_C^p\}$ over the virtual and physical networks simultaneously, where $\mathcal{P}_i^v$ ($\mathcal{P}_i^p$) refers to the i-th community identified from the virtual (physical) network, and C denotes the number of clusters. Meanwhile, $\mathcal{P}^v$ and $\mathcal{P}^p$ should be as consensus as possible from the perspective of local user level and global structure level.

For the ease of formulation, inspired by the work in [46], we define two scaled assignment matrices $\mathbf{G}^v = [G^v(i,c)] \in \mathbb{R}^{N\times C}$ and $\mathbf{G}^p = [G^p(i,c)] \in \mathbb{R}^{N\times C}$ to respectively represent $\mathcal{P}^v$ and $\mathcal{P}^p$. In particular, they are formulated as follows,

$$\begin{cases} G^v(i,c) = \begin{cases} \frac{1}{\sqrt{|\mathcal{P}_c^v|}}, & \text{if } \mathbf{x}_i^v \in \mathcal{P}_c^v \\ 0, & \text{otherwise} \end{cases}, \\ G^p(i,c) = \begin{cases} \frac{1}{\sqrt{|\mathcal{P}_c^p|}}, & \text{if } \mathbf{x}_i^p \in \mathcal{P}_c^p \\ 0, & \text{otherwise} \end{cases}. \end{cases} \tag{3.2}$$

According to Eqn.(3.2), we can derive the orthogonal properties of $\mathbf{G}^v$ and

$\mathbf{G}^p$ as follows,

$$\begin{cases} \mathbf{G}^{v\top}\mathbf{G}^v = \mathbf{I} \\ \mathbf{G}^{p\top}\mathbf{G}^p = \mathbf{I} \end{cases}. \tag{3.3}$$

Rather than forcing $\mathbf{G}^v = \mathbf{G}^p$, *i.e.*, the same community structure over the dual-networks, we remain the different but coherent structures over different networks, *i.e.*, $\mathbf{G}^v$ and $\mathbf{G}^p$, considering that users may have inconsistent online and offline behaviors. Hence, how to bridge the gap between these structures, plays a pivotal role in propagating interdependent information implicitly and further boosting the clustering performance.

3.3.2 Clustering over Individual Network

Clustering [37, 163, 168, 97] over a single network thus far has been well-studied. Especially, spectral clustering [110] has become one of the most popular modern clustering algorithms. Theoretically, it is simple to implement via utilizing eigenvectors of Laplacian matrix derived from the data. In practice, it can be solved efficiently by a standard linear algebra software and generally outperforms traditional clustering methods such as k-means. Following the standard spectral clustering framework, we can separately perform clustering over the online and offline networks via the following formulations,

$$\begin{cases} \min\limits_{\mathbf{G}^v} tr(\mathbf{G}^{v\top}\mathcal{L}^v\mathbf{G}^v), \quad \text{s.t. } \mathbf{G}^{v\top}\mathbf{G}^v = \mathbf{I} \\ \min\limits_{\mathbf{G}^p} tr(\mathbf{G}^{p\top}\mathcal{L}^p\mathbf{G}^p), \quad \text{s.t. } \mathbf{G}^{p\top}\mathbf{G}^p = \mathbf{I} \end{cases}, \tag{3.4}$$

where $\mathcal{L}^v$ and $\mathcal{L}^p$ are the normalized Laplacian matrices over online and offline networks, respectively,

$$\begin{cases} \mathcal{L}^v = \mathbf{I} - (\mathbf{D}^v)^{-\frac{1}{2}}\mathbf{A}^v(\mathbf{D}^v)^{-\frac{1}{2}} \\ \mathcal{L}^p = \mathbf{I} - (\mathbf{D}^p)^{-\frac{1}{2}}\mathbf{A}^p(\mathbf{D}^p)^{-\frac{1}{2}} \end{cases}, \tag{3.5}$$

where $\mathbf{D}^v$ and $\mathbf{D}^p$ are the diagonal degree matrices. Despite huge empirical success of spectral clustering methods over a single network, they are unable to leverage the compatible and complementary relatedness among multiple

networks to enhance the partition performance. Hence, their current stage is suboptimal to community detection over EBSNs.

3.3.3 Clustering over Dual-Networks towards Global Consistency

To achieve improved community detection performance over EBSNs, we simultaneously harvest the information from the online and offline views. The basic idea is that these two networks admit the shared underlying clustering structure. Namely, corresponding users should have the consistent cluster memberships across the dual-networks. Here we encode the cluster memberships of users with the scaled assignment matrices $\mathbf{G}^v$ and $\mathbf{G}^p$, which can be viewed as new user representations (with the i-th row standing for the i-th user representation). With such new representations at hand, we aim to encourage the global consistency.

- ***Global consistency*** *encourages all the pairwise similarities of users under the new representations to be similar over the dual-networks. This amounts to enforcing the online and offline clustering structures to be consistent in a global view.*

Inspired by the disagreement measurement function presented in [83, 153], we propose to minimize the following function over dual-networks to implement the global consistency,

$$D_{global}(\mathbf{G}^v, \mathbf{G}^p) = \left\| \frac{\mathbf{G}^v \mathbf{G}^{v\top}}{\|\mathbf{G}^v\|_F^2} - \frac{\mathbf{G}^p \mathbf{G}^{p\top}}{\|\mathbf{G}^p\|_F^2} \right\|_F^2, \tag{3.6}$$

where $\mathbf{G}^v(i,:)$ and $\mathbf{G}^p(i,:)$ refer to the online and offline cluster memberships for the same user i, respectively. We treat the scaled community assignment matrices $\mathbf{G}^v$ and $\mathbf{G}^p$ as the new user representations [83]. Accordingly, $\mathbf{G}^v\mathbf{G}^{v\top}$ and $\mathbf{G}^p\mathbf{G}^{p\top}$ represent the online and offline pairwise similarities among users, respectively. The similarity matrices are normalized by their norms to make the same users comparable across networks. We note that $||\mathbf{G}^v||_F^2 = tr(\mathbf{G}^{v\top}\mathbf{G}^v) = C$ and $||\mathbf{G}^p||_F^2 = tr(\mathbf{G}^{p\top}\mathbf{G}^p) = C$, where C is the number of communities, and the orthogonal constraints can make them comparable in nature.

We observed that some users have somehow different behaviors across the physical and virtual societies. This may result in that users have different grouping distributions. To address such problem, we intentionally do not

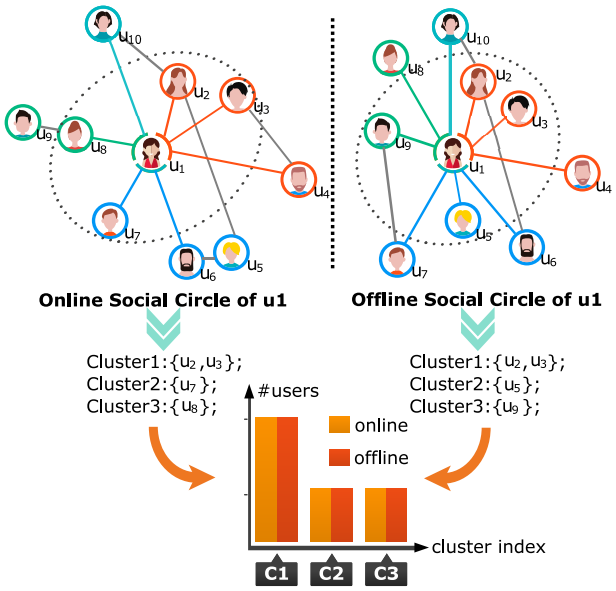


Figure 3.2: Illustration of local consistency over dual-networks for a specific user u_1. Dash circles represent his/her social circles with the size of four and different color edges denote different communities.

enforce the grouping information over the online and offline networks to be equal. Instead, we leverage the pairwise similarity among users to illustrate the global consistency, since we found that users' relations are relatively stable across networks.

Combining global consistency with the spectral clustering objectives mentioned in Eqn.(3.4) and Eqn.(3.5), we obtain a joint optimization problem as follows,

$$\min_{\mathbf{G}^v,\mathbf{G}^p} tr(\mathbf{G}^{v\top}\mathcal{L}^v\mathbf{G}^v) + tr(\mathbf{G}^{p\top}\mathcal{L}^p\mathbf{G}^p) + \frac{\lambda_1}{2}\left\|\frac{\mathbf{G}^v\mathbf{G}^{v\top}}{\|\mathbf{G}^v\|_F^2} - \frac{\mathbf{G}^p\mathbf{G}^{p\top}}{\|\mathbf{G}^p\|_F^2}\right\|_F^2,$$
$$\text{s.t.} \quad \mathbf{G}^{v\top}\mathbf{G}^v = \mathbf{I}, \quad \mathbf{G}^{p\top}\mathbf{G}^p = \mathbf{I}. \tag{3.7}$$

3.3.4 Clustering over Dual-Networks towards Local Consistency

Apart from the global consistency, we further utilize the "neighborhood preserving" matching to explore the consensus over the dual-networks. In particular, the "neighborhood preserving" assumes the same users in different networks may have similar ego-networks and social circles in nature [78, 170, 180]. For each user, we contextualize the social circles by his/her nearest neighbors. We formulate this idea as the problem of "local consistency".

For a given user i, we calculate his/her similarities with others based on

Eqn.(3.1). Hereafter, we select the K users with highest similarities as the nearest neighbors to construct his/her online and offline social circles. We denote them as $\mathcal{N}_K(\mathbf{x}_i^v)$ and $\mathcal{N}_K(\mathbf{x}_i^p)$,

$$\begin{cases} \mathcal{N}_K(\mathbf{x}_i^v) = \{i_1^v, \ldots, i_K^v\} \\ \mathcal{N}_K(\mathbf{x}_i^p) = \{i_1^p, \ldots, i_K^p\} \end{cases}, \tag{3.8}$$

where i_k^v and i_k^p indicate the k-th nearest neighbor of the i-th user in online and offline settings, respectively. We thus can define two selection matrices $\mathbf{S}_i^v = [S_i^v(j,k)] \in \mathbb{R}^{N \times K}$ and $\mathbf{S}_i^p = [S_i^p(j,k)] \in \mathbb{R}^{N \times K}$ for each user to select its neighbors based on Eqn.(3.8),

$$\begin{cases} S_i^v(j,k) = \begin{cases} 1, & \text{if } j = i_k^v \in \mathcal{N}_K(\mathbf{x}_i^v) \\ 0, & \text{otherwise} \end{cases}, \\ S_i^p(j,k) = \begin{cases} 1, & \text{if } j = i_k^p \in \mathcal{N}_K(\mathbf{x}_i^p) \\ 0, & \text{otherwise} \end{cases}. \end{cases} \tag{3.9}$$

We hence can obtain the social circles $\mathbf{X}_i^v \in \mathbb{R}^{D_v \times K}$ and $\mathbf{X}_i^p \in \mathbb{R}^{D_p \times K}$ of the user i as follows,

$$\begin{cases} \mathbf{X}_i^v = \mathbf{X}^v \mathbf{S}_i^v \\ \mathbf{X}_i^p = \mathbf{X}^p \mathbf{S}_i^p \end{cases}. \tag{3.10}$$

Given a specific user i, and his/her local social circles $\mathcal{N}_K(\mathbf{x}_i^v)$ and $\mathcal{N}_K(\mathbf{x}_i^p)$, we can select the corresponding K distribution vectors over C clusters from $\mathbf{G}^v$ and $\mathbf{G}^p$, and we then can define the local scaled cluster assignment matrices $\mathbf{G}_i^v \in \mathbb{R}^{K \times C}$ and $\mathbf{G}_i^p \in \mathbb{R}^{K \times C}$ as follows,

$$\begin{cases} \mathbf{G}_i^v = \mathbf{S}_i^{vT} \mathbf{G}^v \\ \mathbf{G}_i^p = \mathbf{S}_i^{pT} \mathbf{G}^p \end{cases}. \tag{3.11}$$

Given the social circles across dual-networks, as illustrated in Figure 3.2, we aim to regularize the local consistency,

- ***Local consistency*** *encourages the cluster distributions of a specific*

user's social circle over the C clusters to be close across the dual-networks. This amounts to enforcing online and offline contextualization of the specific user to be consistent in a local view.

It is noticeable that the online and offline social circles of the same user may involve different neighbors, namely $\mathcal{N}_K(\mathbf{x}_i^v) \neq \mathcal{N}_K(\mathbf{x}_i^p)$. For instance, the k-th closest neighbors of the i-th user in the dual-networks are not necessarily the same person and hence they can fall into different clusters. However, for the same cluster c, it is assumed to have similar weights over the K neighbors, namely, the difference between $\sum_k^K G_i^v(k,c)$ and $\sum_k^K G_i^p(k,c)$ should be as small as possible. As such, we measure the local consistency by,

$$\begin{aligned} D_{local}(\mathbf{G}_i^v, \mathbf{G}_i^p) &= \sum_{c=1}^{C} \left(\sum_{k=1}^{K} \mathbf{G}_i^v(k,c) - \sum_{k=1}^{K} \mathbf{G}_i^p(k,c) \right)^2 \\ &= \left\| (\mathbf{G}_i^v - \mathbf{G}_i^p)^\top \mathbf{e} \right\|^2, \end{aligned} \tag{3.12}$$

where $\mathbf{e} \in \mathbb{R}^K$ is the all-ones' vector.

Furthermore, we can estimate the consistency of all users' online and offline social circles as follows,

$$D_{local}(\mathbf{G}^v, \mathbf{G}^p) = \sum_{i=1}^{n} \left\| (\mathbf{G}_i^v - \mathbf{G}_i^p)^\top \mathbf{e} \right\|^2. \tag{3.13}$$

Analogous to the clustering over dual-networks towards global consistency, we have the following joint optimization problem towards local consistency,

$$\begin{aligned} &\min_{\mathbf{G}^v, \mathbf{G}^p} tr(\mathbf{G}^{v\top} \mathcal{L}^v \mathbf{G}^v) + tr(\mathbf{G}^{p\top} \mathcal{L}^p \mathbf{G}^p) \\ &+ \frac{\lambda_1}{2} \sum_{i=1}^{n} \left\| (\mathbf{G}_i^v - \mathbf{G}_i^p)^\top \mathbf{e} \right\|^2, \text{s.t.} \quad \mathbf{G}^{v\top} \mathbf{G}^v = \mathbf{I}, \quad \mathbf{G}^{p\top} \mathbf{G}^p = \mathbf{I}. \end{aligned} \tag{3.14}$$

3.3.5 CLEVER Model towards Global and Local Consistency

As discussed in Section 3.3.2, the objective functions of clustering over individual network can be expressed in the form of $tr(\mathbf{G}^{v\top} \mathcal{L}^v \mathbf{G}^v)$ and $tr(\mathbf{G}^{p\top} \mathcal{L}^v \mathbf{G}^p)$, where $\mathcal{L}^v$ and $\mathcal{L}^p$ are the normalized Laplacian matrices defined based on the affinity matrices $\mathbf{A}^v$ and $\mathbf{A}^p$ in Eqn.(3.1), respectively. However, the conventional Laplacian matrix only considers the topological structure of the user networks and overlooks the discriminative community information. As proposed

in [171, 172], the discriminative information is beneficial for exhibiting the manifold structure in the networks and has a positive influence on the community detection performance. To refine the Laplacian matrices, we therefore resort to utilize the social circles across dual-networks and further uncover the intrinsic manifold structures of social communities.

To explore the discriminative information in each social circle, we turn to the Fisher criterion [56], which is widely used in image processing and computer vision. Intuitively, to obtain the better community partitions, distance between users from different communities should be as large as possible, while distance between users within the same community should be smallest. Therefore, for a given social circle of the i-th user, we define the total scatter matrices $\mathbf{T}_i^v$ and $\mathbf{T}_i^p$, which sum up all the pairwise distances between the near neighbors, and the between-cluster scatter matrices $\mathbf{B}_i^v$ and $\mathbf{B}_i^p$, which sum up all the distances between near neighbors from different clusters,

$$\begin{cases} \mathbf{T}_i^v = \sum_{j \in \mathcal{N}_K(\mathbf{x}_i^v)} (\mathbf{x}_j^v - \mu_i^v)(\mathbf{x}_j^v - \mu_i^v)^\top = \mathbf{X}_i^v \mathbf{H} \mathbf{H} \mathbf{X}_i^{v\top}, \\ \mathbf{B}_i^v = \sum_{c=1}^{C} n_c^v (\mu_i^v - \mu_i^{v,c})(\mu_i^v - \mu_i^{v,c})^\top = \mathbf{X}_i^v \mathbf{H} \mathbf{G}_i^v \mathbf{G}_i^{v\top} \mathbf{H} \mathbf{X}_i^{v\top} \\ \mathbf{T}_i^p = \sum_{j \in \mathcal{N}_K(\mathbf{x}_i^p)} (\mathbf{x}_j^p - \mu_i^p)(\mathbf{x}_j^p - \mu_i^p)^\top = \mathbf{X}_i^p \mathbf{H} \mathbf{H} \mathbf{X}_i^{p\top}, \\ \mathbf{B}_i^p = \sum_{c=1}^{C} n_c^p (\mu_i^p - \mu_i^{p,c})(\mu_i^v - \mu_i^{p,c})^\top = \mathbf{X}_i^p \mathbf{H} \mathbf{G}_i^p \mathbf{G}_i^{p\top} \mathbf{H} \mathbf{X}_i^{p\top} \end{cases}, \tag{3.15}$$

where μ_i^v and μ_i^p respectively denote the means of the nearest neighbors in $\mathcal{N}_K(\mathbf{x}_i^v)$ and $\mathcal{N}_K(\mathbf{x}_i^p)$; $\mu_i^{p,c}$ and $\mu_i^{p,c}$ separately represent the mean of the nearest neighbors within the c-th cluster; $\mathbf{H} = \mathbf{I} - \frac{1}{K}\mathbf{E} \in \mathbb{R}^{K \times K}$ is a constant matrix and $\mathbf{E} \in \mathbb{R}^{K \times K}$ is the all-ones matrix, whereby K is the number of the nearest neighbors. To better characterize the underlying structure of local circles, we define a new concept, named local discriminant score as follows,

$$\begin{aligned} F(\mathbf{G}_i^v) &= tr(\mathbf{G}_i^{v\top} \mathbf{H} \mathbf{G}_i^v - (\mathbf{T}_i^v + \lambda \mathbf{I})^{-1} \mathbf{B}_i^v) \\ &= tr(\mathbf{G}^{v\top} \mathbf{S}_i^v \mathbf{H} (\mathbf{H}^\top \mathbf{X}_i^{v\top} \mathbf{X}_i^v \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H} \mathbf{S}_i^{v\top} \mathbf{G}^v) \\ &= tr(\mathbf{G}^{v\top} \mathcal{L}_i^{v*} \mathbf{G}^v), \end{aligned} \tag{3.16}$$

where $\mathcal{L}_i^{v*} = \mathbf{S}_i^v \mathbf{H} (\mathbf{H}^\top \mathbf{X}_i^{v\top} \mathbf{X}_i^v \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H} \mathbf{S}_i^{v\top}$. Similar to Eqn.(3.16), we have $F(\mathbf{G}_i^p) = tr(\mathbf{G}^{p\top} \mathcal{L}_i^{p*} \mathbf{G}^p)$, where $\mathcal{L}_i^{p*} = \mathbf{S}_i^p \mathbf{H} (\mathbf{H}^\top \mathbf{X}_i^{p\top} \mathbf{X}_i^p \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H} \mathbf{S}_i^{p\top}$. A smaller local discriminant score indicates that the neighbors in the social circles

from different clusters are better separated.

Considering all users' social circles over dual-networks, the relations $\mathbf{G}_i^v = \mathbf{S}_i^{vT}\mathbf{G}^v$, and $\mathbf{G}_i^p = \mathbf{S}_i^{pT}\mathbf{G}^p$ in Eqn.(3.11), we reformulate the conventional standard spectral clustering methods with the local discriminant model in Eqn.(3.16) as the following optimization problems,

$$\begin{cases} \min_{\mathbf{G}^v} \sum_{i=1}^N F(\mathbf{G}_i^v) = \min_{\mathbf{G}^v} tr(\mathbf{G}^{v\top}\mathcal{L}^{v*}\mathbf{G}^v) \\ \min_{\mathbf{G}^p} \sum_{i=1}^N F(\mathbf{G}_i^p) = \min_{\mathbf{G}^p} tr(\mathbf{G}^{p\top}\mathcal{L}^{p*}\mathbf{G}^p) \end{cases}, \tag{3.17}$$

where $\mathcal{L}^{v*} = \sum_{i=1}^N \mathcal{L}_i^{v*}$ and $\mathcal{L}^{p*} = \sum_{i=1}^N \mathcal{L}_i^{v*}$ can be viewed as the modified Laplacian matrices by employing discriminant information and manifold information over online and offline networks, respectively.

At last, integrating the refined spectral clustering framework in Eqn.(3.17), the global consistency in Eqn.(3.6), and the local consistency in Eqn.(3.13) over dual-networks, we ultimately reach the objective function of our proposed CLEVER model as follows,

$$\begin{aligned} &\min_{\mathbf{G}^v,\mathbf{G}^p} tr(\mathbf{G}^{v\top}\mathcal{L}^{v*}\mathbf{G}^v) + tr(\mathbf{G}^{p\top}\mathcal{L}^{p*}\mathbf{G}^p) \qquad (3.18)\\ &+\frac{\lambda_1}{2}\sum_{i=1}^n \left\|(\mathbf{G}_i^v - \mathbf{G}_i^p)^\top \mathbf{e}\right\|^2 + \frac{\lambda_2}{2}\left\|\frac{\mathbf{G}^v\mathbf{G}^{v\top}}{\|\mathbf{G}^v\|_F^2} - \frac{\mathbf{G}^p\mathbf{G}^{p\top}}{\|\mathbf{G}^p\|_F^2}\right\|_F^2, \text{s.t.} \quad \mathbf{G}^{v\top}\mathbf{G}^v = \mathbf{I}, \quad \mathbf{G}^{p\top}\mathbf{G}^p = \mathbf{I}. \end{aligned}$$

overall, our proposed CLEVER model has two parameters as shown in Eqn.(3.18). Parameters λ_1 and λ_2 respectively regularize the local and global consistency. We will detail the parameter tuning procedure in the experiments.

Based on the scaled assignment matrices $\mathbf{G}^v$ and $\mathbf{G}^p$, we can obtain the individual partition results corresponding to the online and offline networks. Furthermore, we conducted the following equation on $\mathbf{G}^v$ and $\mathbf{G}^p$ to get the final unified community $\tilde{\mathbf{G}}$,

$$\tilde{\mathbf{G}} = \frac{1}{2}\tilde{\mathbf{G}}^v + \frac{1}{2}\tilde{\mathbf{G}}^p, \quad \tilde{\mathbf{G}}^v = \mathbf{G}^v(\mathbf{G}^v \otimes \mathbf{G}^v)^{-\frac{1}{2}}, \quad \tilde{\mathbf{G}}^p = \mathbf{G}^v(\mathbf{G}^p \otimes \mathbf{G}^p)^{-\frac{1}{2}}, \tag{3.19}$$

where $\tilde{\mathbf{G}}^v$ and $\tilde{\mathbf{G}}^p$ denotes the assignment matrices with their element as the probability of the i-th user to the j-th community; $\tilde{\mathbf{G}}$ represents the final unified assignment matrix which combines the online and offline parts and stands for the

final probability distributions over all communities. Thereafter, for each user, we can assign her/him to the community with the highest probability. From the final partition process we can find that it can guarantee the exact and clear partition result. It is noted that, by setting the flexible probability threshold, our framework can allow the users to drop into multiple communities simultaneously.

3.3.6 Optimization

We can equally transform the third term in Eqn.(3.18), *i.e.,* the local consistency term, into a trance-norm form as follows,

$$tr(\mathbf{G}^{v\top}\mathbf{U}^{v}\mathbf{G}^{v}) + tr(\mathbf{G}^{p\top}\mathbf{U}^{p}\mathbf{G}^{p}) - 2tr(\mathbf{G}^{p\top}\mathbf{U}^{v,p}\mathbf{G}^{v}), \tag{3.20}$$

where $\mathbf{U}^{v} = \sum_{i=1}^{n} \mathbf{S}_i^{v}\mathbf{E}\mathbf{S}_i^{v\top} \in \mathbb{R}^{N\times N}$, $\mathbf{U}^{p} = \sum_{i=1}^{n} \mathbf{S}_i^{p}\mathbf{E}\mathbf{S}_i^{p\top} \in \mathbb{R}^{N\times N}$, and $\mathbf{U}^{v,p} = \sum_{i=1}^{n} \mathbf{S}_i^{p}\mathbf{E}\mathbf{S}_i^{v\top} \in \mathbb{R}^{N\times N}$.

Similarly, we can restate the fourth term, *i.e.,* the global consistency term, in Eqn.(3.18), with its equivalent trace form as follows,

$$\frac{1}{C^2}tr(\mathbf{G}^{v}\mathbf{G}^{v\top}\mathbf{G}^{v}\mathbf{G}^{v\top} - 2\mathbf{G}^{v}\mathbf{G}^{v\top}\mathbf{G}^{p}\mathbf{G}^{p\top} \quad + \mathbf{G}^{p}\mathbf{G}^{p\top}\mathbf{G}^{p}\mathbf{G}^{p\top}) = \frac{2}{C} - \frac{2}{C^2}tr(\mathbf{G}^{v}\mathbf{G}^{v\top}\mathbf{G}^{p}\mathbf{G}^{p\top}).$$

As such, by ignoring the constant additive and scaling terms, our objective function in Eqn.(3.18) can be rewritten as,

$$\begin{aligned}\min_{\mathbf{G}^{v},\mathbf{G}^{p}} \mathcal{F}(\mathbf{G}^{v},\mathbf{G}^{p}) = \min_{\mathbf{G}^{v},\mathbf{G}^{p}} & tr(\mathbf{G}^{v\top}\tilde{\mathcal{L}}^{v}\mathbf{G}^{v}) + tr(\mathbf{G}^{p\top}\tilde{\mathcal{L}}^{p}\mathbf{G}^{p}) \\ & -\lambda_1 tr(\mathbf{G}^{p\top}\mathbf{U}^{v,p}\mathbf{G}^{v}) - \lambda_2 tr(\mathbf{G}^{v}\mathbf{G}^{v\top}\mathbf{G}^{p}\mathbf{G}^{p\top}), \\ \text{s.t.} \quad & \mathbf{G}^{v\top}\mathbf{G}^{v} = \mathbf{I}, \quad \mathbf{G}^{p\top}\mathbf{G}^{p} = \mathbf{I},\end{aligned} \tag{3.21}$$

where $\tilde{\mathcal{L}}^{v} = \mathcal{L}^{v*} + \frac{\lambda_1}{2}\mathbf{U}^{v}$ and $\tilde{\mathcal{L}}^{p} = \mathcal{L}^{p*} + \frac{\lambda_1}{2}\mathbf{U}^{p}$. They can be regarded as the new Laplacian matrices.

3.4 Experiments

All the experiments were conducted over a server equipped with Intel(R) Core(R) i7-4970 CPU at 3.60 GHz on 32G RAM, 4 cores and 64-bit Windows 10 operating system.

Table 3.1: Statistics of the regional Meetup dataset.

City	#User	#Interest Tag	#Event	Avg. Tags per user	Avg. Events per user
California (CA)	5,904	2,664	6,106	23.42	13.55
New York City (NYC)	6,440	2,630	7,054	17.44	12.79

3.4.1 Data Description

We conducted experiments on a publicly benchmark dataset[4], the Meetup dataset [126]. We utilized the user profiles consisting of personal interest labels and event attendance records to construct the online and offline networks, respectively. We detailed the constructions of these two networks below:

- We extracted Meetup users from the dataset and obtained $9,095$ and $8,339$ original users with their profiles and event attendance records from the state of California (CA) and New York City (NYC), respectively.

- We filtered out users with less than five interest tags and five event attendance records, in order to guarantee that all the remaining users are relatively active across dual-networks. We ultimately obtained $5,904$ and $6,440$ users in CA and NYC, respectively. The details are summarized in Table 3.1.

