Exploiting context information in recommendation systems

Pham, Nguyen Tuan Anh

2017

Pham, N.T.A. (2017). Exploiting context information in recommendation systems. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/72740>

<https://doi.org/10.32657/10356/72740>

NANYANG TECHNOLOGICAL UNIVERSITY

Exploiting Context Information in Recommendation Systems

Ph.D Thesis

By

Pham Nguyen Tuan Anh

Supervisor: Prof. Cong Gao

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of Doctor of Philosophy

October, 2017

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Gao Cong for his patience, encouragement, enthusiasm, and sharing of his knowledge. His valuable guidance taught me how to do research, to think critically, and present my work properly. Without his continuous support, I would not be able to finish my PhD study. It is my pleasure and honor to work and study under his mentorship.

I would like to express my gratitude to my friends in our group for their friendship and great collaboration. They include Xutao Li, Xin Cao, Lisi Chen, Quan Yuan, Kaiqi Zhao, Shanshan Feng, Tao Guo, Kaiyu Feng, Zhida Chen and many others who are being and have been supervised by Prof. Gao Cong.

I would like to thank Mr. Loo Kian Hock, Thomas and friends in Data Management and Analytics Graduate Lab (DMAL) for their support during my PhD study.

I would like to thank to my brother, Pham Nguyen Quang Anh, and other PhD friends for sharing research knowledge and experience.

Last but not least, I would like to give my deep gratitude to my family for their unconditional support, encouragement and love.

Abstract

With the rapid growth of data in recent years, especially online and user-generated data, the role of recommendation systems becomes more important. Many recommendation methods have been proposed, among which context-aware recommendation systems have received significant interests from researchers due to their superior performance. Different from traditional recommendation systems, which only consider users and items, context-aware recommendation systems exploit additional information that affects users' decisions on items. Context information is very diverse, from explicit information, such as user profiles (*e.g.*, age, gender) or item description (*e.g.*, weight, price), to implicit one, such as item-item dependency or users' requirements. In this thesis, we consider three context-aware recommendation problems, in which the context information has an essential role on improving recommendation accuracy.

First, we consider the problem of recommendation in heterogeneous networks. Heterogeneous networks are information networks containing different types of entities (*e.g.*, users, items, locations) and relationships between entities (*e.g.*, a user rates an item). Heterogeneous networks can be found in many domains, such as movies (Netflix), music (Last.fm) and social networks (Facebook). Although many context-aware recommendation systems have been proposed for heterogeneous networks, one common shortcoming of those systems is that they cannot explicitly capture and model the interaction between different types of entities. Characterizing and utilizing this information is crucial to gain a good recommendation accuracy because the way each entity type behaves and interacts with others greatly affects the recommendation results. As a result, in this thesis, we propose a general-purpose recommendation model, which is able to solve general recommendation tasks in heterogeneous networks. Different from existing methods, our model is able to explicitly model the interaction between entity types and automatically learn the strength of those interactions. This not only helps our model achieve better performance than state-of-the-art methods, but also enables us to understand roles of entity types in different recommendation problems.

Second, we propose and address a novel problem, namely out-of-town region recommendation. When traveling to new cities or countries, users usually have very limited time to visit places; hence, they tend to visit places in a small region. Therefore, it is more beneficial to recommend a region of point-of-interest (POI) than a list of POIs that locate far away from each other. When users choose a region to visit, one important consideration is whether the POIs in the region are attractive to users as a whole. In particular, users' decisions to visit a POI are also affected by how they are interested in nearby POIs, in addition to the user preference to the POI. Therefore, we propose a general framework for region recommendation in which influences among POIs are taken into consideration. Experiments on real world datasets validate that our proposed model outperforms baseline methods, in which POIs are considered independently.

Third, when submitting recommendation requests, users usually have clear requirements or intention, *e.g.*, having dinner or hanging out. To exploit this information, we propose a model for the problem of requirement-aware POI recommendation. Our model is able to precisely understand user intention by applying attention model, a deep learning model that has been used in many natural language processing tasks, so that we can provide better recommendation for users. Moreover, our model can be easily extended to incorporate additional information such as geographical influence. Empirical studies demonstrate our proposals achieve significant improvement over baseline methods.

In summary, this thesis focuses on exploiting context information in recommendation. Three context-aware recommendation systems are proposed for some specific problems. The solutions are not limited in these problems. They can be applied in other recommendation tasks without much modification. We also discuss some promising directions to improve and extend our techniques in future work.

Contents

Acknowledgments	i
Abstract	ii
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Context