- We characterized the user online and offline representations (i.e., $\mathbf{X}^v$ and $\mathbf{X}^p$) by using their interest tags and physical event attendance records, respectively. In particular, for the i-th user in CA, we transformed her/his distribution over all interest tags and events into $2,664$- and $6,106$-dimensional feature vectors to separately represent $\mathbf{x}_i^v$ and $\mathbf{x}_i^p$, while $2,630$- and $7,054$-dimensional vectors for the users in NYC.

- After obtaining the original user representations, we constructed the dual-networks (i.e., $\mathbf{A}^v$ and $\mathbf{A}^p$) where we treated users as vertices and their online and offline pairwise relationships as edges. In particular, we leveraged the pairwise similarities based on Eqn.(3.1) to stand for the relationships.

As the ground truth of user online and offline communities is unavailable, we adopt the following metrics to evaluate the performance of detection community methods:

[4] `www.ntu.edu.sg/home/gaocong/datacode.htm`.

- Normalized Davies-Bouldin Index (ndbi) [44] measures the uniqueness of clusters *w.r.t.* the unified similarity measure [98, 33, 189],

$$ndbi(\mathcal{P}) = \frac{\sum_{i=1}^{C} \min_{j \neq i} \frac{d(\mathbf{c}_i, \mathbf{c}_j) + d(\mathbf{c}_j, \mathbf{c}_i)}{\sigma_i + \sigma_j + d(\mathbf{c}_i, \mathbf{c}_j) + d(\mathbf{c}_j, \mathbf{c}_i)}}{C}, \tag{3.22}$$

  where $\mathbf{c}_i$ is the centroid of community $\mathcal{P}_i \in \mathcal{P}$, $d(\mathbf{c}_i, \mathbf{c}_j)$ is the distance between centroids $\mathbf{c}_i$ and $\mathbf{c}_j$, and σ_i is the average distance between elements in $\mathcal{P}_i$ and centroid $\mathbf{c}_i$. A higher ndbi value indicates a more cohesive community.

- Silhouette index (sil) [136] validates the clustering performance based on the pairwise difference of between- and within-cluster distances [99, 33],

$$sil(\mathcal{P}) = \frac{1}{C} \sum_{i=1}^{C} \left(\frac{1}{|\mathcal{P}_i|} \sum_{u \in \mathcal{P}_i} \frac{b(u) - a(u)}{\max\{b(u), a(u)\}} \right), \tag{3.23}$$

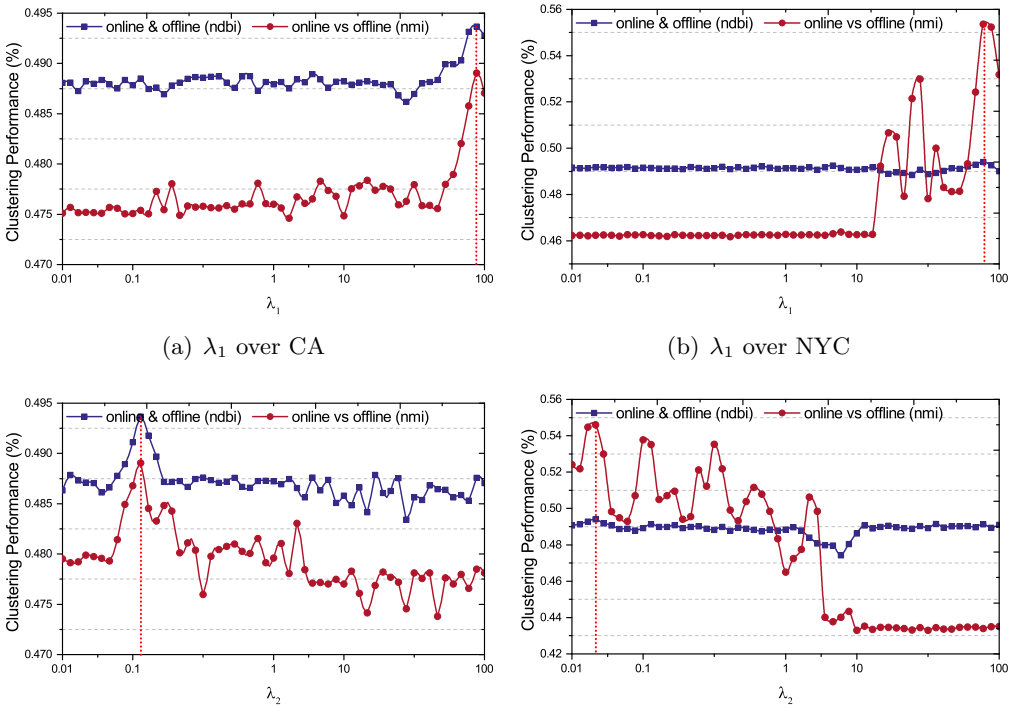
  where $a(u) = \frac{1}{|\mathcal{P}_i| - 1} \sum_{v \in \mathcal{P}_i, u \neq v} d(u, v)$ and $b(u) = \min_{j \neq i} (\frac{1}{|\mathcal{P}_j|} \sum_{v \in \mathcal{P}_j} d(u, v))$. A higher sil corresponds to a better clustering result.

- Normalized mutual information (nmi) [112, 90], as a consensus metric, gives the mutual information between online and offline clustering results normalized by the cluster entropies [83, 112, 113, 33]. We utilized it to measure how similar the online and offline communities are.

$$nmi(\mathcal{P}^v, \mathcal{P}^p) = \frac{mi(\mathcal{P}^v, \mathcal{P}^p)}{\sqrt{H(\mathcal{P}^v) H(\mathcal{P}^p)}}, \tag{3.24}$$

  where $mi(\mathcal{P}^v, \mathcal{P}^p) = \sum_{i,j=1}^{C} P(i,j) \log \frac{P(i,j)}{P(i)P(j)}$ denotes the mutual information between $\mathcal{P}^v$ and $\mathcal{P}^p$; and $H(\mathcal{P}^v) = -\sum_{i=1}^{C} P(i) \log P(i)$ represents the entropy of $\mathcal{P}^v$. We defined $P(i) = \frac{|\mathcal{P}_i^v|}{N}$ and $P(i,j) = \frac{|\mathcal{P}_i^v \cap \mathcal{P}_j^p|}{N}$. The value of nmi ranges between 0 and 1 with higher value indicating closer match between online and offline clustering results.

It is noted that the solution of CLEVER is not unique since the step size of our algorithm is adaptive for different initialization values and global minimization cannot be guaranteed. To study the sensitivity of our model to the initialization, we randomly generated ten different initialization values (*i.e.*, $\mathbf{G}_{(0)}^v$ and $\mathbf{G}_{(0)}^p$) and then fed them into our CLEVER model. For



(a) λ_1 over CA

(b) λ_1 over NYC

(c) λ_2 over CA

(d) λ_2 over NYC

Figure 3.3: Illustration of parameter tuning over CA and NYC datasets by varying one and fixing the other. The optimal setting of each parameter is marked by the red dotted line. We can see the performance regarding ndbi and nmi is non-sensitive nearby the optimal parameters.

other competitors, the initialization procedure is analogous to ensure the fair comparison. Thereafter, we performed paired t-test between our model and each of baselines over the 10-round results.

3.4.2 Parameter Tuning and Sensitivity

We have two implicit parameters, namely, the number of communities C and the size of social circle K, as shown in Eqn.(3.21). Since there is no prior knowledge of community structures in EBSNs, we verified our models and all the baselines with different numbers of communities $C \in \{20, 40, 80, 100, 150, 200, 250, 300\}$. Regarding the size of social circle K, it plays a pivotal role in the local consistency term and affects the refined Laplacian matrices as shown in Eqn.(3.13) and Eqn.(3.17). We set various sizes of social circle $K \in [10, 100]$ with step size 10.

Apart from the implicit parameters, we have two explicit parameters λ_1 and λ_2 in Eqn.(3.21). Grid search was adopted to select the optimal parameters between 10^{-2} to 10^2 with small but adaptive step sizes. In particular, the step

size was set to 0.01, 0.05, 1, and 5 for the ranges of $[0.01, 0.1]$, $[0.1, 1]$, $[1, 10]$, and $[10, 100]$, respectively. The parameters corresponding to the best ndbi were used to report the final results. For other comparing systems, the procedures to tune the parameters are analogous to ensure fair comparison.

Take the tuning procedure over CA dataset as an example. We observed that our model reached the optimal performance when $\lambda_1 = 0.15$ and $\lambda_2 = 95$. Fig. 3.3 illustrates the performance of our model over CA and NYC datasets regarding these two parameters, respectively. Jointly analyzing these four sub-figures, we can safely draw two conclusions: 1) The performance of our proposed model changes within small ranges nearby the optimal settings. This justifies that our model is not sensitive to the parameters around their optimal configuration. And 2) the curves of ndbi and nmi have the similar trends. That further indicates that the consensus of online and offline networks has an immediate impact on the community detection.

3.4.3 Scalability Discussion

Due to the increasing popularity of EBSNs, the scalability has become a major concern regarding community detection across dual-networks. We hence discuss the computational cost of each iteration in our proposed model and illustrate its efficiency in the section.

The computational cost in each iteration mainly comes from three parts:

1. Computation of $\nabla\mathcal{F}(\mathbf{W}_j)$. It is worth mentioning that the cost of the specific matrix multiplication $\mathbf{G}^p_{(t)}{\mathbf{G}^p_{(t)}}^\top$, and that of constructing the constant matrices $\tilde{\mathcal{L}}^v$ and $\mathbf{U}^{v,p}$ remain the same for all iterations in the curvilinear search method. We thus can save much practical time cost by caching the results. The complexity of $\nabla\mathcal{F}(\mathbf{W}_j)$ is $O(N^2C)$, since the multiplications of all three components.

2. Computation of $\mathbf{A}_j$. The time cost of $\mathbf{A}_j$ is $O(N^2C)$.

3. Computation of $\mathbf{Y}_j(\tau_j)$. The speed bottleneck of computing $\mathbf{Y}_j(\tau_j)$ lies in computing the inversion of $(\mathbf{I} + \frac{\tau}{2}\mathbf{A}_j)$, which has the complexity of $O(N^3)$. We thus can estimate the total cost for deriving $\mathbf{Y}_j(\tau_j)$ is $O(N^3)$.

To sum up, the computational complexity of our model is $O(N^3)$, which

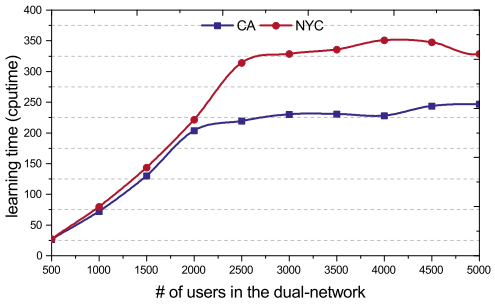


Figure 3.4: Scalability of CLEVER model by varying the number of users over the dual-networks in CA and NYC, respectively.

indeed reflects the unsatisfactory scalability. However, using the method of Coppersmith and Winograd, the cost can be bounded by $O(N^{2.376})$ [176]. Although the computational complexity of our algorithm is the major bottleneck, the scale of users involved into one region or one local community cannot be very large in reality, due to the restrictions of the regional population capacity. According to the running time of different methods in Table 3.2, we can see that the computational cost of our proposed CLEVER is acceptable. Furthermore, we can implement parallel and distributed version of our algorithm to speed up it [59, 38]. For example, we can distribute the users within the same regions to the same compute node via MapReduce and furthermore optimize the matrix multiplications.

We evaluated the scalability of our proposed CLEVER algorithm by varying the scale of users in the dual-networks. As the Fig. 3.4 illustrates, CLEVER is efficient and capable of handling relative large graphs with massive vertices in the dual-networks.

3.4.4 Overall Model Performance Evaluation

To justify the effectiveness of our proposed CLEVER model, we compared it with several state-of-the-art competitors:

1. **PCAKM**. K-means is a traditional clustering method to detect communities [128]. In this experiment, to boost the performance and improve efficiency of community detection, we first applied PCA on the original feature spaces and then adopted K-means to partition online and offline networks, separately.

2. **SC**. Spectral clustering [110] is one of the most popular modern clustering

Table 3.2: Performance comparison among various methods in terms of ndbi, nmi, and sil. We reported the results with variance on two cities CA and NYC with $C = 300$ and $K = 20$. Wherein, we illustrate the learning performance regarding various metrics over different networks, such as denoting the performance *w.r.t.* ndbi over the unified dual-networks as "online & offline (ndbi)". Significance test is based on the ndbi over online & offline networks.

Method		PCAKM	SC	RMKMC	MMSC	CoNMF	**CLEVER**
CA	online (ndbi)	$0.4700 \pm 2e\text{-}5$	$0.4571 \pm 8e\text{-}6$	$0.4472 \pm 1e\text{-}5$	$0.3886 \pm 3e\text{-}4$	$0.4152 \pm 7e\text{-}5$	$\mathbf{0.4719 \pm 7e\text{-}6}$
	offline (ndbi)	$0.4506 \pm 1e\text{-}4$	$0.4600 \pm 6e\text{-}6$	$0.4732 \pm 2e\text{-}5$	$0.4102 \pm 5e\text{-}4$	$0.4424 \pm 5e\text{-}5$	$\mathbf{0.4756 \pm 8e\text{-}6}$
	online vs offline (nmi)	$0.3844 \pm 3e\text{-}5$	$0.3763 \pm 6e\text{-}6$	1	1	1	$\mathbf{0.4891 \pm 1e\text{-}3}$
	online & offline (ndbi)	-	-	$0.4830 \pm 1e\text{-}5$	$0.4129 \pm 4e\text{-}4$	$0.4399 \pm 7e\text{-}5$	$\mathbf{0.4937 \pm 6e\text{-}6}$
	online & offline (sil)	-	-	$-0.3790 \pm 4e\text{-}4$	$-0.1228 \pm 3e\text{-}4$	$-0.2569 \pm 2e\text{-}4$	$\mathbf{-0.1427 \pm 3e\text{-}3}$
	cputime	$4.2993e2$	$2.5074e2$	$4.1844e2$	$5.1466e2$	$5.6756e4$	**4.7297e2**
	p-value	-	-	$2.7334e\text{-}6$	$4.5006e\text{-}7$	$2.1256e\text{-}8$	-
NYC	online (ndbi)	$0.4670 \pm 3e\text{-}5$	$0.4566 \pm 7e\text{-}6$	$0.4463 \pm 6e\text{-}5$	$0.3726 \pm 4e\text{-}4$	$0.4475 \pm 5e\text{-}5$	$\mathbf{0.4736 \pm 7e\text{-}6}$
	offline (ndbi)	$0.3297 \pm 1e\text{-}4$	$0.4620 \pm 2e\text{-}5$	$0.4588 \pm 7e\text{-}5$	$0.3741 \pm 5e\text{-}4$	$0.4536 \pm 4e\text{-}5$	$\mathbf{0.4748 \pm 8e\text{-}6}$
	online vs offline (nmi)	$0.3375 \pm 1e\text{-}5$	$0.4335 \pm 1e\text{-}5$	1	1	1	$\mathbf{0.5536 \pm 5e\text{-}3}$
	online & offline (ndbi)	-	-	$0.4705 \pm 6e\text{-}5$	$0.3856 \pm 5e\text{-}4$	$0.4669 \pm 4e\text{-}5$	$\mathbf{0.4940 \pm 7e\text{-}6}$
	online & offline (sil)	-	-	$-0.3455 \pm 4e\text{-}4$	$-0.1384 \pm 3e\text{-}3$	$-0.2542 \pm 4e\text{-}4$	$\mathbf{-0.1169 \pm 3e\text{-}4}$
	cuptime	$4.6399e2$	$2.5966e2$	$4.8520e2$	$4.8902e2$	$5.9326e4$	**5.1588e2**
	p-value	-	-	$5.1240e\text{-}4$	$6.5857e\text{-}8$	2.7505-6	-

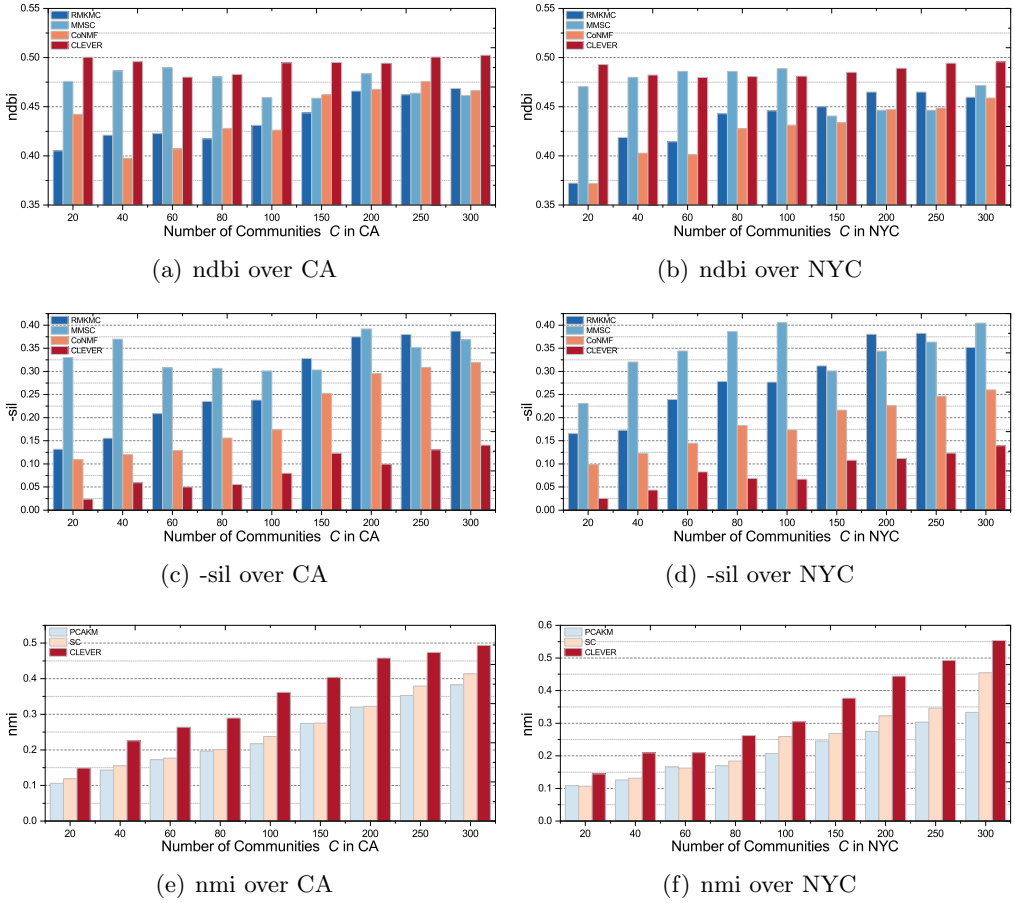
algorithms, which can detect communities over a single network. In this experiment, we separately employed this method on the standard online and offline Laplacian matrices as shown in Eqn.(3.5).

3. **RMKMC**. The novel and robust multi-view k-means clustering method proposed in [20] is able to uncover the consensus pattern and detect communities across multiple networks. We applied it on the dual-networks and obtained a shared clustering result.

4. **MMSC**. This multi-modal spectral clustering method introduced in [21] is to learn a commonly shared graph Laplacian matrix by unifying different views, where each modal stands for a feature type from one single view. We applied it on the dual-networks and obtained a shared clustering result.

5. **CoNMF**. It is a co-regularized nonnegative matrix factorization model presented in [66] by extending NMF for multi-view clustering [98]. It jointly factorizes the multiple matrices through co-regularization. We applied it on the dual-networks and then obtained a shared clustering result.

Table 3.2 summarizes the performance comparison between our model and the baselines in terms of ndbi, nmi, and sil measures. It is worthwhile highlighting that PCAKM, SC, and our model respectively output two partition results corresponding to the online and offline networks; nevertheless the outputs of RMKMC, MMSC, and CoNMF are shared unified partitions across dual-



(a) ndbi over CA

(b) ndbi over NYC

(c) -sil over CA

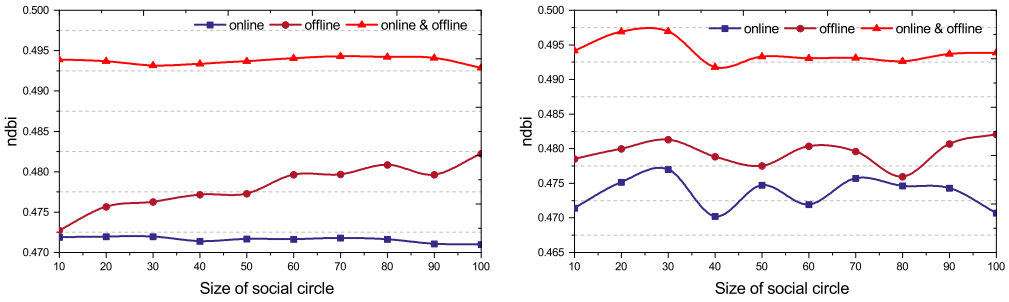
(d) -sil over NYC

(e) nmi over CA

(f) nmi over NYC

Figure 3.5: Performance comparison among various methods by varying the number of communities C. The partition performance over CA and NYC is respectively measured in terms of ndbi, -sil, and nmi metrics.

networks. Due to the output nature, not all the methods can be measured by the three metrics. From Table 3.2, we have the following observations: 1) Regarding the separate partition results over the online and offline networks, CLEVER achieves the best performance in terms of ndbi across two cities. This clearly demonstrates that information encoded in different networks is complementary and is able to reinforce the separate partitions. 2) Our proposed model performs better than PCKAM and SC in term of nmi, which considers all the possible community matching between the online and offline community detection results. This justifies that our local and global consistency constrains are reasonable. 3) Our CLEVER model is superior to RMKMC, MMSC, and CoNMF models in terms of ndbi and sil measures. To compare with them, we linearly fused the community detection results of our model on online and offline networks. Noticeably, we are unable to fuse the clustering results of



(a) CLEVER on CA (b) CLEVER on NYC

Figure 3.6: Performance of CLEVER model in terms of ndbi by varying the social circle size K on CA and NYC. Wherein, ndbi represents the learning performance regarding ndbi over the online, offline, and unified networks, respectively.

PCAKM and SC, since it is hard to align their clusters across dual-networks. Although RMKMC, MMSC, and CoNMF consider the global consistency across the networks, they ignore the local consistency. This further confirms the importance of local consistency in community detection over dual-networks. 4) The partition performance over CA is stably better than NYC, which implies that the online and offline behaviors of CA citizens are more consistent.

In addition, we compared our method with all the baselines by varying the number of communities. As illustrated in Figure 3.5, our model significantly outperforms the other baselines over the two cities *w.r.t.* sil and nmi. Meanwhile, our model can achieve remarkable and comparable performance regarding to ndbi when $C \leq 100$, while showing its superiority with the increasing number of communities C. Besides, we analyzed the sensitivity of our model to the social circle size on the NYC data. The results are demonstrated in Figure 3.6. In this experiment, we found that our model is non-sensitive to the social circle size K. Moreover, we observed that the performance of unified community detection significantly outperforms these separate ones, and this observation reflects the essential importance of unifying the virtual and physical networks.

3.4.5 Component-wise Model Evaluation

The main insight of our proposed CLEVER model is to conduct dual-clustering over dual-networks. Its significant features are to preserve the global consistency of networks and local consistency of social circles, as well as to refine the novel Laplacian matrices. Besides, it enables two flexible community structures, rather

Table 3.3: Effectiveness evaluation of global consistency, local consistency, and refined Laplacian matrix in our proposed CLEVER model over the online and offline networks of Meetup in CA and NYC, when $C = 300$ and $K = 20$. We also provide the variance. Significance test is based on the ndbi over online & offline networks.

Method		Separate	Equal	Global	Local	CLEVER-LM	**CLEVER**
CA	online (ndbi)	$0.4610 \pm 8e\text{-}6$	$0.4669 \pm 6e\text{-}6$	$0.4713 \pm 3e\text{-}5$	$0.4715 \pm 4e\text{-}3$	$0.4643 \pm 1e\text{-}5$	$\mathbf{0.4719 \pm 7e\text{-}6}$
	offline (ndbi)	$0.4621 \pm 6e\text{-}6$	$0.4737 \pm 1e\text{-}5$	$0.4744 \pm 2e\text{-}5$	$0.4738 \pm 6e\text{-}4$	$0.4738 \pm 5e\text{-}6$	$\mathbf{0.4756 \pm 8e\text{-}6}$
	online vs offline (nmi)	$0.4034 \pm 6e\text{-}6$	1	$0.4030 \pm 6e\text{-}6$	$0.4018 \pm 7e\text{-}4$	$0.4810 \pm 4e\text{-}2$	$\mathbf{0.4891 \pm 1e\text{-}3}$
	online & offline (ndbi)	$0.4785 \pm 5e\text{-}6$	$0.4805 \pm 1e\text{-}6$	$0.4894 \pm 1e\text{-}4$	$0.4896 \pm 1e\text{-}5$	$0.4893 \pm 9e\text{-}6$	$\mathbf{0.4937 \pm 6e\text{-}6}$
	online & offline (sil)	$-0.1898 \pm 2e\text{-}5$	$-0.2378 \pm 6e\text{-}5$	$-0.1979 \pm 5e\text{-}4$	$-0.1954 \pm 3e\text{-}4$	$-0.1701 \pm 8e\text{-}4$	$\mathbf{-0.1427 \pm 3e\text{-}3}$
	p-value	3.1136e-6	7.1420e-7	1.3722e-6	3.1496e-6	3.9276e-5	-
NYC	online (ndbi)	$0.4579 \pm 7e\text{-}6$	$0.4720 \pm 1e\text{-}4$	$0.4677 \pm 8e\text{-}6$	$0.4706 \pm 2e\text{-}5$	$0.4655 \pm 8e\text{-}6$	$\mathbf{0.4736 \pm 7e\text{-}6}$
	offline (ndbi)	$0.4616 \pm 2e\text{-}5$	$0.4733 \pm 5e\text{-}5$	$0.4746 \pm 6e\text{-}6$	$0.4751 \pm 2e\text{-}4$	$0.4733 \pm 6e\text{-}6$	$\mathbf{0.4748 \pm 8e\text{-}6}$
	online vs offline (nmi)	$0.4513 \pm 1e\text{-}5$	1	$0.4506 \pm 4e\text{-}4$	$0.4503 \pm 5e\text{-}4$	$0.4889 \pm 6e\text{-}6$	$\mathbf{0.5536 \pm 5e\text{-}3}$
	online & offline (ndbi)	$0.4883 \pm 3e\text{-}5$	$0.4900 \pm 4e\text{-}5$	$0.4905 \pm 5e\text{-}6$	$0.4910 \pm 3e\text{-}5$	$0.4910 \pm 2e\text{-}5$	$\mathbf{0.4940 \pm 7e\text{-}6}$
	online & offline (sil)	$-0.1258 \pm 1e\text{-}6$	$-0.1319 \pm 5e\text{-}5$	$-0.1584 \pm 4e\text{-}6$	$-0.1447 \pm 1e\text{-}4$	$-0.1284 \pm 1e\text{-}5$	$\mathbf{-0.1169 \pm 3e\text{-}4}$
	p-value	1.3872e-7	1.5607e-5	9.2401e-5	7.366e-6	5.1167e-6	-

than ones, over dual-networks. Hence, to investigate the significance of each component, we employed the component-wise model evaluation. In particular, we disabled some terms of our objective function in Eqn.(3.21) as follows,

- **Separate**. To verify that the joint modeling indeed helps, we overlooked all the consensus terms by setting $\lambda_1 = \lambda_2 = 0$. Actually, the method is similar to SC, in which we conduct the spectral clustering over individual networks separately.

- **Equal**. In this method, we verified that the exactly equal community structures somewhat degreases the learning performance due to the overblown consensus. In particular, we set $\mathbf{G}^v = \mathbf{G}^p$, which is equivalent to set $\lambda_2 \to \infty$.

- **Global**. In this method, we only considered the global consistency over dual-networks by setting $\lambda_1 = 0$ in Eqn.(3.21).

- **Local**. In this method, we only took the local consistency over dual-networks into consideration. This is accomplished by setting $\lambda_2 = 0$ in Eqn.(3.21).

- **CLEVER-LM**. To verify the feasibility and efficacy of our refined Laplacian matrix in Eqn.(3.17), we replaced it with the standard Laplacian matrix in Eqn.(3.5).

Table 3.3 displays results of the component-wise evaluation of CLEVER method. From the table, we can make the following observations: 1) No matter

what type of consistency terms we dropped does hurt the performance of our model. This verifies the importance of the global and local consistency over the dual-networks. 2) The Separate method achieves the worst performance. This signals that the joint modeling plays a pivotal role, which can effectively transfer the underlying information between two networks. 3) Meanwhile, the Equal method has poor performance. It verifies that the exactly equal structures somewhat overemphasize the consensus and overlook the flexibility. And 4) CLEVER-LM can obtain the comparable performance regarding ndbi. However, its clustering quality is worse in comparison with CLEVER. This confirms that the refined Laplacian matrices are more suitable than the standard ones. It hence verifies the effectiveness of the exploited manifold structures of social circles. We can hence draw a safe conclusion that jointly modeling the global and local consistency with the refined Laplacian matrices are beneficial to user contextualization and community detection over the dual-networks.

3.5 Application

Aside from the general validation, we also validated our model on a real-world problem: event attendance prediction [98, 54]. Formally, given an event, a small set of early bird, a whole set of attendee candidates, and their online interaction records, we aim to predict who will probably attend this event, and how many users will be present eventually. A good approach to this problem will remarkably facilitate a wide range of event organizers, such as workshop initiators, conference logistics chairs and even the president election committees, by enabling them to design better proactive coping strategies in advance.

It is, however, non-trivial to predict attendance of the given event due to the following reasons: 1) Micro factor. Individual behaviors are often affected by his/her social interactions [42]. To be more specific, if one's close friends confirm to attend the coming event, there exists a better-than-average chance that he/she will show up, too. 2) Macro factor. Communities are usually interest-driven [135]. If the given event matches the taste of one community, the users within the community has higher probabilities of attending this event than those from unmatched communities. For instance, researchers from data mining community prefer participating in the SIGIR conference to the SIGGRAPH

conference. Moreover, traditional classifiers, such as SVM, suffer from insufficient training samples (*i.e.,* usually, only a few early birds are available for a given event), and then they are hard to capture the information propagations among users. We adapted our CLEVER model to solve this problem, since it is able to uncover an better community structure among users.

3.5.1 Experiments

Experimental Settings

Given a happened event e, we can build the ground truth by collecting its real attendees $\mathcal{U} = \{u_1, \ldots, u_{N_e}\}$, where N_e stands for the number of users attended e. To simulate the real case, we randomly selected M attendees from $\mathcal{U}$ to construct the set $\mathcal{M}$, who act as early birds. Based on the partition results of our model, assume that user u_i is assigned into the c-th community which holds a set of members $\mathcal{P}_c$. We can estimate the probability of user u_i in attending the event e by jointly considering the macro and micro factors as follows,

$$\begin{aligned} p(u_i|\mathcal{M}, \mathcal{P}_c) &= p_{macro}(u_i|\mathcal{M}, \mathcal{P}_c) \times p_{micro}(u_i|\mathcal{M}, \mathcal{P}_c) \\ &= \left(\frac{1}{|\mathcal{P}_c|}\frac{|\mathcal{P}_{c,M}|}{|\mathcal{M}|}\right) \times \left(\frac{\sum_{u_j \in \mathcal{P}_{c,M}} sim(u_i, u_j)}{|\mathcal{P}_{c,M}|}\right) \\ &= \frac{\sum_{u_j \in \mathcal{P}_{c,M}} sim(u_i, u_j)}{|\mathcal{P}_c| \times |\mathcal{M}|}, \end{aligned} \tag{3.25}$$

where $\mathcal{P}_{c,M} = \mathcal{P}_c \cap \mathcal{M}$ denotes the attendees from the c-th community, $p_{macro}(u_i|\mathcal{M}, \mathcal{P}_c)$ and $p_{micro}(u_i|\mathcal{M}, \mathcal{P}_c)$ denote the probability of u_i to attend the event affected by the public environment and his/her social connections.

Event Attendance Prediction Evaluation

To measure the effectiveness of event attendance prediction from different angles, we adopted two widely used metrics. One is $F1$ [90, 145, 144]:

$$F1 = 2\frac{precision \cdot recall}{precision + recall}; \tag{3.26}$$

Table 3.4: Performance comparison of event attendance prediction among various methods in term of F1-score by varying the number of early birds, $\{10, \ldots, 50\}$, for the given e. We reported the results with variance on two cities CA and NYC with $C = 300$ and $K = 20$. Significance test is based on the F1-score at $|\mathcal{M}| = 50$

Method		RMKMC	MMSC	CoNMF	Separate	Equal	**CLEVER**
CA	$\|\mathcal{M}\| = 10$	$0.1587 \pm 4e\text{-}4$	$0.1103 \pm 6e\text{-}5$	$0.1372 \pm 9e\text{-}5$	$0.1619 \pm 1e\text{-}5$	$0.1623 \pm 2e\text{-}5$	$\mathbf{0.1832 \pm 7e\text{-}4}$
	$\|\mathcal{M}\| = 20$	$0.2810 \pm 1e\text{-}6$	$0.2023 \pm 6e\text{-}5$	$0.2220 \pm 2e\text{-}4$	$0.2244 \pm 1e\text{-}5$	$0.2462 \pm 4e\text{-}5$	$\mathbf{0.3216 \pm 2e\text{-}3}$
	$\|\mathcal{M}\| = 30$	$0.3290 \pm 5e\text{-}5$	$0.2725 \pm 1\text{-}4$	$0.2737 \pm 4e\text{-}4$	$0.2643 \pm 2e\text{-}5$	$0.4126 \pm 5e\text{-}5$	$\mathbf{0.4252 \pm 3e\text{-}3}$
	$\|\mathcal{M}\| = 40$	$0.3530 \pm 8e\text{-}5$	$0.3208 \pm 2e\text{-}4$	$0.3101 \pm 2e\text{-}4$	$0.3638 \pm 2e\text{-}4$	$0.4874 \pm 7e\text{-}4$	$\mathbf{0.4968 \pm 5e\text{-}3}$
	$\|\mathcal{M}\| = 50$	$0.4719 \pm 6e\text{-}5$	$0.4615 \pm 1e\text{-}4$	$0.4316 \pm 5e\text{-}4$	$0.4289 \pm 2e\text{-}4$	$0.5055 \pm 2e\text{-}4$	$\mathbf{0.5639 \pm 6e\text{-}3}$
	p-value	$3.3881e\text{-}3$	$2.6238e\text{-}5$	$1.0288e\text{-}5$	$2.8812e\text{-}3$	$2.300e\text{-}2$	-
NYC	$\|\mathcal{M}\| = 10$	$0.1501 \pm 2e\text{-}6$	$0.1253 \pm 5e\text{-}5$	$0.1126 \pm 1e\text{-}4$	$0.1515 \pm 6e\text{-}6$	$0.1502 \pm 9e\text{-}6$	$\mathbf{0.1551 \pm 3e\text{-}5}$
	$\|\mathcal{M}\| = 20$	$0.2625 \pm 2e\text{-}4$	$0.2094 \pm 6e\text{-}5$	$0.1936 \pm 1e\text{-}4$	$0.2414 \pm 2e\text{-}5$	$0.2404 \pm 6e\text{-}5$	$\mathbf{0.2743 \pm 2e\text{-}4}$
	$\|\mathcal{M}\| = 30$	$0.3527 \pm 5e\text{-}6$	$0.2512 \pm 4e\text{-}5$	$0.2439 \pm 1e\text{-}4$	$0.3501 \pm 1e\text{-}4$	$0.3841 \pm 1e\text{-}4$	$\mathbf{0.3662 \pm 8e\text{-}4}$
	$\|\mathcal{M}\| = 40$	$0.4125 \pm 5e\text{-}5$	$0.3775 \pm 4\text{-}5$	$0.38801 \pm 9e\text{-}5$	$0.3823 \pm 2e\text{-}4$	$0.4420 \pm 1e\text{-}4$	$\mathbf{0.4498 \pm 1e\text{-}3}$
	$\|\mathcal{M}\| = 50$	$0.4279 \pm 2e\text{-}6$	$0.4104 \pm 2\text{-}5$	$0.4289 \pm 4e\text{-}4$	$0.4064 \pm 4e\text{-}4$	$0.4526 \pm 2e\text{-}4$	$\mathbf{0.5046 \pm 1e\text{-}3}$
	p-value	$3.4562e\text{-}2$	$2.2938e\text{-}8$	$5.4205e\text{-}7$	$5.0421e\text{-}3$	$2.8812e\text{-}3$	-

Table 3.5: Performance comparison among various methods for the number of attendees prediction. For each event listed below, the number of early birds is 20.

	Event ID	# Real Attendees	RMKMC	MMSC	CoNMF	Equal	Seperate	**CLEVER**
CA	1	108	44	20	112	38	37	**107**
	2	85	77	18	127	73	61	**98**
	3	105	69	19	106	63	49	**70**
	4	86	44	12	116	38	50	**81**
	5	82	44	10	150	64	56	**111**
	nMSE	-	0.3344	4.1138	0.1283	0.3668	0.4548	**0.0511**
NYC	6	113	46	44	130	37	49	**106**
	7	119	132	48	189	54	42	**123**
	8	120	138	39	171	65	39	**134**
	9	120	100	42	170	41	49	**164**
	10	116	75	37	196	47	47	**139**
	nMSE	-	0.1223	1.1616	0.1658	0.8374	0.9929	**0.0348**

and the other is normalized mean squared error [179, 69]:

$$nMSE = \frac{\sum_{e=1}^{N}(pred_e - real_e)^2}{N \times \bar{pred} \times \bar{real}}, \tag{3.27}$$

where N, $pred_e$, $real_e$, $\bar{pred}$, and $\bar{real}$ denote the number of testing events, the number of predicted attendees, the number of real attendees for event e, and the average number of predicted and real attendees, respectively.

We chose RMKMC, MMSC, and CoNMF as the baselines, since they output unified partition results over dual-networks. The performance comparison in terms of $F1$ and $nMSE$ are reported in Tables 3.4 and 3.5, respectively. From Table 3.4, we observe that our model stably outperforms the baselines on two cities with different numbers of early birds. This actually reflects that the remarkable improvement of our model in predicting whether a user will attend the given event is much stronger. Namely, our model shows superiority

of predicting who will attend the desired events in most of cases. Table 3.5 illustrates the number of predicted attendees by our model is much more close to the number of real attendees, as compared to other methods. Hence, our model achieves the significant improvements in predicting how many attendees will be present eventually.

3.6 Conclusion and Future Work

In this chapter, we explore the problem of online-offline recommendation. We present a novel dual-clustering model which detects the online and offline communities based on different types of user behaviors, while preserving the local and global consistency constrains across the online-offline networks. Leveraging the augmented communities, we can propagate the user preferences over the communities and further recommend the ongoing events to the users effectively. Extensive experiments are performed over two real-world datasets to demonstrate the performance of community detection and the accuracy of event recommendation. Detailed analysis reveals that incorporating information across heterogeneous channels models the user behaviors well. In the light of this, we aggregated users' footprints across multiple platforms to achieve better performance.

However, this work essentially belongs to the regime of label propagation, which simply assumes that nearby users would have similar preferences on items. It hence is conceptually inferior to model-based methods, since there lacks model parameters to optimize the objective function of recommendation. In future, we consider adopting the idea of deep learning methods and embedding propagation to parameterize users and items.

Chapter 4

Social Recommendation across Information and Social Channels

This chapter investigates the second issue in social recommendation across information-oriented and social-oriented platforms.

4.1 Introduction

Nowadays online platforms play a pivotal role in our daily life and encourage people to share experiences, exchange thoughts, and enjoy online services[1]. Regardless of applications, we can roughly divide the existing platforms into information-oriented and social-oriented domains (or channels)[2]. The former typically refers to forums or E-Commerce sites that have thorough knowledge on items, such as point-of-interests in Trip.com, movies in IMDb, and products in Amazon. These sites have ample user-item interactions available in the form of users' reviews, ratings, along with various kinds of implicit feedback like views and clicks [11]. On the other hand, the social-oriented channels are mainly social network sites, which emphasize the social connections among users [93].

When adopting an item, besides consulting the information sites, a user usually gathers more detailed information from her experienced friends. This refers to *word-of-mouth marketing*, which is widely recognized as the most

[1] `https://goo.gl/iYv78T`.
[2] Here we use "domain" and "channel" interchangeably.

effective strategy for producing recommendation. As reported by Cognizant[3], more than 45% of travelers rely on social networks to seek advice from friends for travel. However, most existing SNSs, like Facebook and Twitter, are designed mainly for users to rebuild their real-world connections, rather than for seeking options regarding items. Though some item cues implying users' preference can be found in SNSs, they typically contain item names only with limited details. The sparse and weak user-item interactions greatly hinder the ability of SNSs to offer item recommendation services.

Fortunately, some users may be simultaneously involved in both SNSs and information-domain sites, who can act as a bridge to propagate user-item interactions across domains. For example, it is not unusual for a user to share her travel experiences in Trip.com; and if the user also holds a Facebook account, we can recommend her friends in Facebook with her liked items from Trip.com. In social circles, these bridge users are like the *silk road* to route relevant items from information domains to (non-bridge) users of social networks. As such, we formulate the task of *cross-domain social recommendation*, which aims to recommend relevant items of information domains to the users of social domains. Apparently, this task is related to the recently emerging topic — cross-domain recommendation [76, 106]. However, we argue that existing efforts have either focused on homogeneous domains (*i.e.,* multiple sites of the information domain) [50, 106], or unrealistically assumed that the users are fully overlapped [76, 166]. Our task to address is particularly challenging due to the following two practical considerations.

- Insufficient bridge users. To gain a deep insight, we collected the users in three cities: Singapore, London, and NewYork City, from Trip.com and obtained their Facebook and Twitter accounts from their public profiles. We analyzed the overlapped users between Trip.com and Facebook/Twitter, and found that only 10.5% of $8,196$ Facebook users and 6.9% of $7,233$ Twitter users have public accounts in Trip.com. It is therefore highly challenging to leverage history of such limited number of bridge users to provide quality recommendation for non-bridge users.

- Rich attributes. The users and items of an information domain are usually

[3] `https://www.cognizant.com`.

associated with rich attributes. For instance, Trip.com enables users to indicate their travel preference explicitly, and associates travel spots (*i.e.,* items) with specific travel modes, among other information. However, little attention has been paid to leverage these attributes to boost the performance of cross-domain recommendation.

In this chapter, we propose a novel solution named *Neural Social Collaborative Ranking* (NSCR) for the new task of cross-domain social recommendation. It is developed based on the recent advance of neural collaborative filtering (NCF) [67], which is further extended to model the cross-domain social relations by combining with the graph regularization technique [64]. We entail two key technical components of our NSCR as follows.

- For the modelling of information domain, we build an attribute-aware recommender based on the NCF framework. To fully exploit the interactions among a user, an item, and their attributes, we enhance NCF by plugging a *pairwise pooling* operation above the embedding vectors of user (item) ID and attributes. In contrast to the default average pooling used by NCF [67] and other recent neural recommenders [41], our use of pairwise pooling better captures feature interactions in the low level [132], greatly facilitating the following deep layers to learn higher-order interactions among users, items and attributes.

- For the modelling of social domain, it is natural to guide the embedding learning of social users by using the embeddings of bridge users. As the embeddings of bridge users are optimized to predict user–item interactions (*e.g.,* ratings and purchases), propagating their embeddings to social users helps to bridge the heterogeneity gap between information domain and social domain. To implement such propagation effect, we employ the *smoothness* constraint (*i.e., graph Laplacian*) on the social network, which enforces close friends to have similar embedding so as to reflect their similar preferences.

To sum up, the key contributions of this work are three-fold:

1. To our knowledge, we are the first to introduce the task of cross-domain social recommendation, which recommends relevant items of information domains to target users of social domains.

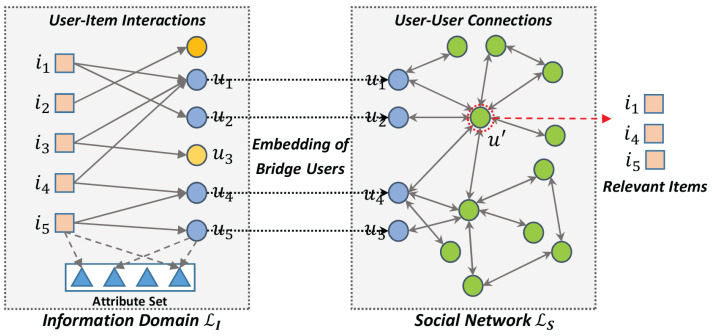


Figure 4.1: Illustration of the cross-domain social recommendation task.

2. We propose a novel solution that unifies the strengths of deep neural networks in modelling attributed user-item interactions and graph Laplacian in modelling user-user social relations.

3. We construct two real-world benchmark datasets for exploring the new task of cross-domain social recommendation and evaluate our proposed solution.

4.2 Preliminary

We first formulate the task of cross-domain social recommendation, and then shortly recapitulate the matrix factorization model, highlighting its limitations for addressing the task.

4.2.1 Problem Formulation

Figure 4.1 illustrates the task of cross-domain social recommendation. In the information domain, we have the interaction data between users and items. Let u and $\mathcal{U}_1 = \{u_t\}_{t=1}^{M_1}$ denote a user and the whole user set of the information domain, respectively; similarly, we use i and $\mathcal{I} = \{i_t\}_{t=1}^{N}$ to denote an item and the whole item set, respectively. The edges between users and items denote their interactions, $\mathcal{Y} = \{y_{ui}\}$, which can be real-valued explicit ratings or binary 0/1 implicit feedback. Traditional collaborative filtering algorithms can then be performed on the user-item interaction data.

In addition to the ID that distinguishes a user or an item, most information-domain sites also associate them with abundant side information, which can help to capture users' preferences and item properties better. For example, in

Trip.com, the user may choose the travel tastes of {*luxury travel, art lover*} in her profile; while, the item *Marina Bay Sands* is tagged most with travel modes {*luxury travel, family travel, nightlife*}. We term these associated information as *attributes*, most of which are discrete categorical variables for the web domain [67, 30]. Formally, we denote g and $\mathcal{G} = \{g_t\}_{t=1}^{V}$ as an attribute and the whole attribute set, respectively; for a user u and an item i, we can then construct the associated attribute set as $\mathcal{G}_u = \{g_1^u, \cdots, g_{V_u}^u\} \subset \mathcal{G}$ and $\mathcal{G}_i = \{g_1^i, \cdots, g_{V_i}^i\} \subset \mathcal{G}$, respectively.

In the social domain, we have social connections between users, such as the undirected friendship or directed follower/followee relations. We denote a social user as u', all users of the social domain as $\mathcal{U}_2 = \{u'_t\}_{t=1}^{M_2}$, and all social connections as $\mathcal{S} = \{s_{u'u''}\}$, that is, the edges between user nodes in the social graph. We define the bridge users as the overlapping users between the information domain and social domain. These bridge users can be expressed as $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2$. In a social network, a user's behaviours and preferences can be propagated along the social connections to influence her friends. As such, these bridge users play a pivotal role in addressing the cross-domain social recommendation problem, which is formally defined as:

Input: An information domain with $\{\mathcal{U}_1, \mathcal{I}, \mathcal{Y}, \mathcal{G}_u, \mathcal{G}_i\}$; a social domain with $\{\mathcal{U}_2, \mathcal{S}\}$; and $\mathcal{U}_1 \cap \mathcal{U}_2$ is nonempty.

Output: A personalized ranking function for each user u' of the social domain $f_{u'} : \mathcal{I} \rightarrow \mathbb{R}$, which maps each item of the information domain to a real number.

It is noted that there indeed exist sparse and weak user-item interactions in SNSs as aforementioned. However, we simplify this scenario of cross-domain social recommendation by only emphasizing the social connections in SNSs and leaving the exploration of weak interactions as the future work.

Our proposed NSCR solution addresses the above limitations of MF as mentioned in Section 2.1.1 by 1) using a deep learning scheme to capture the higher-order correlations between user and item latent factors, and 2) devising a pairwise pooling operation to efficiently model the pair-wise correlations among users, items, and attributes.

4.3 Our NSCR Solution

The goal of cross-domain social recommendation is to select relevant items from the information domain for social users. Under the paradigm of embedding-based methods (*aka.* representation learning), the key for addressing the task is on how to project items (of the information domain) and users (of the social domain) into the same embedding space. A generic solution is the factorization machine (FM) [132, 131], which merges the data from the two domains by an early fusion; that is, constructing the predictive model by incorporating social users as the input features. While the solution sounds reasonable conceptually, the problem is that the training instances which can incorporate social users are only applicable to the bridge users, which can be very few for real-world applications. As such, the generic recommender solution FM can suffer severely from the problem of insufficient bridge users.

To address the challenge of insufficient bridge users, we propose a new framework that separates the embedding learning process of each domain. By enforcing the two learning processes to share the same embeddings for bridge users, we can ensure that items and social users are in the same embedding space. Formally, we devise the optimization framework as:

$$\mathcal{L} = \mathcal{L}_I(\Theta_I) + \mathcal{L}_S(\Theta_S), \tag{4.1}$$

where $\mathcal{L}_I$ (or $\mathcal{L}_S$) denotes the objective function of the information domain (or social domain) learning with parameters Θ_I (or Θ_S), and most importantly, $\Theta_I \cap \Theta_S$ are nonempty denoting the shared embeddings of bridge users.

By separating the learning process for two domains, we allow the design of each component to be more flexible. Specially, we can apply any collaborative filtering solution for $\mathcal{L}_I$ to learn from user-item interactions, and utilize any semi-supervised learning technique for $\mathcal{L}_S$ to propagate the embeddings of bridge users to non-bridge users. In the remainder of this section, we first present our novel neural collaborative ranking solution for $\mathcal{L}_I$, followed by the design of social learning component $\mathcal{L}_S$. Lastly, we discuss how to optimize the joint objective function.

4.3.1 Learning of Information Domain

To estimate the parameters for a CF model from user-item interaction data, two types of objective functions — point-wise [11, 67] and pair-wise [132, 26, 150] — are most commonly used. The point-wise objective functions aim to minimize the loss between the predicted score and its target value. Here, to tailor our solution for both implicit feedback and the personalized ranking task, we adopt the pair-wise ranking objective functions.

Formally, we denote an observed user-item interaction as $y_{ui} = 1$, otherwise $y_{ui} = 0$. Instead of forcing the prediction score $\hat{y}_{ui}$ to be close to y_{ui}, ranking-ware objective functions concern the relative order between the pairs of observed and unobserved interactions:

$$\mathcal{L}_I = \sum_{(u,i,j)\in\mathcal{O}} \mathcal{L}(y_{uij}, \hat{y}_{uij}), \tag{4.2}$$

where $y_{uij} = y_{ui} - y_{uj}$ and $\hat{y}_{uij} = \hat{y}_{ui} - \hat{y}_{uj}$; $\mathcal{O}$ denotes the set of training triplets, each of which comprises of a user u, an item i of observed interactions (*i.e.,* $y_{ui} = 1$), and an item j of unobserved interactions (*i.e.,* $y_{ui} = 0$). An ideal model should rank all (i, j) item pairs correctly for every user. To implement the ranking hypotheses, we adopt the regression-based loss [150, 91]:

$$\mathcal{L}_I = \sum_{(u,i,j)\in\mathcal{O}} (y_{uij} - \hat{y}_{uij})^2 = \sum_{(u,i,j)\in\mathcal{O}} (\hat{y}_{ui} - \hat{y}_{uj} - 1)^2. \tag{4.3}$$

Note that other pair-wise ranking functions can also be applied, such as the bayesian personalized ranking (BPR) [26, 132] and contrastive max-margin loss [143]. In this chapter, we use the regression-based ranking loss as a demonstration for our NSCR, and leave the exploration of other choices as the future work.

Attribute-aware Deep CF Model

Having established the optimization function for learning from information domain, we now present our attribute-aware deep collaborative filtering model to estimate a user-item interaction $\hat{y}_{ui}$. Figure 4.2 illustrates its architecture, which is a multi-layered feed-forward neural network. We elaborate its design

layer by layer.

Input Layer. The input to the model is a user u, an item i, and their associated attributes $\mathcal{G}_u$ and $\mathcal{G}_i$. We transform them into sparse vectors with one-hot encoding, where only the non-zero binary features are recorded.

Embedding Layer. The embedding layer encodes ID information of each user u, item i, user attribute g_t^u, and item attribute g_t^i with one-hot encoding, and then parameterizes them as a vectorized representations: $\mathbf{u}$, $\mathbf{i}$, $\mathbf{g}_t^u$, and $\mathbf{g}_t^i \in \mathbb{R}^d$, where d is the embedding size. Such embeddings are parts of model parameters.

Pooling Layer. The output of the embedding layer is a set of embedding vectors to describe user u and item i, respectively. As different users (items) may have different number of attributes, the size of the embedding vector set may vary for different inputs. To train a neural network of fixed structure, it is essential to convert the set of variable-length vectors to a fixed-length vector, *i.e.,* the pooling operation.

The most commonly used pooling operations in neural network modelling are average pooling and max pooling. However, we argue that such simple operations are insufficient to capture the interaction between users/items and attributes. For example, the average pooling assumes a user and her attributes are linearly independent, which fails to encode any correlation between them in the embedding space. To tackle the problem, we consider to model the pairwise correlation between a user and her attributes, and all nested correlations among her attributes:

$$\mathbf{p}_u = \varphi_{pairwise}(\mathbf{u}, \{\mathbf{g}_t^u\}) = \sum_{t=1}^{V_u} \mathbf{u} \odot \mathbf{g}_t^u + \sum_{t=1}^{V_u} \sum_{t'=t+1}^{V_u} \mathbf{g}_t^u \odot \mathbf{g}_{t'}^u, \tag{4.4}$$

where $\odot$ denotes the element-wise product of two vectors. We term it as *pairwise pooling*, which is originally inspired from the design of factorization machines [130, 65]. By applying pairwise pooling on the item counterpart, we can similarly model the pair-wise correlation between an item and its attributes:

$$\mathbf{q}_i = \varphi_{pairwise}(\mathbf{i}, \{\mathbf{g}_t^i\}) = \sum_{t=1}^{V_i} \mathbf{i} \odot \mathbf{g}_t^i + \sum_{t=1}^{V_i} \sum_{t'=t+1}^{V_i} \mathbf{g}_t^i \odot \mathbf{g}_{t'}^i. \tag{4.5}$$

It is worth pointing out that although pairwise pooling models the correlation

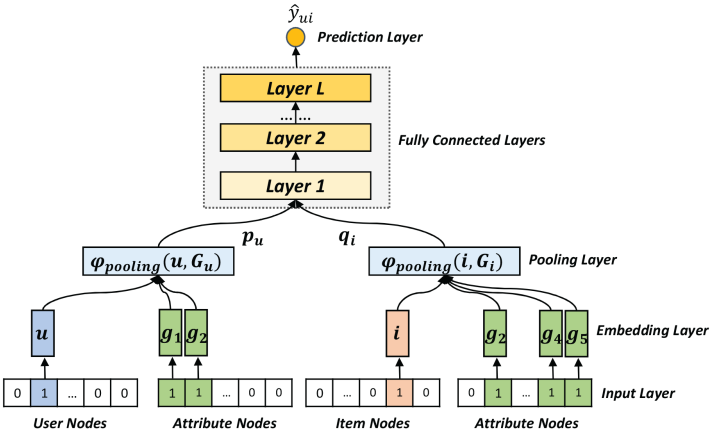


Figure 4.2: Illustration of our Attributed-aware Deep CF model for estimating an user-item interaction.

between each pair of features, it can be efficiently computed in linear time — the same time complexity with average/max pooling. To show the linear time complexity of evaluating pairwise pooling, we reformulate Eqn.(4.4) as,

$$\mathbf{p}_u = \frac{1}{2}\left[(\mathbf{u} + \sum_{t=1}^{V_u} \mathbf{g}_t^u) \odot (\mathbf{u} + \sum_{t=1}^{V_u} \mathbf{g}_t^u) - \mathbf{u} \odot \mathbf{u} - \sum_{t=1}^{V_u} \mathbf{g}_t^u \odot \mathbf{g}_t^u\right], \tag{4.6}$$

which can be computed in $O(KV_u)$ time. This is a very appealing property, meaning that the benefit of pairwise pooling in modelling all pair-wise correlations does not involve any additional cost, as compared to the average pooling that does not model any correlation between input features.

Hidden Layers: Above the pairwise pooling is a stack of full connected layers, which enable us to capture the nonlinear and higher-order correlations among users, items, and attributes. Inspired by the neural network view of matrix factorization (*cf.* Figure 2.1), we first merge user representation $\mathbf{p}_u$ and item representation $\mathbf{q}_i$ with an element-wise product, which models the two-way interaction between u and i. We then place a multi-layer perceptron (MLP) above the element-wise product. Formally, the hidden layers are defined as:

$$\begin{cases} \mathbf{e}_1 = \sigma_1(\mathbf{W}_1(\mathbf{p}_u \odot \mathbf{q}_i) + \mathbf{b}_1) \\ \mathbf{e}_2 = \sigma_2(\mathbf{W}_2\mathbf{e}_1 + \mathbf{b}_2) \\ \cdots\cdots \\ \mathbf{e}_L = \sigma_L(\mathbf{W}_L\mathbf{e}_{L-1} + \mathbf{b}_L) \end{cases}, \tag{4.7}$$

where $\mathbf{W}_l$, $\mathbf{b}_l$, σ_l, and $\mathbf{e}_l$ denote the weight matrix, bias vector, activation

function, and output vector of the l-th hidden layers, respectively. As for the activation function in each hidden layer, we opt for Rectifier (ReLU) unit, which is more biologically plausible and proven to be non-saturated. Regarding the structure of hidden layers, common choices include the tower [67, 41], constant, and diamond, among others. In this work, we simply set all hidden layers have the same size, leaving the further tuning of the deep structure as the future work.

Prediction Layer: At last, the output vector of the last hidden layer $\mathbf{e}_L$ is transformed to the prediction score:

$$\hat{y}_{ui} = \mathbf{w}^\top \mathbf{e}_L, \tag{4.8}$$

where $\mathbf{w}$ represents the weight vector of the prediction layer.

Note that we have recently proposed a neural factorization machine (NFM) model [65], which similarly uses a pairwise pooling operation to model the interaction among features. We point out that the main architecture difference is in our separated treatment of the user and item channel, where each channel can essentially be seen as an application of NFM on the user/item ID and attributes.

4.3.2 Learning of Social Domain

With the above neural collaborative ranking solution, we obtain an attribute-aware representation $\mathbf{p}_u$ and $\mathbf{q}_i$ for each user and item, respectively. To predict the affinity score of a social user to an item of the information domain, we need to also learn an representation for the social user in the same latent space of the information domain. We achieve this goal by propagating $\mathbf{p}_u$ from bridge users to representations for non-bridge users of the social domain. The intuition for such representation propagation is that, if two users are strongly connected (*e.g.,* close friends with frequent interactions), it is likely that they have the similar preference on items; as such, they should have similar representations in the latent space. This suits well the paradigm of graph regularization [64, 53] (*aka.* semi-supervised learning on graph), which has two components:

Smoothness: The smoothness constraint implies the structural consistency — the nearby vertices of a graph should not vary much in their representations. Enforcing smoothness constraint in our context of social domain learning will propagate a user's representation to her neighbors, such that when a steady

state reaches, all vertices should have been placed in the same latent space. The objective function for smoothness constraint is defined as:

$$\theta(\mathcal{U}_2) = \frac{1}{2} \sum_{u',u'' \in \mathcal{U}_2} s_{u'u''} \left\| \frac{\mathbf{p}_{u'}}{\sqrt{d_{u'}}} - \frac{\mathbf{p}_{u''}}{\sqrt{d_{u''}}} \right\|^2, \tag{4.9}$$

where $s_{u'u''} \in \mathcal{S}$ denotes the strength of social connection between user u' and user u'' in the social domain, and $d_{u'}$ (or $d_{u''}$) denotes the outdegree of u' (or u'') for normalization purpose. It is worth noting that the use of normalization is the key difference with the social regularization used by [102], which does not apply any normalization on the smoothness constraint. As pointed out by He *et al.* [64], the use of normalization helps to suppress the impact of popular vertices, which can lead to more effective propagation. We empirically verify this point in Section 4.3.

Fitting: The fitting constraint implies the latent space consistency across two domains — the bridge users' representations should be invariant and act as the anchors across domains. Towards this end, we encourage the two representations of the same bridge users to be close to each other. The objective function for fitting constraint is defined as,

$$\theta(\mathcal{U}) = \frac{1}{2} \sum_{u' \in \mathcal{U}} \left\| \mathbf{p}_{u'} - \mathbf{p}_{u'}^{(0)} \right\|^2, \tag{4.10}$$

where for each bridge user u', $\mathbf{p}_{u'}$ (or $\mathbf{p}_{u'}^{(0)}$) is her representation of the SNS (or information domain). As such, the fitting constraint essentially acts as the bridges connecting the two latent spaces.

Lastly, we combine the smoothness constraint with the fitting constraint and obtain the objective function of the social domain learning as,

$$\mathcal{L}_S = \theta(\mathcal{U}_2) + \mu\theta(\mathcal{U}), \tag{4.11}$$

where μ is a positive parameter to control the tradeoff between two constraints.

Prediction for Social Users

With the representations of social users and items (*i.e.,* $\mathbf{p}_{u'}$ and $\mathbf{q}_i$) at hand, we can feed them into the fully connected layers as Eqn.(4.7) shows and utilize

the prediction layer as Eqn.(4.8) displays. At last, we can obtain the predicted preference $\widehat{y}_{u'i}$, as follows,

$$\begin{cases} \mathbf{e}_1 = \sigma_1(\mathbf{W}_1(\mathbf{p}_{u'} \odot \mathbf{q}_i) + \mathbf{b}_1) \\ \cdots\cdots \\ \mathbf{e}_L = \sigma_L(\mathbf{W}_L\mathbf{e}_{L-1} + \mathbf{b}_L) \\ \widehat{y}_{u'i} = \mathbf{w}^\top \mathbf{e}_L \end{cases} . \tag{4.12}$$

4.3.3 Training

We adopt the alternative optimization strategy on Eqn.(4.1) since it can emphasize exclusive characteristics within individual domains. In the information domain, we employ stochastic gradient descent SGD) to train the attribute-aware NSCR in the mini-batch mode and update the corresponding model parameters. In particular, we first sample a batch of observed user-item interactions (u, i) and adopt negative sampling [67] to randomly select an unobserved item j for each (u, i). We then generate a triplet (u, i, j). Following that, we take a gradient step to optimize the loss function $\mathcal{L}_I$ in Eqn.(4.3). As such, we obtain the enhanced representations of users. In the SNS, we feed the enhanced representations of bridge users into our graph Laplacian to update all representations of social users. Towards this end , we can simplify the derivative of $\mathcal{L}_S$ regarding user representation $\mathbf{P}$ and then obtain the close-form solution as,

$$\mathbf{P} = \frac{\mu}{1+\mu}\left(\mathbf{I} - \frac{1}{1+\mu}\mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}}\right)^{-1}\mathbf{P}^{(0)}, \tag{4.13}$$

where $\mathbf{P}^{(0)}$ is the embedding of social users, which includes the updated representations of bridge users from NSCR part; $\mathbf{S}$ and $\mathbf{D}$ are the similarity matrix and diagonal degree matrix of social users, respectively, whereinto $S_{u'u''} = s_{u'u''}$ and $D_{u'u'} = d_{u'}$. Thereafter, we view the newly updated representations of bridge users as the next initialization for the bridge users in NSCR. We repeat the above procedures to approximate the model parameter set Θ. As for the regularization term in Eqn.(4.1), we omit it since we utilize *dropout* technique in neural network modeling to avoid overfitting.

Dropout: Dropout is an effective solution to prevent deep neural networks

from overfitting. The idea is to randomly drop part of neurons during training. As such, only part of the model parameters, which contribute to the final ranking, will be updated. In our neural CR model, we propose to adopt dropout on the pairwise pooling layer. In particular, we randomly drop ρ of $\mathbf{p}_u$ and $\mathbf{q}_i$, whereinto ρ is the dropout ratio. Analogous to the pooling layer, we also conduct dropout on each hidden layer.

4.4 Experiments

To comprehensively evaluate our proposed method, we conducted experiments to answer the following research questions:

- **RQ1:** Can our NSCR approach outperform the state-of-the-art recommendation methods for the new cross-domain social recommendation task?
- **RQ2:** How do different hyper-parameter settings (*e.g.,* the dropout ratio and tradeoff parameters) affect NSCR?
- **RQ3:** Are deeper hidden layers helpful for learning from user-item interaction data and improving the performance of NSCR?

4.4.1 Data Description

To the best of our knowledge, there is no available public benchmark dataset that fits the task of cross-domain social recommendation. As such, we constructed the datasets by ourselves. We treated Trip.com as the information domain, Facebook and Twitter as the social domains. In Trip.com, we initially compiled $6,532$ active users, who had at least 5 ratings over $2,952$ items (*e.g., gardens by the bay* in Singapore and *eiffel tower* in Pairs). We transformed their $93,998$ ratings into binary implicit feedback as ground truth, indicating whether the user has rated the item. Moreover, we collected 19 general categories regarding the travel mode (*e.g., adventure travel*, *business travel*, and *nightlife*) and used them as the attributes of users and items. Subsequently, we parsed the users' profiles to identify their aligned accounts in Facebook and Twitter, inspired by the methods in [114, 145]. We obtained 858 and 502 bridge users for Facebook and Twitter, respectively. Thereafter, we crawled the public friends or followers of each bridge user to reconstruct the social networks, resulting in $177,042$ Facebook users and

106,049 Twitter users. However, the original social data are highly sparse, where most non-bridge users have only one friend, making it ineffective to propagate users' preferences. To ensure the quality of the social data, we performed a modest filtering on the data, retaining users with at least two friends. This results in a subset of the social data that contains 7,233 Twitter users with 42,494 social connections and 8,196 Facebook users with 49,156 social connections. The statistics of the datasets are summarized in Table 4.1.

4.4.2 Experimental Settings

Evaluation Protocols: Given a social user, each method generates an item ranking list for the user. To assess the ranking list, we adopted two popular IR metrics, *AUC* and *recall*, to measure the quality of preference ranking and top-N recommendation.

- **AUC:** Area under the curve (AUC) [132, 72] measures the probability that a recommender system ranks a positive user-item interaction higher than negative ones:

$$AUC = \frac{\sum_{i \in \mathcal{I}_u^+} \sum_{j \in \mathcal{I}_u^-} \delta(\widehat{y}_{uij} > 0)}{|\mathcal{I}_u^+||\mathcal{I}_u^-|}, \tag{4.14}$$

  where $\mathcal{I}_u^+ = \{i|y_{ui} = 1\}$ and $\mathcal{I}_u^- = \{j|y_{uj} = 0\}$ denote the sets of relevant (observed) item i and irrelevant (unobserved) item j for user u, respectively; and δ is the count function returning 1 if $\widehat{y}_{uij} > 0$ and 0 otherwise. Below we report the averaged AUC for all testing users.

- **R@K:** Recall@K considers the relevant items within the top K positions of the ranking list. A higher recall with lower K indicates a better recommender system, which can be defined as,

$$R@K = \frac{|\mathcal{I}_u^+ \cap \mathcal{R}_u|}{|\mathcal{I}_u^+|}, \tag{4.15}$$

  where $\mathcal{R}_u$ denotes the set of the top-K ranked items for the given user u. Analogous to AUC, we report the average $R@5$ for all testing users.

By learning representations for social users and information-domain items together, our NSCR is capable of recommending items for both bridge and

Table 4.1: Statistics of the complied datasets. The social user set includes the bridge users.

Information Domain	User#	Item#	Interaction#
Trip.com	6,532	2,952	93,998
SNSs	**Bridge User#**	**Social User#**	**Social Connection#**
Twitter	502	7,233	42,494
Facebook	858	8,196	49,156

non-bridge users. However, due to the limitation of our static datasets, it is difficult for us to evaluate the recommendation quality for non-bridge users, since they have no interaction on the information-domain items. As such, we rely on the bridge users for evaluating the performance. Following the common practice in evaluating a recommender algorithm [67, 132], we holdout the latest 20% interactions of a bridge user as the test set. To tune hyper-parameters, we further randomly holdout 20% interactions from a bridge user's training data as the validation set. We feed the remaining bridge users, all the non-bridge users in SNSs, and the remaining user-item interactions in the information domains into our framework for training.

Baselines: To justify the effectiveness of our proposal, we study the performance of the following methods:

- **ItemPop:** This method ranks items base on their popularity, as judged by the number of interactions. It is a non-personalized method that benchmarks the performance of a personalized system [132].

- **MF:** This is the standard matrix factorization model that leverages only user–item interactions of the information domain for recommendation (*cf.* Eqn.(2.2)).

- **SFM:** Factorization machine [130] is a generic factorization model that is designed for recommendation with side information. We construct the input feature vector by using one-hot encoding on the ID and attributes of users and items. To adjust FM for modelling social relations, we further plug a (bridge) user's friends into the input feature vector, dubbed this enhanced model as Social-aware FM (SFM).

- **SR:** This [102] is a state-of-the-art factorization method for social recommendation. It leverages social relations to regularize the latent vectors of friends to be similar. To incorporate attributes into their method, we adjust the similarity of two users based on their attribute

Table 4.2: Performance comparison between all the methods, when the embedding size= 64 and signicance test is based on AUC.

Datasets	Twitter-Trip			Facebook-Trip		
Methods	**AUC**	**R@5**	**p-value**	**AUC**	**R@5**	**p-value**
ItemPop	0.7193	0.0164	3e-5	0.7439	0.0249	8e-6
MF	0.8285	0.0375	3e-4	0.8596	0.0821	1e-4
SFM	0.8832	0.0492	2e-3	0.8908	0.0856	1e-3
SR	0.9013	0.0747	9e-3	0.9267	0.1433	4e-2
NSCR	**0.9222**	**0.0807**	-	**0.9390**	**0.1466**	-

sets, which leads to better performance.

Note that for all model-based methods, we optimize them with the same pair-wise ranking function of Eqn.(4.3) for a fair comparison on the model's expressiveness. To explore the efficacy of attributes, we further explore variants that remove attribute modelling from SFM, SR, and NSCR, named as SFM-a, SR-a, and NSCR-a, respectively.

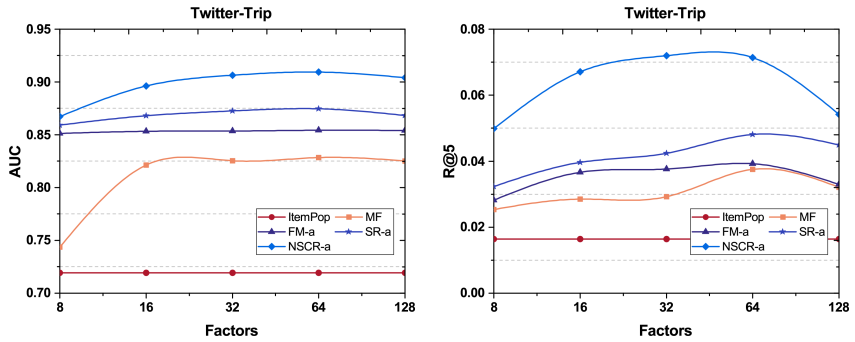
Parameter Settings: We implemented our proposed framework on the basis of Tensorflow[4], which will be made publicly available, as well as our datasets. For all the neural methods, we randomly initialized model parameters with a Gaussian distribution, whereinto the mean and standard deviation is 0 and 0.1, respectively. The mini-batch size and learning rate for all methods was searched in $[128, 256, 512, 1024]$ and $[0.0001, 0.0005, 0.001, 0.05, 0.1]$, respectively. We selected Adagrad as the optimizer. Moreover, we empirically set the size of hidden layer same as the embedding size (the dimension of the latent factor) and the activation function as ReLU. Without special mention, we employed two hidden layers for all the neural methods, including SFM, SR, and NSCR. We randomly generated ten different initializations and feed them into our NSCR. For other competitors, the initialization procedure is analogous to ensure the fair comparison. Thereafter, we performed paired t-test between our model and each of baselines over 10-round results.

4.4.3 Performance Comparison (RQ1)

We first compare the recommendation performance of all the methods. We then purpose to justify how the social modelling and the attribute modelling affect the recommendation performance.

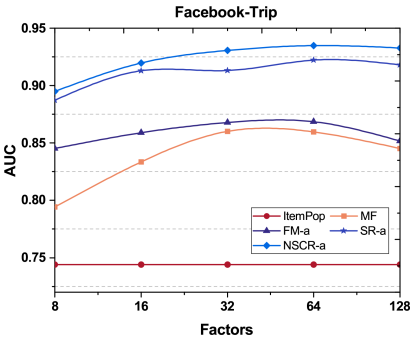
Overall Comparison: Table 4.2 displays the performance comparison *w.r.t.* AUC and R@5 among the recommendation methods on Twitter-Trip

[4]`https://www.tensorflow.org`.

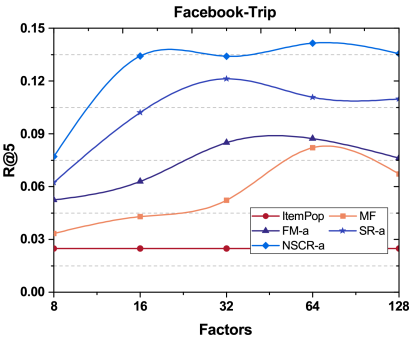


(a) AUC on Twitter-Trip

(b) R@5 on Twitter-Trip



(c) AUC on Facebook-Trip



(d) R@5 on Facebook-Trip

Figure 4.3: Performance comparison of AUC and R@5 *w.r.t.* the embedding size on Twitter-Trip and Facebook-Trip datasets.

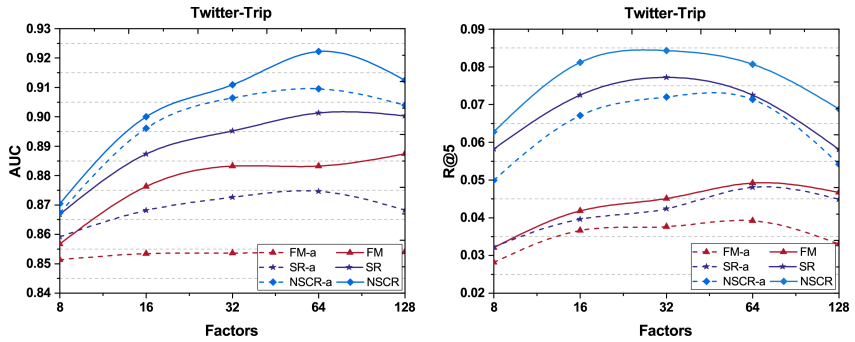
and Facebook-Trip datasets, where the embedding size is 64 for all the methods. We have the following findings:

- ItemPop achieves the worst performance, indicating the necessity of modelling users' personalized preferences, rather than just recommending popular items to users. As for MF, its unsatisfied performance reflects that the independence assumption is insufficient to capture the complex and non-linear structure of user-item interactions.

- NSCR substantially outperforms the state-of-the-art methods, SFM and SR. We further conduct one-sample t-tests, verifying that all improvements are statistically significant with p-value < 0.05. It justifies the effectiveness of our proposed framework.

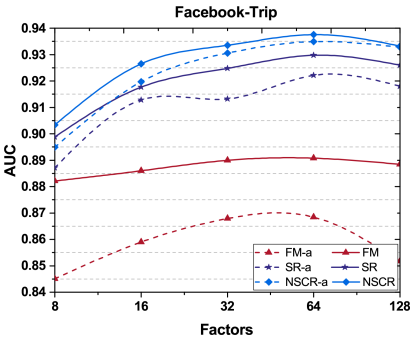
- The performance on Twitter-Trip clearly underperforms that of Facebook-Trip. It is reasonable since more bridge users are available in Facebook, which can lead to better embedding learning in SNSs. It again verifies the significance of the bridge users.

Effect of Social Modelling: To analyze the effect of social modelling, we

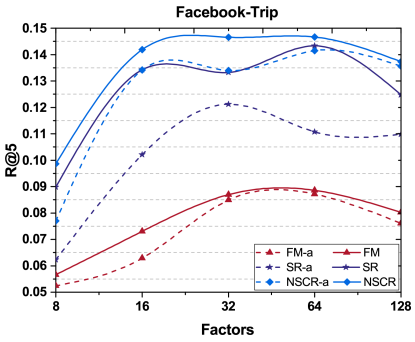


(a) AUC on Twitter-Trip

(b) R@5 on Twitter-Trip



(c) AUC on Facebook-Trip



(d) R@5 on Facebook-Trip

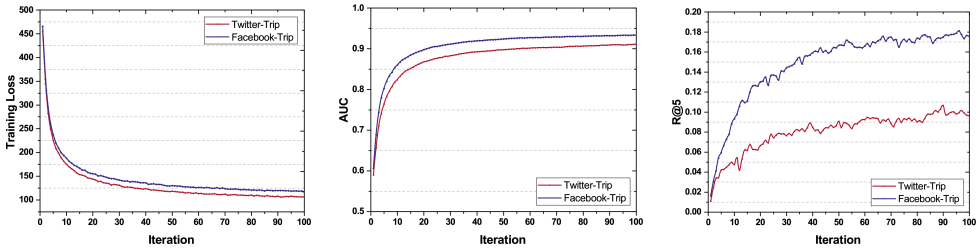
Figure 4.4: Performance comparison of AUC and R@5 *w.r.t.* the embedding size on Twitter-Trip and Facebook-Trip datasets.

only consider the variants, SFM-a, SR-a, and NSCR-a. Figure 4.3 presents the performance comparison *w.r.t.* the number of latent factors on two datasets. We have the following observations.

- ItemPop and MF perform worst since neither of them considers the social connections from SNSs. It highlights the necessity of social modelling in cross-domain social recommendation.

- Clearly, NSCR-a significantly outperforms SFM-a and SR-a by a large margin. Formally, in terms of AUC, the relative improvement over SFM-a and SR-a, on average, is 3.19% and 1.01% respectively. While SFM-a considers modelling the social connections, it treats these connections as ordinary features, overlooking the exclusive characteristics of social networks. This leads to the poor expressiveness of the social users' embedding. On the contrary, SR-a and NSCR-a emphasizes the social modelling via the effective social regularization.

- Lastly, NSCR-a shows consistent improvements over SR-a, admitting the importance of the normalized graph Laplacian. It again verifies that the



(a) Training Loss (b) AUC (c) R@5

Figure 4.5: Training loss and recommendation performance regarding AUC and R@5 *w.r.t.* the number of iterations.

normalized graph Laplacian can suppress the popularity of friends and further prevent the social modelling from being dominated by popular social users.

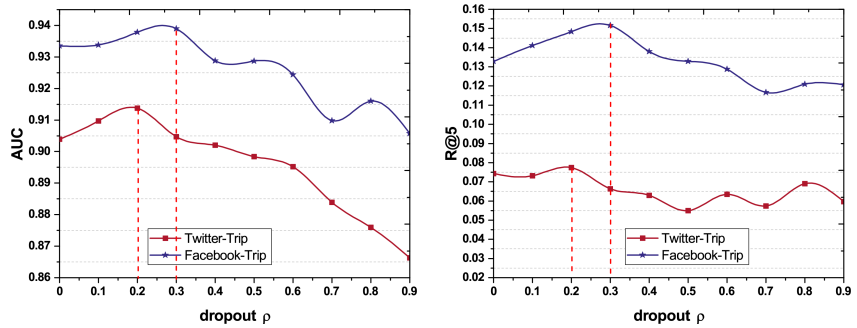
Effect of Attribute Modelling: As Figure 4.4 demonstrates, we verify the substantial influence of attribute modelling and the effectiveness of our pairwise pooling operation. Due to the poor performance of ItemPop and MF, they are omitted. Jointly analyzing the performance of all the methods and their variants, we find that,

- For all methods, modelling user/item attributes can achieve significant improvements. By leveraging the similarity of users' attributes, SR enriches the pairwise similarity of any two users and strengthens their connections; meanwhile, SFM can model the correlations of user-attribute, item-attribute, and attribute-attribute, and accordingly enhances the user-item interactions. Benefiting from the pairwise pooling operation, NSCR can encode the second-order interactions between user/item and attributes and boost the representation learning.

- Varying the embedding size, we can see that large embedding may cause overfitting and degrade the performance. In particular, the optimal embedding size is 64 and 32 for AUC and R@5, respectively. It indicates that the setting of embedding size can effect the expressiveness of our model.

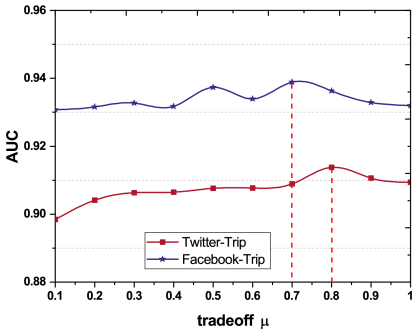
4.4.4 Study of NSCR (RQ2)

In this subsection, we empirically study the convergence of NSCR and then purpose to analyse the influences of several factors, such as dropout ratio and

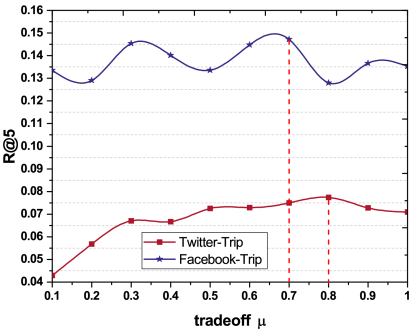


(a) AUC vs. dropout ratio ρ

(b) R@5 vs. dropout ratio ρ



(c) AUC vs. tradeoff parameter μ



(d) R@5 vs. tradeoff parameter μ

Figure 4.6: Performance comparison of AUC and R@5 *w.r.t.* the dropout ratio ρ and tradeoff parameter μ on Twitter-Trip and Facebook-Trip datasets.

tradeoff parameter, on our framework.

Convergence: We separately present the training loss and the performance w.r.t. AUC and R@5 of each iteration in Figures 4.5(a), 4.5(b), and 4.5(c). Jointly observing these Figures, we can see that training loss of NSCR gradually decreases with more iterations, whereas the performance is generally improved. This indicates the rationality of our learning framework. Moreover, the most effective updates occurs in the first 20 iterations, which indicates that effectiveness of our learning framework. As Figure 4.5(c) shows, the performance regarding R@5 fluctuates markedly over the iteration times, while that regarding AUC is quite stable. It is reasonable since R@5 only considers the top-5 results rather than the relative order as AUC defined.

Impact of Dropout: We employ the dropout technique in NSCR to prevent our model from overfitting, instead of regularizing model parameters. Figures 4.6(a) and 4.6(b) present the performance *w.r.t.* AUC and R@5 of NSCR-0 by varying the dropout ratio ρ on the pairwise pooling layer, respectively. As we can see, when dropout ratio equals to 0, NSCR-0 suffers severely from overfitting. Moreover, using a dropout ratio of 0.3 and 0.2

Table 4.3: Recommendation performance of NSCR with different hidden layers.

Metrics	AUC			R@5		
Factors	**NSCR-0**	**NSCR-1**	**NSCR-2**	**NSCR-0**	**NSCR-1**	**NSCR-2**
Twitter-Trip						
8	0.8598	0.8630	**0.8704**	0.0585	0.0604	**0.0628**
16	0.8883	0.8984	**0.9026**	0.0738	0.0672	**0.0812**
32	0.9018	0.9056	**0.9109**	0.0723	0.0742	**0.0843**
64	0.9138	0.9175	**0.9222**	0.0717	0.0697	**0.0725**
128	0.9003	0.9034	**0.9125**	0.0519	0.0653	**0.0688**
Facebook-Trip						
8	0.8978	0.8922	**0.9034**	0.0860	0.0872	**0.0986**
16	0.9165	0.9197	**0.9265**	0.1048	0.1388	**0.1419**
32	0.9303	0.9322	**0.9335**	0.1441	**0.1486**	0.1465
64	0.9337	0.9376	**0.9390**	0.1353	0.1359	**0.1466**
128	0.9270	0.9310	**0.9332**	0.1168	0.1304	**0.1373**

leads to the best performance on Twitter-Trip and Facebook-Trip datasets, respectively. However, when the optimal dropout ratio exceeds the optimal settings, the performance of NSCR-0 greatly decreases, which suffers from insufficient information. This highlights the significance of using dropout, which can be seen as ensembling multiple sub-models [147].

Impact of Tradeoff Parameter: There is one positive parameter μ in the social modelling, which can capture the tradeoff between the fitting regularizer and the normalized graph Laplacian, as Eqn.(4.13) shows. Figures 4.6(c) and 4.6(d) present the performance *w.r.t.* . AUC and R@5, respectively. As we can see, setting μ of 0.8 and 0.7 can lead to the optimal performance on Twitter-Trip and Facebook-Trip datasets, respectively. And the performance of NSCR-0 changes within small ranges nearby the optimal settings. It justifies that our model is relatively insensitive to the parameter around its optimal configuration.

4.4.5 Impact of Hidden Layer (RQ3)

To capture the complex and non-linear inherent structure of user-item interactions, we employ the a deep neural network for our task. It is curious whether NSCR can benefit from the deep architecture. Towards this end, we further investigate NSCR with different number of hidden layers. As it is computationally expensive to tune the dropout ratio ρ for each hidden layer, we simply apply the same settings for all layers. The empirical results on two datasets are summarized in Table 4.3 whereinto NSCR-2 indicates the NSCR method with two hidden layers (besides the embedding layer and prediction layer), and similar notations for others. We have the following observations:

- In most cases, stacking more hidden layers is helpful for the recommendation performance. NSCR-2 and NSCR-1 achieve consistent improvement over NSCR-0, which has no hidden layers and directly projects the embedding to the prediction layer. We attributed the improvement to the high nonlinearity achieved by stacking more hidden layers. Our finding is consistent with [63] and again verifies the deep neural networks have strong generalization ability. However, it is worth mentioning that such a deep architecture needs more time to optimize our framework and easily leads to the overfitting due to the limited training data in our datasets.

- Increasing the width of hidden layers (*i.e.,* the embedding size) from 8 to 64 can improve the performance significantly, as that of increasing their depth. However, with the embedding size of 128, NSCR degrades the performance. It again verifies that using a large number of the embedding size has powerful representation ability [63], but may adversely hurt the generalization of the model (*e.g.,* overfitting the data) [67, 63].

4.5 Conclusion

In this chapter, we systematically investigated cross-domain social recommendation, a practical task that has rarely been studied previously. Towards this end, we proposed a generic neural social collaborative ranking (NSCR) solution, which seamlessly integrates user-item interactions of the information domain and user-user social relations of the social domain. To validate our solution, we constructed two real-world benchmarks of the travel domain, performing extensive experiments to demonstrate the effectiveness and rationality of our NSCR solution. The key finding of the work is that social signals contain useful cues about users' preference, even if the social signals are from social networks in a different domain. We achieved the goal by leveraging bridge users to unify the relevance signals from the two heterogeneous domains.

Due to our restricted resources in collecting cross-domain data, the result is preliminary. Here we discuss several limitations of the current work, and our plans to address them in future. First, in this chapter, we studied the recommendation performance of a travel-based information domain only, which

is mainly for the ease of accessing the users' account on Facebook/Twitter. This results in a relatively small number of bridge users of our cross-domain datasets. As a future work, we will collect a larger-scale set of data from the more popular information domains, such as E-commence sites, to explore the generalization ability of our solution to other information domains. Second, due to the small number of bridge users, we leave the study of user cold-start problem as the extensive of our work, as further holding out bridge users to simulate the cold-start scenario will pose challenge to the stability of evaluation. With a larger-scale cross-domain data, we will study the effectiveness of our solution for cold-start users, as well as the influence of the bridge users' percentage. Moreover, we restricted the SNSs by emphasizing only the social connections and omitting the weak user-item interactions in user-generated-contents. We will consider the weak user-item interaction in both domains to improve the recommendation performance.

Chapter 5

Explainable Recommendation using User and Item Channels

This chapter focuses on the third issue in explainable recommendation. Explainability of a recommender system is defined in terms of the system being able to reveal the features that contribute most to a decision. Here we treat the feature combinations across user-centric and item-centric channels, termd feature crossing, as the explanations.

5.1 Introduction

Personalized recommendation is at the core of many online customer-oriented services, such as E-commerce, social media, and content-sharing websites. Technically speaking, the recommendation problem is usually tackled as a matching problem, which aims to estimate the relevance score of a user for an item based on their available profiles. Regardless of the application domain, a user's profile usually consists of an ID (to identify which specific user) and some side information like age, gender, and income level. Similarly, an item's profile typically contains an ID and some attributes like category, tags, and price. We denote these information as user-centric and item-centric channels, respectively.

Collaborative filtering (CF) is the most prevalent technique for building a personalized recommendation system [81, 67]. It leverages users' interaction histories on items to select the relevant items for a user. From the matching view, CF uses the ID information only as the profile for a user and an

item, and forgoes other side information. As such, CF can serve as a generic solution for recommendation without requiring any domain knowledge. However, the downside is that it lacks necessary reasoning or explanations for a recommendation. Specially, the explanation mechanisms are either *because your friend also likes it* (*i.e.,* user-based CF [70]) or *because the item is similar to what you liked before* (*i.e.,* item-based CF), which are too coarse-grained and may be insufficient to convince users on a recommendation [155, 166, 36].

To persuade users to perform actions on a recommendation, we believe it is crucial to provide more concrete reasons in addition to similar users or items. For example, we recommend *iPhone 7 Rose Gold* to user *Emine*, because we find females aged 20-25 with a monthly income over $\$10,000$ (which are *Emine*' demographics) generally prefer Apple products of pink color. To supercharge a recommender system with such informative reasons, the underlying recommender shall be able to (i) **explicitly** discover effective cross features[1] from the rich side information of users and items, and (ii) estimate user-item matching score in an **explainable** way. In addition, we expect the use of side information will help in improving the performance of recommendation.

Nevertheless, none of existing recommendation methods can satisfy the above two conditions together. In the literature, embedding-based methods such as matrix factorization [81, 133, 69] is the most popular CF approach, owing to the strong power of embeddings in generalizing from sparse user-item relations. Many variants have been proposed to incorporate side information, such as factorization machine (FM) [130], Neural FM [65], Wide&Deep [32], and Deep Crossing [139]. While these methods can learn feature interactions from raw data, we argue that the cross feature effects are only captured in a rather implicit way during the learning process; and most importantly, the cross features cannot be explicitly presented [139]. Moreover, existing works on using side information have mainly focused on the cold-start issue [11], leaving the explanation of recommendation relatively less touched.

In this chapter, we aim to fill the research gap by developing a recommendation solution that is both accurate and explainable. By **accurate**, we expect our method to achieve the same level of performance as existing

[1]In the example of *Emine*, the expected cross feature is over five fields, {gender, age, income level, product company, product color}.

embedding-based approaches [130, 139]. By **explainable**, we would like our method to be transparent in generating a recommendation and is capable of identifying the key cross features for a prediction. Towards this end, we propose a novel solution named *Tree-enhanced Embedding Method* (TEM), which combines embedding-based methods with decision tree-based approaches. First, we build a *gradient boosting decision trees* (GBDT) on the side information of users and items to derive effective cross features. We then feed the cross features into an embedding-based model, which is a carefully designed neural attention network that reweights the cross features according to the current prediction. Owing to the explicit cross features extracted by GBDTs and the easy-to-interpret attention network, the overall prediction process is fully transparent and self-explainable. Particularly, to generate reasons for a recommendation, we just need to select the most predictive cross features based on their attention scores.

As a main technical contribution, this work presents a new scheme that unifies the strengths of embedding-based and tree-based methods for recommendation. Embedding-based methods are known to have strong generalization ability [32, 65], especially in predicting the unseen crosses on user ID and item ID (*i.e.,* capturing the CF effect). However, when operating on the rich side information, embedding-based methods lose the important property of explainability — the cross features that contribute most to the prediction cannot be revealed. On the other hand, tree-based methods predict by generating explicit decision rules, making the resultant cross features directly interpretable. While such a way is highly suitable for learning from side information, it fails to predict unseen cross features, thus being unsuitable for incorporating user ID and item ID. To build an explainable recommendation solution, we combine the strengths of embedding-based and tree-based methods in a natural and effective manner, which to our knowledge has never been studied before.

5.2 Related Work

We can roughly divide explanation styles into similarity-based and content-based categories. The similarity-based methods [1, 2] present explanations as a list of most similar users or items. For example, Behnoush *et al.* [1] used Restricted Boltzmann Machines to compute the explainability scores of the items in the

top-K recommendation list. While the similarity-based explanation can serve as a generic solution for explaining a CF recommender, the drawback is that it lacks concrete reasoning.

Content-based works have considered various side information, ranging from item tags [154, 157], social relationships [140, 129], contextual reviews written by users [103, 182, 47, 129, 34] to knowledge graphs [174, 23, 5].

Item Tags. To explain a recommendation, the work [157] considered the matching between the relevant tags of an item and the preferred tags of the user.

Social Relations. Considering the user friendships in social networks, [140] proposed a generative model to investigate the effects of social explanations on user preferences.

Contextual Reviews. Zhang *et al.* [182] developed an explicit factor model, which incorporated user sentiments *w.r.t.* item aspects as well as the user-item ratings, to facilitate generating aspect-based explanations. Similarly, He *et al.* [64] extracted item aspects from user reviews and modeled the user-item-aspect relations in a hybrid collaborative filtering model. More recently, Ren *et al.* [129] involved the viewpoints, a tuple of user sentiment and item aspect, and trusted social relations in a latent factor model to boost recommendation performance and present personalized viewpoints as explanations.

Knowledge Graphs. Knowledge graphs show great potential on explainable recommendation. Yu *et al.* [174] introduced a meta-path-based factor model that paths learned from an information graph can enhance the user-item relations and further provide explainable reasoning. Recently, Alashkar *et al.* [5] integrated domain knowledge represented as logic-rules with the neural recommendation method.

Despite the promising attempts achieved, most previous works treat the extracted feature (*e.g.,* item aspect, user sentiment, or relationship) as an individual factor in factor models, same as the IDs. As such, little attention has been paid to discover the effects of cross features (or feature combinations) explicitly.

In terms of techniques, existing works have also considered combining tree-based and embedding-based models, among which the most popular method is boosting [28, 96, 186]. These solutions typically perform a late fusion on

the prediction of two kinds of models. GB-CENT proposed in [186] composes of embedding and tree components to achieve the merits of both models. Particularly, GB-CENT achieves CF effect by conducting MF over categorical features; meanwhile, it employs GBDT on the supporting instances of numerical features to capture the nonlinear feature interactions. Ling *et al.* [96] shows that boosting neural networks with GBDT achieves the best performance in the CTR prediction. However, these boosting methods only fuse the outputs of different models and may be insufficient to fully propagate information between tree-based and embedding-based models. Distinct from the previous works, our TEM treats the cross features extracted from GBDT as the input of embedding-based model, facilitating the information propagation between two models. More importantly, the main focus of TEM is to provide explanations for a recommendation, rather than only for improving the performance.

5.3 Preliminary

We first review the embedding-based model, discussing its difficulty in supporting explainable recommendation. We then introduce the tree-based model and emphasize its explanation mechanism.

5.3.1 Embedding-based Model

Embedding-based model is a typical example of representation learning [14], which aims to learn features from raw data for prediction, as MF and FM introduced in Section 2.1.1.

With the recent advances of deep learning, neural network methods have also been employed to build embedding-based models [32, 139, 65]. Specially, Wide&Deep [32] and Deep Crossing [139] learn feature interactions by placing a multi-layer perceptron (MLP) above the concatenation of the embeddings of nonzero features; the MLP is claimed to be capable of learning any-order cross features. Neural FM (NFM) [65] first applies a bilinear interaction pooling on feature embeddings (*i.e.,* $\sum_{t=1}^{n}\sum_{j=t+1}^{n} x_t\mathbf{v}_t \odot x_j\mathbf{v}_j$) to learn second-order feature interactions, followed by a MLP to learn high-order features interactions.

Despite the strong representation ability of existing embedding-based methods in modeling side information, we argue that they are not suitable

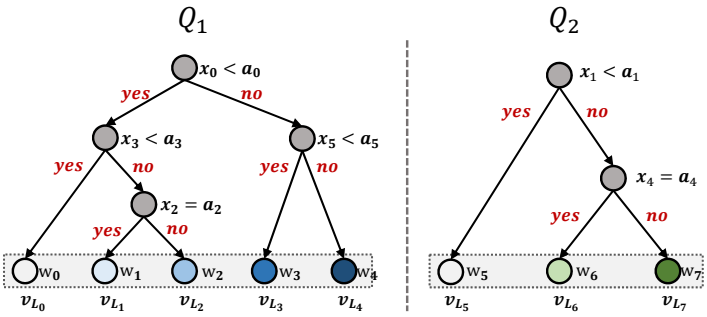


Figure 5.1: An example of a GBDT model with two subtrees.

for providing explanations. FM models second-order feature interactions only and cannot capture high-order cross feature effects; moreover, it uniformly considers all second-order interactions and cannot distinguish which interactions are more important for a prediction [169]. While neural embedding models are able to capture high-order cross features, they are usually achieved by a nonlinear neural network above feature embeddings. The neural network stacks multiple nonlinear layers and is theoretically guaranteed to fit any continuous function [71], however, the fitting process is opaque and cannot be explained. To the best of our knowledge, there is no way to extract explicit cross features from the neural network and evaluate their contributions to a prediction.

5.3.2 Tree-based Model

In contrast to representation learning methods, tree-based models do not learn features for prediction. Instead, they perform prediction by learning decision rules from data. We represent the structure of a tree model as $Q = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ denote the nodes and edges, respectively. The nodes in $\mathcal{V}$ have three types: the root node v_0, the internal (*aka.* decision) nodes $\mathcal{V}_T$, and the leaf nodes $\mathcal{V}_L$. Figure 5.1 illustrates an example of a decision tree model. Each decision node v_t splits a feature x_t with two decision edges: for numerical feature (*e.g.,* time), it chooses a threshold a_j and splits the feature into $[x_t < a_j]$ and $[x_t \geq a_j]$; for binary feature (*e.g.,* features after one-hot encoding on a categorical variable), it determines whether the feature equals to a value or not, *i.e.,* the decision edges are like $[x_t = a_j]$ and $[x_t \neq a_j]$.

A path from the root node to a leaf node forms a *decision rule*, which can also be seen as a *cross feature*, such as in Figure 5.1 the leaf node v_{L_2} represents $[x_0 < a_0]\&[x_3 \geq a_3]\&[x_2 \neq a_2]$. Each leaf node v_{L_i} has a value w_i, denoting the

prediction value of the corresponding decision rule. Given a feature vector $\mathbf{x}$, the tree model first determines which leaf node $\mathbf{x}$ falls on, and then takes the value of the leaf node as the prediction: $\hat{y}_{DT}(\mathbf{x}) = w_{Q(\mathbf{x})}$, where Q maps the feature vector to the leaf node based on the tree structure. We can see that under such a prediction mechanism, the leaf node can be regarded as the most prominent cross feature for the prediction. As such, the tree-based model is self-interpretable by nature.

As one single tree may not be expressive enough to capture complex patterns in data, a more widely used solution is to build a forest, such as gradient boosting decision trees (GBDT) which boosts the prediction by leveraging multiple additive trees:

$$\hat{y}_{GBDT}(\mathbf{x}) = \sum_{s=1}^{S} \hat{y}_{DT_s}(\mathbf{x}), \tag{5.1}$$

where S denote the number of additive trees, and $\hat{y}_{DT_s}$ denotes the predictive model for the s-th tree. We can see that GBDT extracts S rules to predict the target value of a given feature vector, whereas a single tree model predicts based on one rule. As such, GBDT usually leads to better accuracy than a single tree model [55, 18].

While tree-based models are effective in generating interpretable predictions from rich side features, they suffer from generalizing to unseen feature interactions. As such, tree-based models cannot be used for collaborative filtering which needs to model the sparse ID features of users and items.

We can see that the pros and cons of embedding-based and tree-based models complement each other, in terms of generalization ability and interpretability. Hence, to build an effective and explainable recommender systems, a natural solution is to combine the two types of models.

5.4 Tree-enhanced Embedding Method

We first present our tree-enhanced embedding method (TEM) that unifies the strengths of MF for sparse data modeling and GBDTs for cross feature learning. We then discuss the explainability and scrutability and analyze the time complexity of TEM.

5.4.1 Predictive Model

Given a user u, an item i, and their feature vectors $[\mathbf{x}_u, \mathbf{x}_i] = \mathbf{x} \in \mathbb{R}^n$ as the input, TEM predicts the user-item preference as,

$$\hat{y}_{TEM}(u, i, \mathbf{x}) = b_0 + \sum_{t=1}^{n} b_t x_t + f_\Theta(u, i, \mathbf{x}), \tag{5.2}$$

where the first two terms model the feature biases similar to that of FM, and $f_\Theta(u, i, \mathbf{x})$ is the core component of TEM with parameters Θ to model the cross feature effect, which is shown in Figure 5.2. In what follows, we elaborate the design of f_Θ step by step.

Constructing Cross Features.

Instead of embedding-based methods that capture the cross feature effect opaquely during the learning process, our primary consideration is to make the cross features explicit and explainable. A widely used solution in industry is to manually craft cross features, and then feed them into an interpretable method that can learn the importance of each cross feature, such as logistic regression. For example, we can cross all values of feature variables *age* and *traveler style* to obtain the second-order cross features like *[age≥ 18] & [traveler style=friends]*. However, the difficulty of such method is that it is not scalable. For modeling higher-order feature interactions, one has to cross multiple feature variables together, resulting in exponential increase in complexity. With a large space of billions of features, even performing feature selection [161] is highly challenging, not to mention learning from them. Although through careful feature engineering such as crossing important variables or values only [32], one can control the complexity to a certain extent, it requires extensive domain knowledge to develop an effective solution and is not easily domain-adaptable.

To avoid such labor-intensive feature engineering, we leverage the GBDT (briefed in Section 5.3.2), to automatically identify useful cross features. While GBDT is not specially designed for extracting cross features, considering that a leaf node represents a cross feature and the trees are constructed by optimizing predictions on historical interactions, it is reasonable to think that the leaf nodes are useful cross features for prediction.

Formally, we denote a GBDT as a set of decision trees, $\mathcal{Q} = \{Q_1, \cdots, Q_S\}$, where each tree maps a feature vector $\mathbf{x}$ to a leaf node (with a weight); we use L_s to denote the number of leaf nodes in the s-th tree. Distinct from the original GBDT that sums over the weights of activated leaf nodes as the prediction, we keep the activated leaf nodes as cross features, feeding them into a neural attention model for more effective learning. We represent the cross features as a multi-hot vector $\mathbf{q}$, which is a concatenation of multiple one-hot vectors (where a one-hot vector encodes the activated leaf node of a tree):

$$\mathbf{q} = GBDT(\mathbf{x}|\mathcal{Q}) = [Q_1(\mathbf{x}), \cdots, Q_S(\mathbf{x})]. \tag{5.3}$$

Here $\mathbf{q}$ is a sparse vector, where an element of value 1 indicates an activated leaf node and the number of nonzero elements in $\mathbf{q}$ is S. Let the size of $\mathbf{q}$ be $L = \sum_s L_s$. For example, in Figure 5.1, there are two subtrees Q_1 and Q_2 with 5 and 3 leaf nodes, respectively. If $\mathbf{x}$ ends up with the second and third leaf node of Q_1 and Q_2, respectively, the resultant multi-hot vector $\mathbf{q}$ should be $[0, 1, 0, 0, 0, 0, 0, 1]$. Let the semantics of feature variables (x_0 to x_5) and values (a_0 to a_5) of Figure 5.1 be listed in Table 5.1, then $\mathbf{q}$ implies the two cross features extracted from $\mathbf{x}$:

1. v_{L_1}: *[Age< 18] & [Country≠France] & [Restaurant Tag= French].*
2. v_{L_7}: *[Expert Level≥ 4] & [Traveler Style≠Luxury Traveler].*

Prediction with Cross Features.

With the explicit cross features, we can employ sparse linear methods to learn the importance of each cross feature, and select the top cross features as the explanation for a prediction. The prior work by Facebook [68] has demonstrated the effectiveness of such a solution, which feeds the leaf nodes of a GBDT into a logistic regression (LR) model. We term this solution as *GBDT+LR*. Although GBDT+LR is capable of learning the importance of cross features, it assigns a cross feature the same weight for predictions of all user-item pairs, which limits the modeling fidelity. In real applications, it is common that users with similar demographics may choose similar items, but they are driven by different intents or reasons.

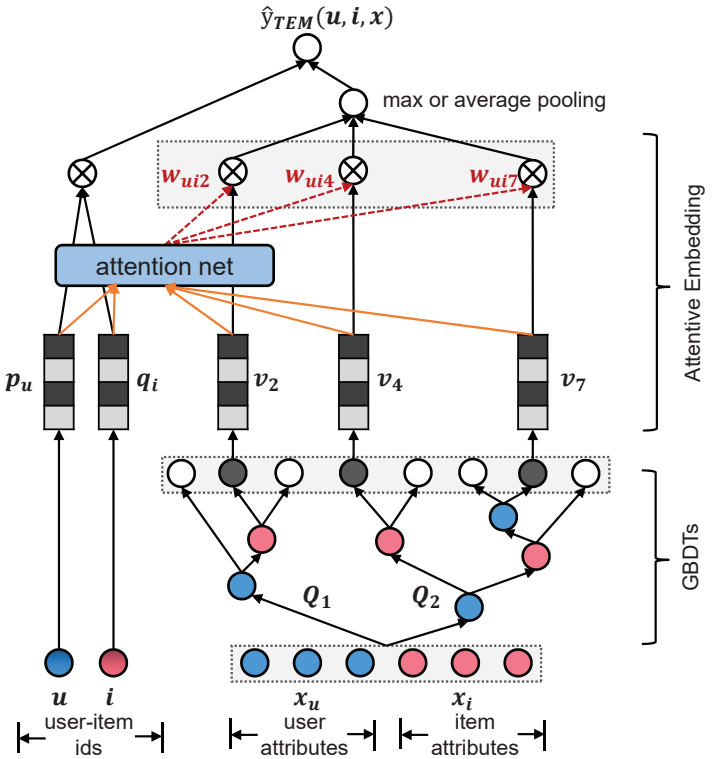


Figure 5.2: Illustrative architecture of our TEM framework.

As an example, let $(u, i, \mathbf{x})$ and $(u', i', \mathbf{x}')$ be two positive instances (*i.e.,* observed user-item interactions, such as purchases and clicks). Assuming $\mathbf{x}$ equals to $\mathbf{x}'$, then the two instances will have the same cross features from GBDT. Since each cross feature has a global weight independent of the training instance in LR, the predictions of (u, i) and (u', i') will be interpreted as the same top cross features, regardless of the possibility that the actual reasons behind u chose i and u' chose i' are different. To ensure the expressiveness, we believe it is important to score the cross features differently for different user-item pairs, *i.e.,* personalizing the weights on cross features rather than using a global weighting mechanism.

Recent advances on neural recommender models such as Wide&Deep [32] and NFM [65] can allow personalized importance on cross features. This is achieved by embedding user ID, item ID, and cross features together into a shared embedding space, and then performing nonlinear transformations (*e.g.,* by fully connected layers) on the embedding vectors. The strong representation power of nonlinear hidden layers enables complicated interactions among user ID, item ID, and cross features to be captured. As such, a cross feature can impact differently when predicting with different user-item pairs. However, such

Table 5.1: The semantics of feature variables and values of the GBDT model in Figure 5.1.

$x_0 \leftarrow$ *Age*	$x_1 \leftarrow$ *Expert Level*	$x_2 \leftarrow$ *Restaurant Tag*
$a_0 \leftarrow 18$	$a_1 \leftarrow 4$	$a_2 \leftarrow$ *French*
$x_3 \leftarrow$ *Country*	$x_4 \leftarrow$ *Traveler Style*	$x_5 \leftarrow$ *Price*
$a_3 \leftarrow$ *France*	$a_4 \leftarrow$ *Luxury Traveler*	$a_5 \leftarrow$ *$$$$*

methods cannot interpret the personalized weights of cross features, due to the hardly explainable nonlinear hidden layers. As such, for explainability purpose we have to discard the use of fully connected hidden layers, although they are helpful to a model's performance in existing methods.

To develop a method that is both effective and explainable, we introduce two essential ingredients of our TEM — embedding and attention. Specifically, we first associate each cross feature with an embedding vector, allowing the correlations among cross features to be captured. We then devise an attention mechanism to explicitly model the personalized weights on cross features. Lastly, the embeddings of user ID, item ID, and cross features are integrated together for the final prediction. The use of embedding and attention endows TEM strong representation ability and guarantees the effectiveness, even though it is a shallow model without any fully connected hidden layer. In what follows, we elaborate the two key ingredients of TEM.

Embedding. Given the cross feature vector $\mathbf{q}$ generated by GBDT, we project each cross feature j into an embedding vector $\mathbf{v}_j \in \mathbb{R}^k$, where k is the embedding size. After the operation, we obtain a set of embedding vectors $\mathcal{V} = \{q_1\mathbf{v}_1, \cdots, q_L\mathbf{v}_L\}$. Since $\mathbf{q}$ is a sparse vector with only a few nonzero elements, we only need to include the embeddings of nonzero features for a prediction, *i.e.,* $\mathcal{V} = \{\mathbf{v}_l\}$ where $q_l \neq 0$. We use $\mathbf{p}_u$ and $\mathbf{q}_i$ to denote the user embedding and item embedding, respectively.

There are two advantages of embedding the cross features into a vector space, compared to LR that uses a scalar to weight a feature. First, learning with embeddings can capture the correlations among features, *e.g.,* frequently co-occurred features may yield similar embeddings, which can alleviate the data sparsity issue. Second, it provides a means to seamlessly integrate the output of GBDT with the embedding-based collaborative filtering, being more flexible than a late fusion on the model predictions (*e.g.,* boosting GBDT with FM as used in [186]).

Attention. Inspired by the previous work [169, 26], we explicitly capture the varying importance of cross features on prediction by assigning an attentive weight for the embedding of each cross feature. Here we consider two ways to aggregate the embeddings of cross features, average pooling and max pooling, to obtain a unified representation $\mathbf{e}(u, i, \mathcal{V})$ for cross features:

$$\begin{cases} \mathbf{e}_{avg}(u, i, \mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{v}_l \in \mathcal{V}} w_{uil} \mathbf{v}_l, \\ \mathbf{e}_{max}(u, i, \mathcal{V}) = max\_pool_{\mathbf{v}_l \in \mathcal{V}}(w_{uil} \mathbf{v}_l), \end{cases} \tag{5.4}$$

where w_{uil} is a trainable parameter denoting the attentive weight of the l-th cross feature in constituting the unified representation, and importantly, it is personalized to be dependent with (u, i).

While the above solution seems to be sound and explainable, the problem is that for (u, i) pairs that have never co-occurred before, the attentive weight w_{uil} cannot be estimated. In addition, the parameter space of w is too large — there are UIL weights in total (where U, I, and L denote the number of users, items, and the size of $\mathbf{q}$, respectively), which is impractical to materialize for real-world applications. To address the generalization and scalability issues, we consider modeling w_{uil} as a function dependent on the embeddings of u, i, and l, rather than learning w_{uil} freely from data. Inspired by the recent success [169, 7, 26] that uses multi-layer perceptrons (MLPs) to learn the attentive weights, we similarly use a MLP to parameterize w_{uil}. We call the MLP as the *attention network*, which is defined as:

$$\begin{cases} w'_{uil} & = \mathbf{h}^\top ReLU(\mathbf{W}([\mathbf{p}_u \odot \mathbf{q}_i, \mathbf{v}_l]) + \mathbf{b}) \\ w_{uil} & = \frac{exp(w'_{uil})}{\sum_{(u,i,\mathbf{x}) \in \mathcal{O}} exp(w'_{uil})} \end{cases}, \tag{5.5}$$

where $\mathbf{W} \in \mathbb{R}^{a \times 2k}$ and $\mathbf{b} \in \mathbb{R}^a$ denote the weight matrix and bias vector of the hidden layer, respectively, and a controls the size of the hidden layer. The vector $\mathbf{h} \in \mathbb{R}^a$ projects the hidden layer into the attentive weight for output. We used the rectifier as the activation function and normalized the attentive weights using softmax. Figure 5.3 illustrates the architecture of our attention network, and we term a as the *attention size*.

Final Prediction. Having established the attentive embeddings, we obtain

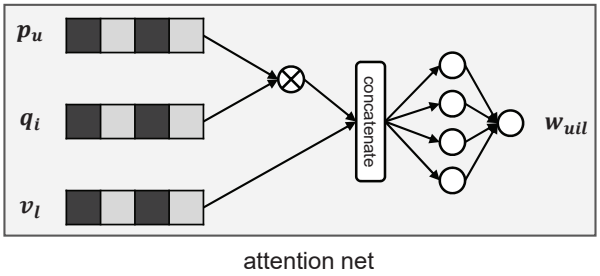


Figure 5.3: Illustration of the attention network in TEM.

a unified embedding vector $\mathbf{e}(u, i, \mathcal{V})$ for cross features. To incorporate the CF modeling, we concatenate $\mathbf{e}(u, i, \mathcal{V})$ with $\mathbf{p}_u \odot \mathbf{q}_i$, which reassembles MF to model the interaction between user ID and item ID. We then apply a linear regression to project the concatenated vector to the final prediction. This leads to the predictive model of our TEM as:

$$\hat{y}_{TEM}(u, i, \mathbf{x}) = b_0 + \sum_{t=1}^{m} b_t x_t + \mathbf{r}_1^\top (\mathbf{p}_u \odot \mathbf{q}_i) + \mathbf{r}_2^\top \mathbf{e}(u, i, \mathcal{V}), \tag{5.6}$$

where $\mathbf{r}_1 \in \mathbb{R}^k$ and $\mathbf{r}_2 \in \mathbb{R}^k$ are the weights of the final linear regression layer. As can be seen, our TEM is a shallow and additive model. To interpret a prediction, we can easily evaluate the contribution of each component. We use **TEM-avg** and **TEM-max** to denote the TEM that uses $\mathbf{e}_{avg}(\cdot)$ and $\mathbf{e}_{max}(\cdot)$, respectively, and discuss their explanation schemes in Section 5.4.3.

5.4.2 Learning

Similar to the recent work on neural collaborative filtering [67], we solve the item recommendation task as a binary classification problem. Specifically, an observed user-item interaction is assigned to a target value 1, otherwise 0. We optimize the pointwise log loss, which forces the prediction score $\hat{y}_{ui}$ to be close to the target y_{ui}:

$$\mathcal{L} = \sum_{(u,i,\mathbf{x}) \in \mathcal{O}} -y_{ui} \log \sigma(\hat{y}_{ui}) - (1 - y_{ui}) \log \left(1 - \sigma(\hat{y}_{ui})\right) + \lambda \left\|\Theta\right\|_2^2, \tag{5.7}$$

where σ is the activation function to restrict the prediction to be in $(0, 1)$, set as sigmoid $\sigma(x) = 1/(1 + e^{-x})$ in this chapter; L_2 regularization with λ and Θ is conducted to avoid overfitting, where Θ is the set of model parameters.

The regularization terms are omitted here for clarity (we tuned the L_2

regularization in experiments when overfitting was observed). Note that optimizing other objective functions are also technically viable, such as the pointwise regression loss [65] and ranking loss [132, 26, 165]. In this chapter, we use the log loss as a demonstration of our TEM.

Since TEM consists of two cascaded models, both them are trained to optimize the same log loss. We first train the GBDT, which greedily fits additive trees on the whole training data [27]. After obtaining the cross features from GBDT, we optimize the embedding-based prediction model using the mini-batch Adagrad [49]. Since considering all negaitve instances (*i.e.,* unobserved user-item interactions)is is really expensive and will slow down the training, we adopt the negative sampling [67]. That is, each mini-batch contains stochastic positive instances and randomly paired negative ones. Same as the optimal setting of [67], we pair one positive instance with four negative instances, which empirically shows good performance.

5.4.3 Discussion

Explainability & Scrutability

The two pooling methods as defined in Equation (5.4) aggregate the embeddings of cross features differently, resulting in different explanation mechanisms for TEM-avg and TEM-max. Specifically, the average pooling linearly combines all embeddings, with each embedding a weight to denote its importance. As such, the w_{uil} of $\mathbf{e}_{avg}(u, i, \mathcal{V})$ can be directly used to select top cross features (*i.e.,* decision rules) as the explanation of a prediction [169, 7]. In contrast, the max pooling is a nonlinear operator, where the d-th dimension of $\mathbf{e}_{max}(u, i, \mathcal{V})$ is set to be that of the l-th cross feature embedding with the maximum $w_{uil}v_{ld}$. As such, at most k cross feature embeddings will contribute to the unified representation[2], and we can treat the max pooling as performing feature selection on cross features in the embedding space. To select top cross features for explanation, we need to track the embeddings of which cross features contribute most during the max pooling, rather than simply relying on w_{uil}. We conduct a case study on explainability of TEM in Section 5.5.4.

Empowered by the transparency in generating a recommendation, TEM

[2]Typically, the embedding size k is smaller than the number of trees S in GBDT.

allows the recommender to be scrutable [155]. If a user is unsatisfied with a recommendation due to improper reasons, TEM allows a user to correct the reasoning process to obtain refreshed recommendations. As Equation (5.6) shows, we can easily obtain the contribution of each cross feature on the final prediction, *e.g.,* $y_{uil} = w_{uil}\mathbf{r}_2^\top \mathbf{v}_l$ for TEM-avg. When getting feedback from a user (*i.e.,* the signals indicating what she likes or not), we can localize the cross features that contain the signals, and then modify the corresponding attentive weights. As such, we can refresh the predictions and re-rank the recommendation list without re-training the whole model. We use a case study to demonstrate the scrutability of TEM in Section 5.5.4.

Time Complexity Analysis

As we separate the learning procedure into two phases, we can calculate the computational costs step by step. Generally, the time complexity of building a GBDT model is $O(SD \left\|\mathbf{x}\right\|_0 \log n)$, where S is the number of trees, D is the maximum depth of trees, n is the number of training instances, and $\left\|\mathbf{x}\right\|_0$ denotes the average number of non-zero entries in the training instances. Moreover, we can speed up the greedy algorithm in GBDT by using the block structure like XGBoost [27].

For the embedding component, calculating the attention score for each (u, i, l) costs time $O(2ak)$, where a and k are the attention and embedding size, respectively. Accordingly, adopting the pooling operation for each (u, i) costs $O(2akS)$. As such, to train the embedding model of TEM over n training instances, the complexity is $O(2akSn)$. Therefore, the overall time complexity for training TEM from scratch is $O(SD \left\|\mathbf{x}\right\|_0 \log n + 2akSn)$.

5.5 Experiments

As the key contribution of the work is to generate accurate and explainable recommendations, we conduct experiments to answer the following questions:

1. **RQ1**: Compared with the state-of-the-art recommendation methods, can our TEM achieve comparable accuracy?

2. **RQ2**: Can TEM make the recommendation results easy-to-interpret by

using cross features and the attention network?

3. **RQ3**: How do different hyper-parameter settings (*e.g.,* the number of trees and embedding size) affect TEM?

5.5.1 Data Description

We collect data from two populous cities in TripAdvisor[3], London (LON) and New York City (NYC), and separately perform experiments of tourist attraction and restaurant recommendation. We term the two datasets as LON-A and NYC-R respectively. In particular, we crawl $1,001$ tourist attractions (*e.g., British Museum*) from LON with the corresponding ratings written by $17,238$ users from August 2014 to August 2017; similarly, $8,791$ restaurants (*e.g., The River Cafe*) and $16,015$ users are obtained from NYC. The ratings are transformed into binary implicit feedback as ground truth, indicating whether the user has interacted with the specific item. To ensure the quality of the data, we retain users/items with at least five ratings only. The statistics of two datasets are summarized in Table 5.2. Moreover, we have collected the natural or system-generated labels that are affiliated with users and items as their side information (*aka.* profile). Particularly, the profile of each user includes gender (*e.g., Female*), age (*e.g.,* 25-34), and traveler styles (*e.g., Foodie* and *Beach Goer*); meanwhile, the side information of an item consists of attributes (*e.g., Art Museum* and *French*), tags (*e.g., Rosetta Stone* and *Madelenies*), and price (*e.g., $$$*). We have summarized all types of user/item side information in Table 5.3.

For each dataset, we holdout the latest 20% interaction history of each user to construct the test set, and randomly split the remaining data into training (70%) and validation (10%) sets. The validation set is used to tune hyper-parameters and the final performance comparison is conducted on the test set.

5.5.2 Experimental Settings

Evaluation Protocols

Given one positive user-item interaction in the testing set, we pair it with 50 negative instances that the user did not consume before. Then each method outputs prediction scores for these 51 instances. To evaluate the model capacity

[3]`https://www.tripadvisor.com`.

Table 5.2: Statistics of the datasets.

Dataset	User#	User Feature#	Item#	Item Feature#	Interaction#
LON-A	16,315	3,230	953	4,731	136,978
NYC-R	15,232	3,230	6,258	10,411	129,964

Table 5.3: Statistics of the side information, where the dimension of each feature is shown in parentheses.

Side Information	Features (Category#)
LON-A/NYC-R User Feature	Age (6), Gender (2), Expert Level (6), Traveler Styles (18), Country (126), City (3,072)
LON-A Attraction Feature	Attributes (89), Tags (4,635), Rating (7)
NYC-R Restaurant Feature	Attributes (100), Tags (10,301), Price (3), Rating (7)

and generalization ability of each model, we adopt logloss; meanwhile, we use ndcg@K to evaluate the performance of top-K recommendation.

- **logloss**: logarithmic loss [139] measures the probability that one predicted user-item interaction diverges from the ground truth. A lower logloss indicates a better performance.

- **ndcg@K**: ndcg@K [64, 67, 117, 115, 52] assigns the higher importance to the items within the top K positions of the ranking list. A higher ndcg@K reflects a better accuracy of getting top ranks correct.

We report the average scores for all testing instances. The same settings apply for the hyper-parameter tuning on the validation set.

Baselines

We compare our TEM with the following methods to justify the rationality of our proposal:

- **XGBoost** [27]: This is the state-of-the-art tree-based method that captures complex feature dependencies (*aka.* cross features).

- **GBDT+LR** [68]: This method feeds the cross features extracted from GBDT into the logistic regression, aiming to refine the weights for each cross feature.

- **GB-CENT** [186]: Such state-of-the-art boosting method combines the prediction results from MF and GBDT. To adjust GB-CENT to perform our tasks, we input the ID features and side information to MF and GBDT, respectively.

- **FM** [130]: This is a generic embedding-based model that encodes side information and IDs with embedding vectors. It implicitly models all

the second-order cross features via the inner product of any two feature embeddings.

- **NFM** [65]: Neural FM is the state-of-the-art factorization model under the neural network framework. It stacks multiple fully connected layers above the inner products of feature embeddings to capture higher-order and nonlinear cross features. Specially, we employed one hidden layers for NFM as suggested in [65].

Parameter Settings

For a fair comparison, we optimize all the methods with the same objective function of Equation (5.7). We implement our proposed TEM[4] using Tensorflow[5]. We use XGBoost[6] to implement the tree-based components of all methods, where the number of trees and the maximum depth of trees is searched in $\{100, 200, 300, 400, 500\}$ and $\{3, 4, 5, 6\}$, respectively. For all embedding-based components, we test the embedding size of $\{5, 10, 20, 40\}$, and empirically set the attention size same as the embedding size. All embedding-based methods are optimized using the mini-batch Adagrad for a fair comparison, where the learning rate is searched in $\{0.005, 0.01, 0.05, 0.1, 0.5\}$. Moreover, the early stopping strategy is performed, where we stopped training if the logloss on the validation set increased for four successive epoches. Without special mention, we show the results of tree number 500, maximum depth 6, and embedding size 20, and more results of the key parameters are shown in Section 5.5.5.

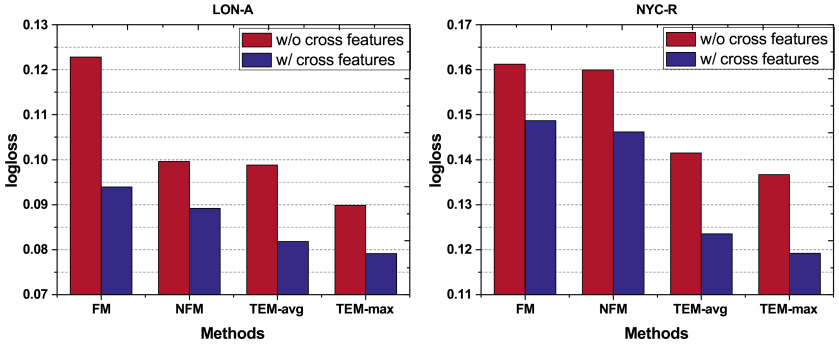
5.5.3 Performance Comparison (RQ1)

We start by comparing the performance of all the methods. We then explore how the cross features affect the recommendation results.

Overall Comparison

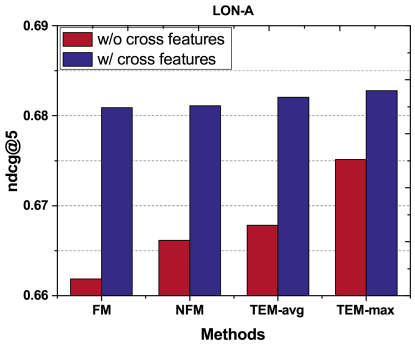
Table 5.4 displays the performance comparison *w.r.t.* logloss and ndcg@5 on LON-A and NYC-R datasets. We have the following observations:

[4] `https://github.com/xiangwang1223/TEM`.
[5] `https://www.tensorflow.org`.
[6] `https://xgboost.readthedocs.io`.

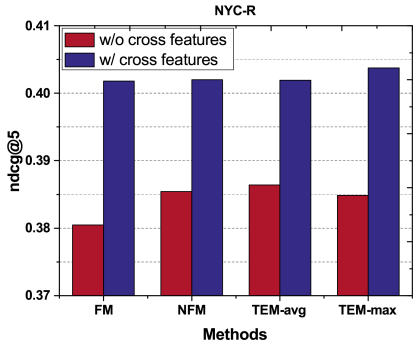


(a) logloss on LON-A

(b) logloss on NYC-R



(c) ndcg@5 on LON-A



(d) ndcg@5 on NYC-R

Figure 5.4: Performance comparison of logloss *w.r.t.* the cross features on LON-A and NYC-R datasets.

Table 5.4: Performance comparison between all the methods, where the significance test is based on logloss of TEM-max.

Dataset	LON-A			NYC-R		
Method	logloss	ndcg@5	p-value	logloss	ndcg@5	p-value
XGBoost	0.1251	0.6785	8e−5	0.1916	0.3943	4e−5
GBDT+LR	0.1139	0.6790	2e−4	0.1914	0.3997	4e−4
GB-CENT	0.1246	0.6784	6e−5	0.1918	0.3995	4e−5
FM	0.0939	0.6809	1e−2	0.1517	0.4018	5e−5
NFM	0.0892	0.6812	2e−2	0.1471	0.4020	8e−4
TEM-avg	0.0818	0.6821	–	0.1235	0.4019	–
TEM-max	**0.0791**	**0.6828**	–	**0.1192**	**0.4038**	–

- XGBoost achieves poor performance since it treats sparse IDs as ordinary features and hardly derives useful cross features based on the sparse data. It hence fails to capture the collaborative filtering effect. Moreover, it cannot generalize to unseen feature dependencies. GBDT+LR slightly outperforms XGBoost, verifying the feasibility of treating cross features as the input of one classifier and revising the weight of each cross feature.

- The performance of GB-CENT indicates that such boosting may be insufficient to fully facilitate information propagation between two models. Note that to reduce the computational complexity, the modified GB-CENT only conducts GBDT over all the instances, rather than performing

GBDT over the supporting instances of each categorical feature as suggested in [186]. Such modification may contribute to the unsatisfactory performance.

- When performing our recommendation tasks, FM and NFM, outperform XGBoost, GBDT+LR, and GB-CENT. It is reasonable since they are good at modeling the sparse interactions and the underlying second-order cross features. NFM benefits from the higher-order and nonlinear feature correlations by leveraging neural networks, thus leads to better performance than FM.

- TEM achieves the best performance, substantially outperforming NFM *w.r.t.* logloss and obtaining a comparable ndcg@5. By integrating the embeddings of cross features, TEM can achieve the comparable expressiveness to NFM. While NFM treats all feature interactions equally, TEM can employ the attention networks on identifying the personalized attention of each cross feature. We further conduct one-sample t-tests to verify that all improvements are statistically significant with p-value < 0.05.

Effect of Cross Features

To analyze the effect of cross features, we consider the variants that remove cross feature modeling, termed as FM-c, NFM-c, TEM-avg-c, and TEM-max-c. For FM and NFM, one user-item interaction is represented only by the sum of the user and item ID embeddings and their attribute embeddings, without any interactions among features. For TEM, we skip the cross feature extraction and direct feed into the raw features. As shown in Figure 5.4, we have the following findings:

- For all methods, removing cross feature modeling hurts the expressiveness adversely and degrades the recommendation performance. FM-c and NFM-c assume one user/item and her/its attributes are linearly independent, which fail to encode any interactions between them in the embedding space. Taking advantages of the attention network, TEM-avg-c and TEM-max-c still model the interactions between IDs and attributes, and achieve better representation ability than FM-c and NFM-c.

- As Figures 5.4(a) and 5.4(b) demonstrate, TEM significantly outperforms FM and NFM by a large margin *w.r.t.* logloss, verifying the substantial influence of explicit cross feature modeling. While FM and NFM consider all the underlying feature correlations, neither of them explicitly presents the cross features or identifies the importance of each cross feature. This makes them work as a black-box and hurts their explainability. Therefore, the improvement achieved by TEM again verifies the effectiveness of the explicit cross features refined from the tree-based component.

- Lastly, while exhibiting the lowest logloss, TEM achieves only comparable performance *w.r.t.* ndcg@5 to that of NFM, as shown in Figures 5.4(c) and 5.4(d). It indicates the unsatisfied generalization ability of TEM, since the cross features extracted from GBDT only reflect the feature dependencies observed in the dataset and consequently TEM cannot generalize to the unseen rules. We leave the further exploration of the generalization ability of our TEM as the future work.

5.5.4 Case Studies (RQ2)

Apart from being comparable at predictive accuracy, the key advantage of TEM over other methods is that its learning process is transparent and easily explainable. Towards this end, we show examples drawn from TEM-avg on LON-A to demonstrate its explainability and scrutability.

Explainability

To demonstrate the explainability of TEM, we focus on a sampled user, whose profile is {age: 35-49, gender: *female*, country: *the United Kingdom*, city: *London*, expert level: 4, traveler styles: *Art and Architecture Lover, Peace and Quite Seeker, Family Vacationer, Urban Explorer*}; meanwhile, we randomly select five attractions, {i_{31}: *National Theatre*, i_{45}: *The View form the Shard*, i_{49}: *The London Eye*, i_{93}: *Camden Street Art Tours*, i_{100}: *Royal opera House*}, from the user's holdout testing set. Figure 5.5 visualizes the learning results, where a row represents an attraction, and a column represents a cross feature (we sample five cross features which are listed in Table 5.5). The left heat map presents her attention scores over the five sampled cross features and the right

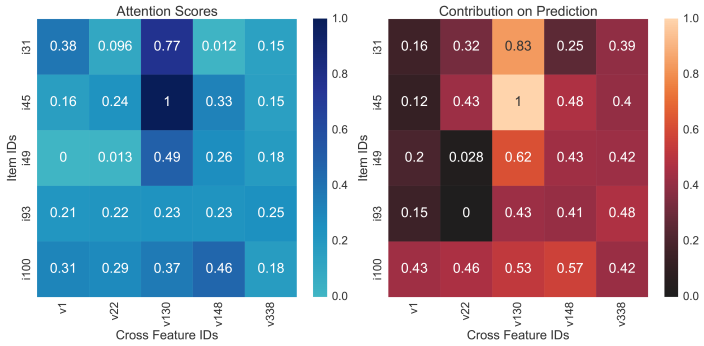


Figure 5.5: Visualization of cross feature attentions produced by TEM-avg on LON-A. An entry of the left and right heat map visualizes the attention value w_{uil} and its contribution to the final prediction, *i.e.*, $w_{uil}\mathbf{r}_2^\top \mathbf{v}_l$, respectively.

displays the contributions of these cross features for the final prediction.

We first focus on the left heat map of attention scores. Examining the attention scores of a row, we can explain the recommendation for the corresponding attraction using the top cross features. For example, we recommend *The View from the Shard* (*i.e.,* the second row i_{45}) for the user mainly because of the dominant cross feature v_{130}, evidenced by the highest attention score of 1 (*cf.* the entry at the second row and the third column). Based on the attention scores, we can attribute her preferences on *The View from the Shard* to her special interests in the item aspects of *Walk Around* (from v_{130}), *Top Deck & Canary Wharf* (from v_{22}), and *Camden Town* (from v_{148}). To justify the rationality of the reasoning, we further check the user's visiting history, finding that the three item aspects have frequently occurred in her historical items.

In right heat map of Figure 5.5, an entry denotes the contribution of the corresponding cross feature (*i.e.,* $y'_{uil} = w_{uil}\mathbf{r}_2^\top \mathbf{v}_l$) to the final prediction Jointly analyzing the left and right heat maps, we find that the attention score w_{uil} is generally consistent with y_{uil}, which contains useful cues about the user's preference. Based on such outcome, we can utilize the attention scores of cross

Table 5.5: Descriptions of the cross features in Figure 5.5.

ID	Description of Cross Features shown in Figure 5.5
v_1	[User Country=UK] & [User Style=Art and Architecture Lover] ⇒ [Item Attribute=Concerts and Shows] & [Item Tag=Imelda Staunton]
$\mathbf{v_{22}}$	[User Age=35-49] & [User Country=UK] ⇒ [Item Tag=**Camden Town**] & [Item Rating=4.0]
$\mathbf{v_{130}}$	[User Age≠ 25-34] & [User Gender=Female] & [User Style=Peace and Quiet Seeker] ⇒ [Item Attribute=Sights & Landmarks] & [Item Tag=**Walk Around**]
$\mathbf{v_{148}}$	[User Age≠ 50-64] & [User Country≠USA] ⇒ [Item Tag=**Top Deck & Canary Wharf**]
v_{336}	[User Age=35-49] & [User Country=UK] & [User Style=Art and Architecture Lover] ⇒ [Item Tag=Royal Opera House] & [Item Tag=Interval Drinks]

Table 5.6: Scrutable recommendation for a sampled user on LON-A, where the first row and second row list the original and adjusted recommended attractions, respectively.

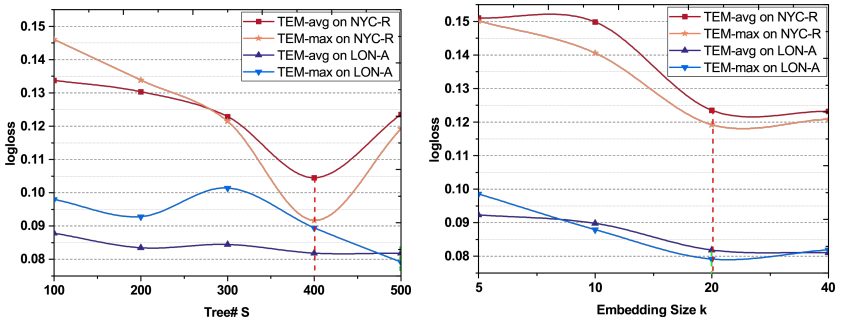
Top Ranked Recommendation List on LON-A	
1. Original	1. London Fields Park, 2. Old Compton Street, 3. The Mall, 4. West End, 5. Millennium Bridge
2. Adjusted	1. London Fields Park, 2. Greenwich Foot Tunnel, 3. Covent Garden, 4. Kensington Gardens, 5. West End

features to explain a recommendation (*e.g.,* the user prefers i_{45} owing to the top rules of v_{130} and v_{148} weighted with personalized attention scores of 1 and 0.33). This case demonstrates TEM's capability of providing more informative explanations according to a user's preferred cross features, which we believe are better than mere labels or similar user/item list.

Scrutability

Apart from making the recommendation process transparent, our TEM can further allow a user to correct the process, so as to refresh the recommendation as she desires. This property of adjusting recommendation is known as the scrutability [155, 64]. As for TEM, the attention scores of cross features serve as a gateway to exert control on the recommendation process. We illustrate it using another sampled user in Table 5.6.

The profile of this user indicates that she enjoys the traveler style of *Urban Explorer* most; moreover, most attractions in the historical interactions of her are tagged with *Sights & Landmarks*, *Points of Interest* and *Neighborhoods*. Hence, TEM detects such frequent co-occurred cross features and accordingly recommends some attractions like *Old Compton Street* and *The Mall* to her. Assuming that the user attempts to scrutinize TEM and would like to visit some attractions tagged with *Garden* that are suitable for the *Nature Lover*. Towards this end, we assign the cross features containing *[User Style=Nature Lover] & [Item Attribute=Garden]* with a higher attentive weight, and then get the predictions of TEM to refresh the recommendations. In the adjusted recommendation list, the *Greenwich Foot Tunnel*, *Covent Garden*, and *Kensington Gardens* are ranked at the top positions. Therefore, based on the transparency and simulated scrutability, we believe that our TEM is easy-to-interpret, explainable and scrutable.



(a) logloss vs. tree number S (b) logloss vs. embedding size k

Figure 5.6: Performance comparison of logloss *w.r.t.* the tree number S and the embedding size k.

5.5.5 Hyper-parameter Studies (RQ3)

We empirically study the influences of several factors, such as the number of trees and the embedding size, on our TEM method.

Impact of Tree Number.

The number of trees in TEM indicates the coverage of cross features, reflecting how much useful information is derived from the datasets. Figure 5.6(a) presents the performance *w.r.t.* logloss by varying the tree number S. We can see the logloss of TEM gradually decreases with more trees, whereas the performance is generally improved. Using a tree number of 400 and 500 leads to the best performance on NYC-R and LON-A, respectively. When the tree number exceeds the optimal settings (*e.g.,* S equals to 500 on NYC-R), the logloss increases, which may suffer from overfitting. This emphasizes the significance of the tree settings, which is consistent with [68, 186]

Impact of Embedding Size.

The empirical results displayed in Figure 5.6(b) indicates the substantial influence of embedding size upon TEM. Enlarging the embedding size, TEM benefits from more powerful representations of the user-item pairs. Moreover, TEM-max shows consistent improvement over TEM-avg in most cases. We attributed such improvement to the nonlinearity achieved by the max pooling operation, which can select most informative cross features out, as discussed in Section 5.4.1. However, the oversized embedding may cause overfitting and degrade the performance, which is consistent with [65, 165]

5.6 Conclusion

In this chapter, we proposed a tree-enhanced embedding method (TEM), which seamlessly combines the generalization ability of embedding-based models with the explainability of tree-based models. Owing to the explicit cross features extracted from tree-based part and the easy-to-interpret attention network, the whole prediction process of our solution is fully transparent and self-explainable. Meanwhile, TEM can achieve comparable performance as the state-of-the-art recommendation methods.

While providing explainable results, TEM model has several limitations. First, it is a two-stage method, rather than an end-to-end model, since the cross features needs to be extracted first. Hence the performance depends heavily on the quality of cross features. Second, it embeds the ID of feature crossing only, failing to consume their semantics and correlations among each other. These shortcoming would limit the effectiveness of TEM.

In future, we will extend our TEM in three directions. First, we attempt to jointly learn the tree-based and embedding-based models, rather than separately modelling two components. This can facilitate the information propagation between two components. Second, we consider other context information, such as time, location, and user sentiments, to further enrich our explainability. Third, we will explore the effectiveness of involving knowledge graphs and logic rules into our TEM.

Chapter 6

Knowledge-enhanced Recommendation

So far, we have demonstrated works that exploit auxiliary information across different channels to boost the recommendation performance. To complete this thesis, in this chapter, we plan to investigate reasoning for recommendation, specially to uncover the logical reasons with rich semantics behind the recommendations. Towards this end, we propose to incorporate knowledge graph with the user-item interaction data to discover the knowledge-level connections between a user and the target item.

6.1 Introduction

Prior efforts have shown the importance of incorporating auxiliary data into recommender systems, such as user profiles [165, 164] and item attributes [12, 35]. Recently, knowledge graphs (KGs) have attracted increasing attention [178, 142, 159], due to its comprehensive auxiliary data: background knowledge of items and their relations amongst them. It usually organizes the facts of items in the form of triplets like (*Ed Sheeran*, *IsSingerOf*, *Shape of You*), which can be seamlessly integrated with user-item interactions [24]. More important, by exploring the interlinks within a KG, the connectivity between users and items reflects their underlying relationships, which are complementary to user-item interaction data.

Extra user-item **connectivity** information derived from KG endows

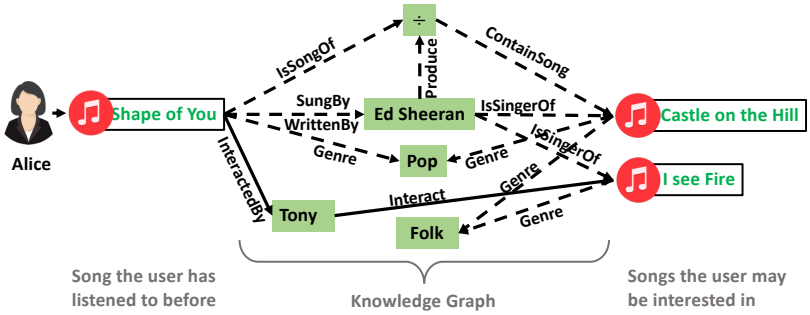


Figure 6.1: Illustration of KG-aware recommendation in the music domain. The dashed lines between entities are the corresponding relations, while the sold lines are the user-item interactions.

recommender systems the ability of **reasoning** and **explainability**. Taking music recommendation as an example (Figure 6.1), a user is connected to *I See Fire* since she likes *Shape of You* sung by the same singer *Ed Sheeran*. Such connectivity helps to **reason** about unseen user-item interactions (*i.e.,* a potential recommendation) by synthesizing information from paths.

Running Example: (*Alice, Interact, Shape of You*)∧(*Shape of You, SungBy, Ed Sheeran*)∧(*Ed Sheeran, IsSingerOf, I See Fire*)⇒(*Alice, Interact, I See Fire*). Clearly, the reasoning unveils the possible user intents behind an interaction, offering **explanations** behind a recommendation. How to model such connectivity in KGs, hence, is of critical importance to inject knowledge into a recommender systems.

Prior efforts on knowledge-aware recommendation are roughly categorized into path and embedding fashion. Path-based methods [174, 173, 58] introduce *meta-paths* to refine the similarities between users and items. However, we argue that meta-path is inefficient in reasoning over KGs, owing to the following limitations: 1) As relations are usually excluded from meta-paths, they hardly specify the holistic semantics of paths, especially when similar entities but different relations are involved in a meta-path; and 2) They fail to automatically uncover and reason on unseen connectivity patterns, since meta-paths requires domain knowledge to be predefined.

Another line of research [178, 160, 73] leverages knowledge graph embedding (KGE) techniques, such as TransE [17] and TransR [94], to regularize the representations of items. As a result, items with similar connected entities have similar representations, which facilitate the collaborative learning of user

interests. Despite performance improvements, we argue that KGE regularization lacks the reasoning ability. Specially, it only considers direct relations between entities, rather than the multi-hop relation paths as the Running Example shows. Moreover, the characterization of user-item connectivity is achieved in a rather implicit way, that is, to guide the representation learning, but not to infer a user's preference.

In this work, we aim to fill the research gap by developing a solution that reasons on paths to infer user preferences on items. In terms of reasoning, we expect our method to model the sequential dependencies of entities and sophisticated relations of a path connecting a user-item pair. In terms of explainability, we would like our method to discriminate the different contributions of different paths, when inferring user interests.

Towards this end, we propose a new solution, named Knowledge-aware Path Recurrent Network (KPRN), which not only generates representations for paths by accounting for both entities and relations, but also performs reasoning based on paths to infer user preference. Specifically, we first extract qualified paths between a user-item pair from the KG, each of which consists of the related entities and relations. We then adopt a Long Short-Term Memory (LSTM) network to model the sequential dependencies of entities and relations. Thereafter, a pooling operation is performed to aggregate the representations of paths to obtain prediction signal for the user-item pair. More importantly, the pooling operation is capable of discriminating the contributions of different paths for a prediction, which functions as the attention mechanism [26, 109]. Owing to such attentive effect, our model can offer path-wise explanations such as *Castle on the Hill is recommended since you have listened to Shape of You sung and written by Ed Sheeran.* We conduct extensive experiments on two datasets to verify our method.

The contributions of this work are threefold:

- We highlight the importance of performing explicit reasoning on KG to better reveal the reasons behind a recommendation.
- We propose an end-to-end neural network model to learn path semantics and integrate them into recommendation.
- We contribute a dataset to study KG for recommendation by aligning a

MovieLens benchmark with IMDB. We verify our method on the data, and release the data and the codes to facilitate the community working on emerging field of KG-enhanced recommendation.

6.2 Knowledge-aware Path Recurrent Network

In this section, we elaborate our proposed method, as illustrated in Figure 6.2. Before introducing our proposed method, we first formally define Knowledge Graph, user-item data and describe how to combine them in an enriched knowledge graph as the inputs of our model.

6.2.1 Background

A knowledge Graph (KG) is a directed graph whose nodes are entities $\mathcal{E}$ and edges $\mathcal{R}$ denote their relations. Formally, we define KG as $\mathcal{KG} = \{(h, r, t)|h, r \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet (h, r, t) indicates a fact that there is a relationship r from head entity h to tail entity t.

The user-item interaction data is usually presented as a bipartite graph. In particular, we use $\mathcal{U} = \{u_t\}_{t=1}^{M}$ and $\mathcal{I} = \{i_t\}_{t=1}^{N}$ to separately denote the user set and the item set, where M and N are the number of users and items, respectively. Following [24], we represent the interaction between a user and an item with a triplet τ =(u, *interact*, i), if there is an observed interaction (*e.g.,* rate, click, and view feedbacks), where *interact* is a pre-defined relation.

We merge the item set and the entity set through string matching: $\mathcal{I} \subseteq \mathcal{E}$, so that the two structural data are integrated into an enriched knowledge graph $\mathcal{G} = \{(h, r, t)|h, r \in \mathcal{E}', r \in \mathcal{R}'\}$, where $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$ and $\mathcal{R}' = \mathcal{R} \cup \{\text{interact}\}$. For consistency, the Knowledge Graph (KG) denotes the combined graph $\mathcal{G}$ including both original KG and user-item data, otherwise noted.

6.2.2 Preference Inference via Paths

The triplets in the KG clearly describe direct or indirect (i.e. multiple-step) relational properties of items, which shall constitute one or several paths between the given user and item pair. We explore these paths in order to achieve comprehensively reasoning and understanding for recommendation.

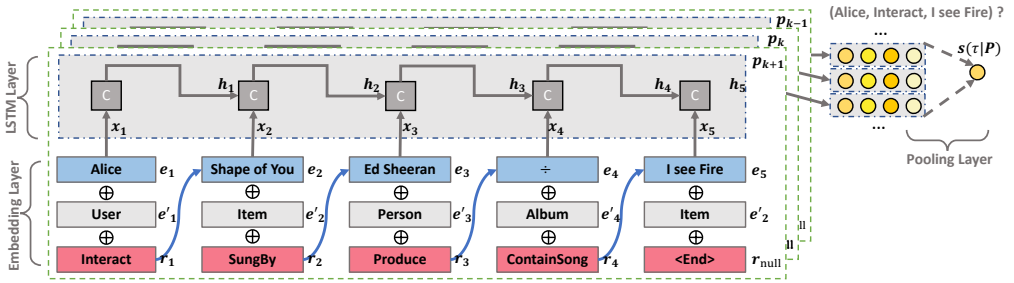


Figure 6.2: Schematic overview of our model architecture. The embedding layer contains 3 individual layers for entity, entity type, and relation type, respectively. The concatenation of the 3 embedding vectors is the input of LSTM for each path.

Within $\mathcal{G}$, we formally define the path from the user u to the item i as a sequence of entities and relations: $p = [e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{L-1}} e_L]$, where $e_1 = u$, $e_L = i$; (e_l, r_l, e_{l+1}) is the l-th triplet in p, and L denotes the number of triplets in the path. The construction of paths will be elaborated in the section of Section Path Extraction.

Next, we will use a realistic example to show the sophisticated relations (i.e. paths) between a user and an item behind their possible interactions, which inspires us to model the high-level semantics of path compositionally by considering both entities and (multiple-step) relations.

Examples: Consider the music recommendation shown in Figure 6.1, where the "listen to Castle on the Hill" behavior of user Alice can be referred by the following paths:

- $p_1 = [\text{Alice} \xrightarrow{\text{Interact}} \text{Shape of You} \xrightarrow{\text{IsSongOf}} \div \xrightarrow{\text{ContainSong}} \text{Castle on the Hill}]$;
- $p_2 = [\text{Alice} \xrightarrow{\text{Interact}} \text{Shape of You} \xrightarrow{\text{SungBy}} \text{Ed Sheeran} \xrightarrow{\text{IsSingerOf}} \text{Castle on the Hill}]$.
- $p_3 = [\text{Alice} \xrightarrow{\text{Interact}} \text{Shape of You} \xrightarrow{\text{InteractedBy}} \text{Tony} \xrightarrow{\text{Interact}} \text{Castle on the Hill}]$;

These paths from the same user *Alice* to the same item *Castle on the Hill* obviously express their different multiple-step relations, and implies various compositional semantics and possible explanations of the listen behavior. In particular, p_1 and p_2 infer that *Alice* may prefer songs that belonging to the album ÷ and the songs sung by *Ed Sheeran*, while p_3 reflects the collaborative filtering (CF) effect: similar users tend to have similar preferences. Therefore,

from the view of reasoning, we consume the connectivity along all paths to learn compositional relation representations, and weighted pool them together for predicting the *interact* relation between the user and the target item.

Task Definition: Our task can be formulated as follows: given a user u, a target item i, and a set of paths $\mathcal{P}(u,i) = \{p_1, p_2, \cdots, p_K\}$ connecting u and i, the holistic goal is to estimate the interaction by:

$$\hat{y}_{ui} = f_\Theta(u, i|\mathcal{P}(u,i)), \tag{6.1}$$

where f denotes the underlying model with parameters Θ, and $\hat{y}_{ui}$ presents the predicted score for the user-item interaction. Distinct from embedding-based methods, we can explain $\hat{y}_{ui}$ as the plausibility score of the triplet $\tau = (u, \textit{interact}, i)$ inferred by the connectivity $\mathcal{P}(u,i)$.

6.2.3 Modeling

KPRN takes a set of paths of each user-item pair as input, and outputs a score indicating how possible the user will interact the target item. As illustrated in Figure 6.2, there are three key components: (1) embedding layer to project three types of IDs information: the entity, entity type, and the relation pointing to the next node into a latent space, (2) LSTM layer that encodes the elements sequentially with the goal of capturing the compositional semantics of entities conditioned on relations, and (3) pooling layer to combine multiple paths and output the final score of the given user interacting the target item.

Embedding Layer

Given a path p_k, we project the type (*e.g.,* person or movie) and specific value (*e.g.,* Peter Jackson or The Hobbit II) of each entity into two separate embedding vectors, $\mathbf{e}_l \in \mathbb{R}^d$ and $\mathbf{e}'_l \in \mathbb{R}^d$, where d is the embedding size.

In real-world scenarios, it is common that the same entity-entity pairs may have different semantics due to different relations connecting them. Such differences may reveal the diverse intents about why a user selected the item. As an example, let (*Ed Sheeran, IsSingerOf, Shape of You*) and (*Ed Sheeran, IsSongwriterOf, Shape of You*) be the triplets in two paths referring a user's preferences. Without specifying the relations, these paths will be represented

as the same embeddings, regardless of the possibility that the user only prefers songs sung by Ed Sheeran, rather than that written by Ed Sheeran. We hence believe that it is important to explicitly incorporate the semantics of relations into path representation learning. Towards this end, each relation r_l in p_k is represented as an embedding vector $\mathbf{r}_l \in \mathbb{R}^d$. As a result, we obtain a set of embeddings for path p_k, $[\mathbf{e}_1, \mathbf{r}_1, \mathbf{e}_2, \cdots, \mathbf{r}_{L-1}, \mathbf{e}_L]$, where each element denotes an entity or a relation.

LSTM Layer

With the embedding sequence to describe a path, we can employ RNN models to explore the sequential information, and generate a single representation for encoding its holistic semantics. Among various RNN methods [84], we adopt LSTM since it is capable of memorizing long-term dependency in a sequence. Such long-term sequential pattern is crucial to reason on paths connecting a user and item entities to estimate the confidence of the "interact" relation.

At the path-step $l-1$, the LSTM layer outputs a hidden state vector $\mathbf{h}_{l-1}$, consuming the subsequence $[e_1, r_1, \cdots, e_{l-1}, r_{1-1}]$. Simultaneously, we concatenate the embedding of current entity e_{l-1} and relation r_{l-1} as the input vector:

$$\mathbf{x}_{l-1} = \mathbf{e}_{l-1} \oplus \mathbf{e}'_{l-1} \oplus \mathbf{r}_{l-1}, \tag{6.2}$$

where $\oplus$ is the concatenation operation. Noted that, for the last entity e_L, a null relation r_L is padded to the end of path. As such, the input vector contains not only the sequential information, but also the semantic information of the entity and its relation to the next entity. Consequently, $\mathbf{h}_{l-1}$ and $\mathbf{x}_{l-1}$ are used to learn the hidden state of the next path-step l, which is defined via the following

equations:

$$\begin{aligned}
\mathbf{z}_l &= \tanh(\mathbf{W}_z\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_z) \\
\mathbf{f}_l &= \sigma(\mathbf{W}_f\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_f) \\
\mathbf{i}_l &= \sigma(\mathbf{W}_i\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_i) \\
\mathbf{o}_l &= \sigma(\mathbf{W}_o\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_o) \\
\mathbf{c}_l &= \mathbf{f}_l \odot \mathbf{c}_{l-1} + \mathbf{i}_l \odot \mathbf{z}_l \\
\mathbf{h}_l &= \mathbf{o}_l \odot \tanh(\mathbf{c}_l)
\end{aligned} \tag{6.3}$$

where $\mathbf{c}_l \in \mathbb{R}^{d'}$, $\mathbf{z} \in \mathbb{R}^{d'}$ denote the cell (memory) state vector and information transform module, respectively, and d' is the number of hidden units; $\mathbf{i}_l$, $\mathbf{o}_l$, and $\mathbf{f}_l$ separately represents the input, output, and forget gate. $\mathbf{W}_z$, $\mathbf{W}_i$, $\mathbf{W}_f$, and $\mathbf{W}_o \in \mathbb{R}^{d' \times 3d}$, and $\mathbf{W}_h \in \mathbb{R}^{d' \times d'}$ are mapping coefficient matrices, while $\mathbf{b}_z$, $\mathbf{b}_i$, $\mathbf{b}_f$, and $\mathbf{W}_o$ are bias vectors. $\sigma(\cdot)$ is the activation function set as sigmoid, and $\odot$ stands for the element-wise product of two vectors. Taking advantages of the memory state, the last state $\mathbf{h}_L$ is capable of representing the whole path $\mathbf{p}_k$.

Having established the representation of path $\mathbf{p}_k$, we aim to predict the plausibility of $\tau = (u, \text{interact}, i)$. Towards this end, two fully-connected layers are adopted to project the final state into the predictive score for output, given by:

$$s(\tau|\mathbf{p}_k) = \mathbf{W}_2^\top \text{ReLU}(\mathbf{W}_1^\top \mathbf{p}_k), \tag{6.4}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the coefficient weights of the first and second layers respectively, bias vectors are omitted form simplicity, and the rectifier is adopted as the activation function.

Weighted Pooling Layer

A user-item entity pair usually has a set of paths connecting them in a KG. Let $\mathcal{S} = \{s_1, s_2, \cdots, s_K\}$ be the predictive scores for K paths, $\mathcal{P}(u, i) = \{p_1, p_2, \cdots, p_K\}$, connecting a user-item pair (u, i), where each element is calculated based on Equation (6.4). The final prediction could be the average of

the scores of all paths, which is formulated as,

$$\hat{y}_{ui} = \sigma(\frac{1}{K}\sum_{k=1}^{K} s_k). \tag{6.5}$$

Nevertheless, prior studies [174, 142] suggest that different paths have varying contributions to model user preferences, while Equation (6.5) fails to specify importance of each path. Inspired by previous work [104, 26], we design a weighted pooling operation to aggregate scores of all paths. Here the pooling function is defined as follows,

$$g(s_1, s_2, \cdots, s_K) = \log\left[\sum_{k=1}^{K} \exp\left(\frac{s_k}{\gamma}\right)\right], \tag{6.6}$$

and the final prediction score is given by,

$$\hat{y}_{ui} = \sigma(g(s_1, s_2, \cdots, s_K)), \tag{6.7}$$

where γ is the hyper-parameter to control each exponential weight. Such pooling is capable of distinguishing the path importance, which is attributed by the gradient:

$$\frac{\partial g}{\partial s_k} = \frac{\exp(s_k/\gamma)}{\gamma \sum_{k'} \exp(s_{k'}/\gamma)}, \tag{6.8}$$

which is proportional to the score of each path during the back-propagation step. Moreover, the pooling function endows the final prediction more flexibility. In particular, when setting $\gamma \to 0$, the pooling function can degenerate to max-pooling; whereas, it can degrade to mean-pooling by setting $\gamma \to \infty$. We conduct a case study on exploring the utility of the weighted pooling operation in Section Case Studies.

6.2.4 Learning

Similar to the spirit in recent work [67], we treat the recommender learning task as a binary classification problem, where an observed user-item interaction is assigned a target value 1, otherwise 0. We use the pointwise learning methods to learn the parameters of our model. In particular, the negative log-likelihood

is adopted as the objective function, which is defined as follows,

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{O}^+} \log \hat{y}_{ui} + \sum_{(u,j)\in\mathcal{O}^-} \log(1-\hat{y}_{uj}), \tag{6.9}$$

where $\mathcal{O}^+ = \{(u,i)|y_{ui}=1\}$ and $\mathcal{O}^- = \{(u,j)|y_{uj}=0\}$ are the positive and negative user-item interaction pairs, respectively. We conduct L_2 regularization on the trainable parameters Θ, which is omitted here for simplicity, to avoid overfitting. We elaborate the implementation details in the section of Experimental Settings.

6.3 Experiments

In this section, we perform experiments on two real-world datasets to evaluate our proposed method. We aim to answer the following research questions:

- **RQ1:** Compared with the state-of-the-art KG-enhanced methods, how does our method perform?
- **RQ2:** How does the multi-step path modeling (*e.g.,* the incorporation of both entity and relation types) affect KPRN?
- **RQ3:** Can our proposed method reason on paths to infer user preferences towards items?

6.3.1 Dataset Description

We consider two scenarios: movie recommendation and music recommendation. For movie domain, we use the combination of MovieLens-1M and IMDb datasets, named MI, which are linked by the titles and release dates of movies. In particular, MovieLens-1M offers the user-item interaction data, while IMDb serves as the KG part that contains auxiliary information on movies, such as genre, actor, director, and writer. For music domain, we use the benchmark dataset, KKBox, which is adopted from the WSDM cup 2018 Challenge and is provided by the music streaming service KKBox. Beyond the user-item interaction data, this dataset contains the descriptions of music like singer, songwriter, and genre. The statistics of two datasets are summarized in Table 6.1.

Table 6.1: Statistics of our datasets.

	Dataset	MI	KKBox
User-Item Interaction	#Users	6,040	34,403
	#Items	3,859	2,296,833
	#Interactions	998,034	3,714,655
Knowledge Graph	#Entities	11,462	2,851,220
	#Entity Types	4	4
	#Relation Types	6	6
	#Triplets	1,017,030	11,182,682
Path	#Paths	55,573,556	38,192,484
	Avg Path Length	5.07	5.09

Following previous efforts [174, 67, 142], we process the datasets as: if a user rates a movie or has an interaction record with a song, we set the user-movie or user-song pair as the observed positive feedback with the target value of 1, and 0 otherwise.

For each dataset, we holdout the 80% and 20% interaction history of each user randomly to construct the training and test sets. For each positive user-item interaction pair in the training set, we conducted the negative sampling strategy to pair it with four negative items that the user has not interacted with. During the test stage, the ratio between positive and negative interactions is set as $1:100$, namely, 100 negative items are randomly sampled and pair with one positive item in the testing set.

6.3.2 Path Extraction

In practice, it is labor intensive and infeasible to fully exploring all connected paths over the KG. Especially, the number of paths grows exponentially *w.r.t.* the length of path, where millions of interlinks will be generated. As suggested in prior efforts [142], truncating all paths at a certain length and disregarding remote connections are sufficient to model the connectivity between a user-item pair. Moreover, as pointed out by [149], paths with length greater than six will introduce noisy entities. Therefore, we extract all qualified paths, each with length up to six, that connect all user-item pairs.

6.3.3 Experimental Settings

Evaluation Metrics

We adopt two evaluation protocols to evaluate the performance of top-K recommendation and preference ranking, respectively, given by:

- **hit**@K considers whether the relevant items are retrieved within the top K positions of the recommendation list.
- **ndcg**@K measures the relative orders among positive and negative items within the top K of the ranking list.

We report the average metrics at $K = \{1, 2, \cdots, 15\}$ of all instances in the test set.

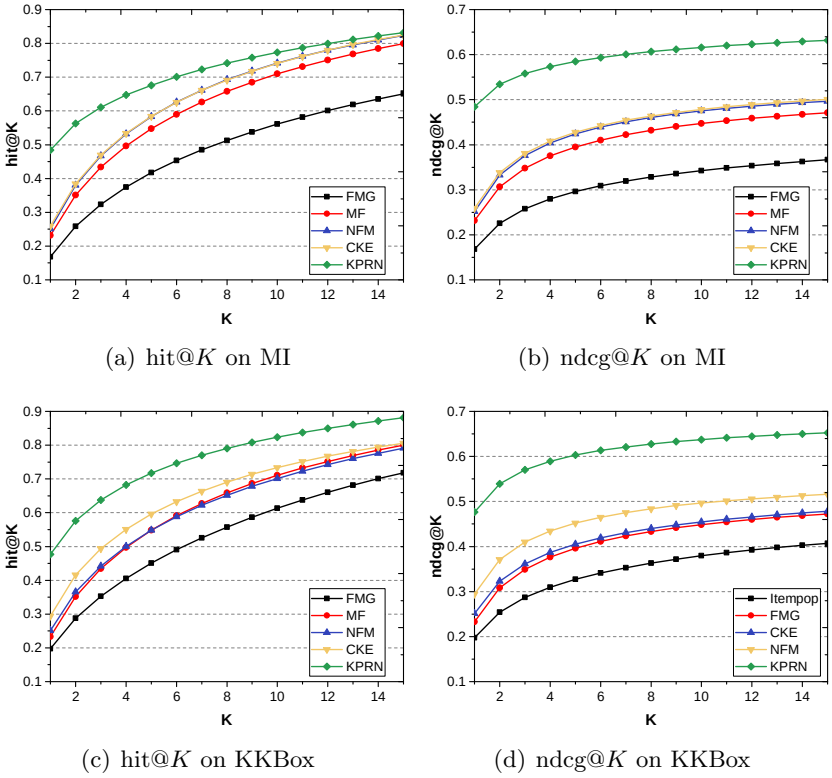
Baselines

We compare our proposed method with the following methods:

- **MF** [132]: This is matrix factorization with Bayesian personalized ranking (BPR) loss, which solely utilizes user-item interaction.
- **NFM** [65]: The method is a state-of-the-art factorization model which treats historical items as the features of users. Specially, we employed one hidden layer as suggested in [65].
- **CKE** [178]: Such embedding-based method is tailored for KG-enhanced recommendation, which integrates the representations from Matrix Factorization [132] and TransR [94] to enhance the recommendation.
- **FMG** [185]: This is a state-of-the-art meta-path based method, which predefines various types of meta-graphs and employs Matrix Factorization on each meta-graph similarity matrix to make recommendation.

Parameter Settings

For fair comparison, we learn all models from scratch without any pretrained parameters. We optimize all models with Adaptive Moment Estimation (Adam) and apply a grid search to find out the best settings of hyperparameters. The learning rate is searched in $\{0.001, 0.002, 0.01, 0.02\}$, while the coefficient of L_2



(a) hit@K on MI (b) ndcg@K on MI

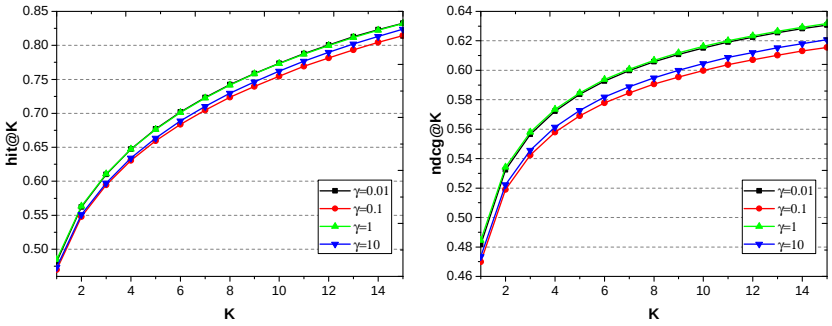
(c) hit@K on KKBox (d) ndcg@K on KKBox

Figure 6.3: Top-K recommendation performance between all the methods on MI and KKBox datasets *w.r.t.* hit@K and ndcg@K.

Table 6.2: Performance comparison of KPRN and KPRN-r and their effects on relation modeling.

	MI						KKBox					
	hit@5	hit@10	hit@15	ndcg@5	ndcg@10	ndcg@15	hit@5	hit@10	hit@15	ndcg@5	ndcg@10	ndcg@15
KPRN-r	0.635	0.738	0.801	0.533	0.566	0.583	0.712	0.821	0.878	0.607	0.632	0.647
KPRN	0.676	0.773	0.832	0.584	0.616	0.632	0.717	0.823	0.881	0.613	0.637	0.652

regularization is tuned amongst $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. Other hypermeters of our proposed model are empirically set as follows: the batch size is 256, the embedding size of relation and entity type is 32, the embedding size of entity value is 64, and the unit number of LSTM is 256. The dimensions of latent factors for MF, NFM, and CKE are empirically set to be 64. For FMG, we set the rank used to factorize meta-graph similarity matrices to be 10, and the factor size of the second-order weights as 10, as suggested by [185]. Moreover, the early stopping strategy is performed, *i.e.,* premature stopping if hit@15 on the test data does not increase for five successive epochs.



(a) hit@K on MI (b) ndcg@K on MI

Figure 6.4: Performance comparison *w.r.t.* γ on the MI dataset.

6.3.4 Performance Comparison (RQ1)

Figure 6.3 reports our experimental results on two datasets *w.r.t.* hit@K and ndcg@K. We have the following findings:

- FMG gives poor performance in both datasets. This indicates that meta-graph based methods, which rely heavily on the predefined meta-graph patterns, may introduce remote entities and fail to fully explore the user-item connectivity.

- NFM achieves better performance than MF. It makes sense since by treating the rated items as the user features, NFM essentially enhances the second-order user-item proximity, while MF only considers the first-order user-item connections.

- Compared to CF-based methods (MF and NFM), the performance of CKE indicates that incorporating KG can solve the data sparsity issue effectively. In particular, CKE shows consistent improvement over KKBox dataset that is extremely sparse, while only achieving comparable performance to NFM on MI dataset which has denser interaction data.

- KPRN substantially outperforms CKE *w.r.t.* hit@K and ndcg@K, achieving the best performance. By leveraging paths to infer user preference, KPRN is capable of exploring the user-item connectivity in an explicit way, while the embedding-based method (CKE) only utilizes KG to guide the representation learning of items. This verifies the importance of leveraging both entities and relations of KG. Further

analyzing Figures 6.3(b) and 6.3(d) reveal that KPRN outperforms other baselines by a larger margin *w.r.t.* ndcg@K, demonstrating the strong capacity of preference ranking.

6.3.5 Study of KPRN (RQ2)

To investigate the role of path modeling, we start by explore the influence of relation in paths. We then study how the weighted pooling operation affects the performance.

Effects of Relation Modeling

We consider one variant of KPRN without the relation modeling, termed as KPRN-r. In particular, the relation embedding $\mathbf{r}_{l-1}$ in Equation (6.2) is discarded to generate the input vector $\mathbf{x}_{l-1}$. In Table 6.2, we compare KPRN with KPRN-r in terms of hit@K and ndcg@K, where K is selected from $\{5, 10, 15\}$. We have the following observations:

- Without considering relations in paths, the performance of KPRN-r decreases on both datasets. This justifies our intuition that specifying different relations is of importance to capture the path semantics, especially when the same entities are involved.

- We find that KPRN improves KPRN-r by 6.45% *w.r.t.* hit@5 on MI, while only 0.70% on KKBox. One reason may be that as MI is much denser than KKBox and it is common that, in MI, multiple paths connect a user-item pair with similar entities but different relations, whereas fewer paths are offered in KKBox. This demonstrates that, given strong connectivity between users and items, specifying relations of paths is of more importance to explore the fine-grained interests of users.

Effects of Weighted Pooling

To integrate the prediction scores of multiple paths between a user-item pair, a weighted pooling operation is carefully designed. To analyze its effect, we set the value γ as $\{0.01, 0.1, 1, 10\}$ and report the performance on MI in Figure 6.4. We find that,

- When γ decrease from 1 to 0.1, the weighted pooling operation degrades the performance, since it is similar to max-pooling and selects only the most important paths as the user-item connectivity.

- The performance *w.r.t.* hit@K and ndcg@K becomes poorer, when increasing γ from 1 to 10. It makes sense since it tends to aggregate contributions from more paths, rather than the most informative ones.

6.3.6 Case Studies (RQ3)

Another desirable property of KPRN is to reason on paths to infer the user preferences towards target items and generate reasonable explanations. This is because our model capture the higher-level semantics from these key factors: entity, entity type, and relation. To demonstrate this, we show an example drawn from KPRN on movie recommendation task.

We randomly select a user, whose ID is u4825 in MovieLens-1M, and select the movie Shakespeare in Love from her interaction record. We then extract all the qualified paths connecting the user-item pair and present the subgraph in Figure 6.5. We have several observations.

- Collaborative filtering effect plays a pivotal rule to recommend the movie Shakespeare in Love to the user, since the interaction behaviors from other users (*e.g.,* u940 and u5448) are involved in two paths. In particular, the path containing u5448 offers the high contribution score of 0.356 to infer the user's interest.

- The target item is connected to what u4825 has watched before (*e.g., Rush Hour, Titanic, and Fantasia*) by the shared knowledge entities, such as actor (*Tom Wilkinson*) and director (*James Algar*). This shows that KPRN is capable of extending user interests along KG paths.

- Analyzing these three paths jointly, we find that different paths describe the user-item connectivity from dissimilar angles, which can be treated as the evidence why the item is suitable for the user. Specially, we can offer path-wise explanations such as *Shakespeare in Love is recommended since you have watched Rush Hour acted by the same actor Tom Wilkinson*

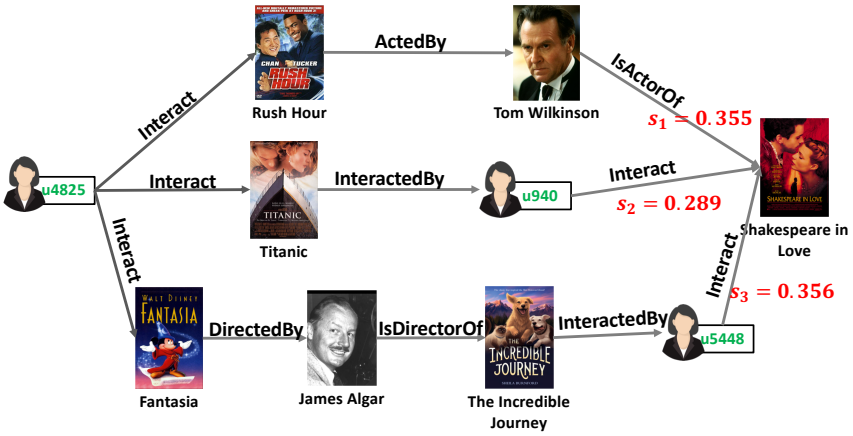


Figure 6.5: Visualization of three paths with prediction scores for the user of u4825 in MI dataset. The prediction scores are normalized for illustration.

or *since it is similar to Titanic that you watched before.* This case demonstrates KPRN's capacity of providing informative explanations.

6.4 Conclusions

In this work, we exploit knowledge graph to construct paths as extra user-item connectivity, which is complementary to user-item interactions. We propose a knowledge-aware path recurrent network to generate representation for each path by composing semantics of entities and relations. By adopting LSTM on paths, we can capture the sequential dependencies of elements and reason on paths to infer user preference. Extensive experiments are performed to show the effectiveness and explainability of our model.

In future, we will extend our work in two directions. First, we attempt to mimic the propagation process of user preferences within KGs via Graph Neural Networks, since extracting qualified paths is labor-intensive. Second, as KG links multiple domains (*e.g.,* movie and book) together with overlapped entities, we plan to adopt zero-shot learning to solve the cold start issues in the target domain.

Chapter 7

Conclusion

In this thesis, we investigated how rich auxiliary information, especially across different channels, is beneficial for personalized recommendation. Various types of auxiliary data are considered, including online and offline, information- and social-oriented, user- and item-centric, and knowledge-aware channels, towards enriching the connections between users and items. We have proposed new methods to incorporate cross-channel information into recommender systems, and demonstrated their effectiveness and explainability on real-world datasets.

7.1 Conclusion

In this thesis, we explored the following scenarios that exploit different cross-channel information,

- We explore the co-existence and relations of users' online and offline behaviors, which profile users from different angles, towards bridging the channel gap in the event recommendation task. We proposed a dual-clustering method to detect users communities, where global and local consistency constraints are adopted to model user representations better.
- We investigate how to route items from information-oriented platforms to users within social-oriented networks. Particularly, we proposed an attribute-aware method, which uses attributes of users and items to enrich their representations; and the representations of bridge users are used to guide the preference modeling of social users.

- We conduct the study on explainable recommendation based on cross features, which consist of user- and item-centric attributes. We proposed a tree-enhanced embedding method that combines the explainability of tree-based methods and the strong expressiveness of embedding-based methods.

- We incorporate external knowledge graph into recommender systems to investigate knowledge-aware recommendation. A graph-based method is proposed to capture user preference towards historical items and knowledge-aware entities attentively.

We conducted extensive experiments on real-world datasets to conclusively demonstrate the effectiveness of the above proposed methods.

7.2 Future Work

Exploiting auxiliary information is an important task in recommendation tasks. We believe future work can be extended in different directions, from both information source and technique views.

In this thesis, we have focused on categorical data (*e.g.,* user age, gender, and item price) and structural information (*e.g.,* social network and knowledge graph), but overlooked other types of auxiliary information. We thus plan to extend our work by considering other multimedia contexts, such as user comments, item images, and micro-videos. For example, user comments not only indicate their interaction with items, but also explicitly reflect user sentiments towards some properties of items; and when conducting fashion recommendation, visual features encode more signals than textual descriptions. Hence, we would like to use multi-modal features to boost recommendation performance.

Another direction for future work is to investigate the evolution of user preferences, *i.e.,* sequential recommendation. That is, when new user-item interactions stream in, the user preference should be updated in time. In this thesis, we only focus on the static user-item interactions, failing to predict successive items based on users' past records and capture the sequential patterns. We thus would like to use auxiliary data to represent users in an explicit fashion and adopt sequential neural model to model dynamic user preference over time.

From the technical point of views, CF-based methods suffer from the cold-

start problems since they solely model the edges between users and items, but fail to model the information propagation or message passing within the bipartite user-item graph. To solve this issue, existing work focus on involving more side information to enrich the representations of users and items, still in the CF fashion. We thus would like to mimic information passing along with the structure of user-item graphs.

While cross-channel recommendation is promising to boost the performance, incorporating heterogeneous data might be a double edged sword, since privacy concerns would come up. To be specific, the huge amount of personal data (*e.g.,* demographics, behaviors, and preferences) accessible online may put users at a high risk of privacy leakage. Hence, it is of crucial importance to avoid privacy leakage during cross-channel recommendation.

Bibliography

[1] B. Abdollahi and O. Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. 2016.

[2] B. Abdollahi and O. Nasraoui. Using explainability for constrained matrix factorization. In *RecSys*, pages 79–83, 2017.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.

[4] D. Agarwal and B. Chen. Regression-based latent factor models. In *SIGKDD*, pages 19–28, 2009.

[5] T. Alashkar, S. Jiang, S. Wang, and Y. Fu. Examples-rules guided deep neural network for makeup recommendation. In *AAAI*, pages 941–947, 2017.

[6] N. X. Bach, N. D. Hai, and T. M. Phuong. Personalized recommendation of stories for commenting in forum-based social media. *Inf. Sci.*, 352-353:48–60, 2016.

[7] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[8] M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

[9] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, pages 2–8, 2014.

[10] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI*, pages 714–720, 1998.

[11] I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, pages 1341–1350, 2017.

[12] I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, pages 1341–1350, 2017.

[13] V. Bellini, V. W. Anelli, T. D. Noia, and E. D. Sciascio. Auto-encoding user ratings via knowledge graphs in recommendation scenarios. In *DLRS@RecSys*, pages 60–66, 2017.

[14] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[15] M. B. Blaschko and C. H. Lampert. Correlational spectral clustering. In *CVPR*, pages 1–8, 2008.

[16] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *KDD*, pages 1258–1266, 2012.

[17] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[18] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[19] E. Bruno and S. Marchand-Maillet. Multiview clustering: a late fusion approach using latent models. In *SIGIR*, pages 736–737, 2009.

[20] X. Cai, F. Nie, and H. Huang. Multi-view k-means clustering on big data. In *IJCAI*, pages 2598–2604, 2013.

[21] X. Cai, F. Nie, H. Huang, and F. Kamangar. Heterogeneous image feature integration via multi-modal spectral clustering. In *CVPR*, pages 1977–1984, 2011.

[22] I. Cantador, A. Bellogín, and D. Vallet. Content-based recommendation in social tagging systems. In *RecSys*, pages 237–240, 2010.

[23] R. Catherine and W. W. Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *RecSys*, pages 325–332, 2016.

[24] S. Chaudhari, A. Azaria, and T. M. Mitchell. An entity graph based recommender system. In *RecSys*, 2016.

[25] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan. Multi-view clustering via canonical correlation analysis. In *ICML*, pages 129–136, 2009.

[26] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T. Chua. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR*, pages 335–344, 2017.

[27] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794, 2016.

[28] T. Chen, L. Tang, Q. Liu, D. Yang, S. Xie, X. Cao, C. Wu, E. Yao, Z. Liu, Z. Jiang, et al. Combining factorization model and additive forest for collaborative followee recommendation. *KDD CUP*, 2012.

[29] W. Chen, W. Hsu, and M. Lee. Making recommendations from multiple domains. In *SIGKDD*, pages 892–900, 2013.

[30] Y. Chen, X. Zhao, and M. de Rijke. Top-n recommendation with high-dimensional side information via locality preserving projection. In *SIGIR*, pages 985–988, 2017.

[31] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. Wide & deep learning for recommender systems. *DLRS*, abs/1606.07792, 2016.

[32] 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 *DLRS*, pages 7–10, 2016.

[33] W. Cheng, X. Zhang, Z. Guo, Y. Wu, P. F. Sullivan, and W. Wang. Flexible and robust co-regularized multi-domain graph clustering. In *KDD*, pages 320–328, 2013.

[34] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *WWW*, 2018.

[35] Z. Cheng, Y. Ding, L. Zhu, and M. S. Kankanhalli. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *WWW*, pages 639–648, 2018.

[36] Z. Cheng and J. Shen. On effective location-aware music recommendation. *TOIS*, 34(2):13:1–13:32, 2016.

[37] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*, pages 153–162, 2007.

[38] F. Chierichetti, N. N. Dalvi, and R. Kumar. Correlation clustering in mapreduce. In *KDD*, pages 641–650, 2014.

[39] W. Chu and Y. Tsai. A hybrid recommendation system considering visual information for predicting favorite restaurants. *WWW*, 20(6):1313–1331, 2017.

[40] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combing content-based and collaborative filters in an online newspaper. 1999.

[41] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[42] P. Cui, F. Wang, S. Liu, M. Ou, S. Yang, and L. Sun. Who should share what?: item-level social influence prediction for users and posts ranking. In *SIGIR*, pages 185–194, 2011.

[43] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280, 2007.

[44] D. L. Davies and D. W. Bouldin. A cluster separation measure. *TPAMI*, 1(2):224–227, 1979.

[45] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 107–144. 2011.

[46] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD*, pages 551–556, 2004.

[47] Q. Diao, M. Qiu, C. Wu, A. J. Smola, J. Jiang, and C. Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *KDD*, pages 193–202, 2014.

[48] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pages 135–144, 2017.

[49] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[50] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*, pages 278–288, 2015.

[51] A. Farseev, I. Samborskii, A. Filchenkov, and T.-S. Chua. Cross-domain recommendation via clustering on multi-layer graphs. In *SIGIR*, 2017.

[52] F. Feng, X. He, Y. Liu, L. Nie, and T. Chua. Learning on partial-order hypergraphs. In *WWW*, 2018.

[53] F. Feng, L. Nie, X. Wang, R. Hong, and T.-S. Chua. Computational social indicators: a case study of chinese university ranking. In *SIGIR*, 2017.

[54] J. Foley, M. Bendersky, and V. Josifovski. Learning to extract local events from the web. In *SIGIR*, pages 423–432, 2015.

[55] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[56] M. Friedman and A. Kandel. *Introduction to Pattern Recognition - Statistical, Structural, Neural and Fuzzy Logic Approaches*, volume 32 of *Series in Machine Perception and Artificial Intelligence*. WorldScientific, 1999.

[57] J. Gao, J. Han, J. Liu, and C. Wang. Multi-view clustering via joint nonnegative matrix factorization. In *ICDM*, pages 252–260, 2013.

[58] L. Gao, H. Yang, J. Wu, C. Zhou, W. Lu, and Y. Hu. Recommendation with multi-source heterogeneous information. In *IJCAI*, pages 3378–3384, 2018.

[59] A. Ghoting, P. Kambadur, E. P. D. Pednault, and R. Kannan. NIMBLE: a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In *KDD*, pages 334–342, 2011.

[60] D. Greene and P. Cunningham. A matrix factorization approach for integrating multiple data views. In *ECML PKDD*, pages 423–438, 2009.

[61] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. Deepfm: A factorization-machine based neural network for CTR prediction. In *IJCAI*, pages 1725–1731, 2017.

[62] C. A. Halcrow, L. Carr, and S. Halford. Using the SPENCE model of online/offline community to analyse sociality of social machines. In *WWW*, pages 769–774, 2016.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[64] X. He, T. Chen, M. Kan, and X. Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670, 2015.

[65] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[66] X. He, M. Kan, P. Xie, and X. Chen. Comment-based multi-view clustering of web 2.0 items. In *WWW*, pages 771–782, 2014.

[67] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[68] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*, pages 5:1–5:9, 2014.

[69] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.

[70] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, pages 241–250, 2000.

[71] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[72] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW*, pages 595–606, 2013.

[73] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*, pages 505–514, 2018.

[74] M. Jamali and M. Ester. *TrustWalker*: a random walk model for combining trust-based and item-based recommendation. In *KDD*, pages 397–406, 2009.

[75] P. James, Y. Nadarajah, K. Haive, and V. Stead. *Sustainable communities, sustainable development: Other paths for Papua New Guinea*. University of Harwaii Press, 2012.

[76] M. Jiang, P. Cui, X. Chen, F. Wang, W. Zhu, and S. Yang. Social recommendation with cross-domain transferable knowledge. *TKDE*, 27(11):3084–3097, 2015.

[77] M. Jiang, P. Cui, R. Liu, Q. Yang, F. Wang, W. Zhu, and S. Yang. Social contextual recommendation. In *CIKM*, pages 45–54, 2012.

[78] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang. Scalable recommendation with social contextual information. *TKDE*, 26(11):2789–2802, 2014.

[79] M. Kompan and M. Bieliková. Content-based news recommendation. In *EC-Web*, pages 61–72, 2010.

[80] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[81] Y. Koren and R. M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 77–118. 2015.

[82] A. Kumar and H. D. III. A co-training approach for multi-view spectral clustering. In *ICML*, pages 393–400, 2011.

[83] A. Kumar, P. Rai, and H. D. III. Co-regularized multi-view spectral clustering. In *NIPS*, pages 1413–1421, 2011.

[84] W. Lei, X. Jin, M.-Y. Kan, Z. Ren, X. He, and D. Yin. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL*, volume 1, pages 1437–1447, 2018.

[85] K. W. Leung, D. L. Lee, and W. Lee. CLR: a collaborative location recommendation framework based on co-clustering. In *SIGIR*, pages 305–314, 2011.

[86] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, pages 2052–2057, 2009.

[87] L. Li, D. Wang, T. Li, D. Knox, and B. Padmanabhan. SCENE: a scalable two-stage personalized news recommendation system. In *SIGIR*, pages 125–134, 2011.

[88] X. Li, G. Cong, X. Li, T. N. Pham, and S. Krishnaswamy. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In *SIGIR*, pages 433–442, 2015.

[89] X. Li, M. K. Ng, and Y. Ye. HAR: hub, authority and relevance scores in multi-relational data for query search. In *SIAM*, pages 141–152, 2012.

[90] X. Li, M. K. Ng, and Y. Ye. Multicomm: Finding community structure in multi-dimensional networks. *TKDE*, 26(4):929–941, 2014.

[91] X. Li and J. She. Collaborative variational autoencoder for recommender systems. In *KDD*, pages 305–314, 2017.

[92] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *WWW*, pages 689–698, 2018.

[93] L. Liao, X. He, H. Zhang, and T.-S. Chua. Attributed social network embedding. *arXiv preprint arXiv:1705.04969*, 2017.

[94] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[95] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[96] X. Ling, W. Deng, C. Gu, H. Zhou, C. Li, and F. Sun. Model ensemble for click prediction in bing search ads. In *WWW*, pages 689–698, 2017.

[97] H. Liu, T. Liu, J. Wu, D. Tao, and Y. Fu. Spectral ensemble clustering. In *KDD*, pages 715–724, 2015.

[98] X. Liu, Q. He, Y. Tian, W. Lee, J. McPherson, and J. Han. Event-based social networks: linking the online and offline social worlds. In *KDD*, pages 1032–1040, 2012.

[99] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of internal clustering validation measures. In *ICDM*, pages 911–916, 2010.

[100] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. 2011.

[101] H. Ma. An experimental study on implicit social recommendation. In *SIGIR*, pages 73–82, 2013.

[102] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, pages 287–296, 2011.

[103] J. J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172, 2013.

[104] A. McCallum, A. Neelakantan, R. Das, and D. Belanger. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*, pages 132–141, 2017.

[105] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI*, pages 187–192, 2002.

[106] G. D. F. Morales, A. Gionis, and C. Lucchese. From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *WSDM*, pages 153–162, 2012.

[107] C. C. Musat, Y. Liang, and B. Faltings. Recommendation using textual opinions. In *IJCAI*, pages 2684–2690, 2013.

[108] C. Musto, G. Semeraro, P. Lops, and M. de Gemmis. Combining distributional semantics and entity linking for context-aware content-based recommendation. In *UMAP*, pages 381–392, 2014.

[109] A. Neelakantan, B. Roth, and A. McCallum. Compositional vector space models for knowledge base completion. In *ACL*, pages 156–166, 2015.

[110] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[111] M. K. Ng, X. Li, and Y. Ye. Multirank: co-ranking for objects and relations in multi-relational data. In *KDD*, pages 1217–1225, 2011.

[112] N. Nguyen and R. Caruana. Consensus clusterings. In *ICDM*, pages 607–612, 2007.

[113] J. Ni, H. Tong, W. Fan, and X. Zhang. Flexible and robust multi-network clustering. In *KDD*, pages 835–844, 2015.

[114] L. Nie, X. Song, and T. Chua. *Learning from Multiple Social Networks*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2016.

[115] L. Nie, M. Wang, Z. Zha, and T. Chua. Oracle in image search: A content-based approach to performance prediction. *TOIS*, 30(2):13:1–13:23, 2012.

[116] L. Nie, M. Wang, Z. Zha, G. Li, and T. Chua. Multimedia answering: enriching text QA with media information. In *SIGIR*, pages 695–704, 2011.

[117] L. Nie, S. Yan, M. Wang, R. Hong, and T. Chua. Harvesting visual concepts for image search with complex queries. In *MM*, pages 59–68, 2012.

[118] L. Nie, Y. Zhao, M. Akbari, J. Shen, and T. Chua. Bridging the vocabulary gap between health seekers and healthcare knowledge. *TKDE*, 27(2):396–409, 2015.

[119] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *RecSys*, pages 155–162, 2012.

[120] S. Oramas, V. C. Ostuni, T. D. Noia, X. Serra, and E. D. Sciascio. Sound and music recommendation with knowledge graphs. *TIST*, 8(2):21:1–21:21, 2017.

[121] E. Palumbo, G. Rizzo, and R. Troncy. entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *RecSys*, pages 32–36, 2017.

[122] S. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, pages 21–28, 2009.

[123] S. Park, D. M. Pennock, O. Madani, N. Good, and D. DeCoste. Naïve filterbots for robust cold-start recommendations. In *KDD*, pages 699–705, 2006.

[124] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[125] S. Pero and T. Horváth. Opinion-driven matrix factorization for rating prediction. In *User Modeling, Adaptation, and Personalization - 21th International Conference, UMAP*, pages 1–13, 2013.

[126] T. N. Pham, X. Li, G. Cong, and Z. Zhang. A general graph-based model for recommendation in event-based social networks. In *ICDE*, pages 567–578, 2015.

[127] T. N. Pham, X. Li, G. Cong, and Z. Zhang. A general recommendation model for heterogeneous networks. *TKDE*, 28(12):3140–3153, 2016.

[128] G. Qi, C. C. Aggarwal, and T. S. Huang. Community detection with edge content in social media networks. In *ICDE*, pages 534–545, 2012.

[129] Z. Ren, S. Liang, P. Li, S. Wang, and M. de Rijke. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*, pages 485–494, 2017.

[130] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[131] S. Rendle. Factorization machines with libfm. *TIST*, 3(3):57:1–57:22, 2012.

[132] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[133] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[134] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.

[135] I. Ronen, I. Guy, E. Kravi, and M. Barnea. Recommending social media content to community owners. In *SIGIR*, pages 243–252, 2014.

[136] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, pages 53–65, 1987.

[137] S. Sahebi and P. Brusilovsky. Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In *UMAP*, pages 289–295, 2013.

[138] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[139] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *KDD*, pages 255–262, 2016.

[140] A. Sharma and D. Cosley. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *WWW*, pages 1133–1144, 2013.

[141] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *CIKM*, pages 453–462, 2015.

[142] Z. Shu, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, and C. Xu. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*, 2018.

[143] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, pages 926–934, 2013.

[144] X. Song, Z. Ming, L. Nie, Y. Zhao, and T. Chua. Volunteerism tendency prediction via harvesting multiple social networks. *TOIS*, 34(2):10, 2016.

[145] X. Song, L. Nie, L. Zhang, M. Akbari, and T. Chua. Multiple social network learning and its application in volunteerism tendency prediction. In *SIGIR*, pages 213–222, 2015.

[146] X. Song, L. Nie, L. Zhang, M. Liu, and T. Chua. Interest inference via structure-constrained multi-source multi-task learning. In *IJCAI*, pages 2371–2377, 2015.

[147] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[148] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, pages 111–120, 2009.

[149] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[150] G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *RecSys*, pages 83–90, 2012.

[151] J. Tang, S. Wu, J. Sun, and H. Su. Cross-domain collaboration recommendation. In *SIGKDD*, pages 1285–1293, 2012.

[152] L. Tang and H. Liu. Uncovering cross-dimension group structures in multi-dimensional networks. In *SDM*, pages 568–575, 2009.

[153] L. Tang, H. Liu, and J. Zhang. Identifying evolving groups in dynamic multimode networks. *TKDE*, 24(1):72–85, 2012.

[154] N. Tintarev. Explanations of recommendations. In *RecSys*, pages 203–206, 2007.

[155] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. *Recommender Systems Handbook*, pages 479–510, 2011.

[156] T. X. Tuan and T. M. Phuong. 3d convolutional networks for session-based recommendation with content features. In *RecSys*, pages 138–146, 2017.

[157] J. Vig, S. Sen, and J. Riedl. Tagsplanations: explaining recommendations using tags. In *IUI*, pages 47–56, 2009.

[158] C. Wang, R. Raina, D. Fong, D. Zhou, J. Han, and G. J. Badros. Learning relevance from heterogeneous social network and its application in online targeting. In *SIGIR*, pages 655–664, 2011.

[159] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*, pages 417–426, 2018.

[160] H. Wang, F. Zhang, X. Xie, and M. Guo. DKN: deep knowledge-aware network for news recommendation. *WWW*, abs/1801.08284, 2018.

[161] S. Wang, C. Aggarwal, and H. Liu. Randomized feature engineering as a fast and accurate alternative to kernel methods. In *SIGKDD*, pages 485–494, 2017.

[162] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *WWW*, pages 391–400, 2017.

[163] X. Wang and I. Davidson. Flexible constrained spectral clustering. In *KDD*, pages 563–572, 2010.

[164] X. Wang, X. He, F. Feng, L. Nie, and T. Chua. TEM: tree-enhanced embedding model for explainable recommendation. In *WWW*, pages 1543–1552, 2018.

[165] X. Wang, X. He, L. Nie, and T. Chua. Item silk road: Recommending items from information domains to social users. In *SIGIR*, pages 185–194, 2017.

[166] X. Wang, L. Nie, X. Song, D. Zhang, and T.-S. Chua. Unifying virtual and physical worlds: Learning toward local and global consistency. *TOIS*, 36(1):4, 2017.

[167] C. Wartena, W. Slakhorst, and M. Wibbels. Selecting keywords for content based recommendation. In *CIKM*, pages 1533–1536, 2010.

[168] F. L. Wauthier, N. Jojic, and M. I. Jordan. Active spectral clustering via iterative uncertainty reduction. In *KDD*, pages 1339–1347, 2012.

[169] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, pages 3119–3125, 2017.

[170] Y. Yang, C. Lan, X. Li, B. Luo, and J. Huan. Automatic social circle detection using multi-view clustering. In *CIKM*, pages 1019–1028, 2014.

[171] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang. Image clustering using local discriminant models and global integration. *TIP*, 19(10):2761–2773, 2010.

[172] S. X. Yu and J. Shi. Multiclass spectral clustering. In *ICCV*, pages 313–319, 2003.

[173] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI*, 27, 2013.

[174] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: a heterogeneous information network approach. In *WSDM*, pages 283–292, 2014.

[175] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *KDD*, pages 797–802, 2006.

[176] D. Zhai, H. Chang, S. Shan, X. Chen, and W. Gao. Multiview metric learning with global consistency and local smoothness. *TISIT*, 3(3):53, 2012.

[177] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. M. Kaplan, S. Wang, and J. Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *SIGIR*, pages 513–522, 2016.

[178] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362, 2016.

[179] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T. Chua. Discrete collaborative filtering. In *SIGIR*, pages 325–334, 2016.

[180] J. Zhang, L. Nie, X. Wang, X. He, X. Huang, and T. Chua. Shorter-is-better: Venue category estimation from micro-video. In *MM*, pages 1415–1424, 2016.

[181] S. Zhang, L. Yao, and A. Sun. Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435, 2017.

[182] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92, 2014.

[183] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu. COSNET: connecting heterogeneous social networks with local and global consistency. In *KDD*, pages 1485–1494, 2015.

[184] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *WWW*, pages 1511–1520, 2013.

[185] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *KDD*, pages 635–644, 2017.

[186] Q. Zhao, Y. Shi, and L. Hong. GB-CENT: gradient boosted categorical embedding and numerical trees. In *WWW*, pages 1311–1319, 2017.

[187] Z. Zhao, H. Lu, D. Cai, X. He, and Y. Zhuang. User preference learning for online social recommendation. *TKDE*, 28(9):2522–2534, 2016.

[188] D. Zhou and C. J. C. Burges. Spectral clustering and transductive learning with multiple views. In *ICML*, pages 1159–1166, 2007.

[189] Y. Zhou and L. Liu. Social influence based clustering of heterogeneous information networks. In *KDD*, pages 338–346, 2013.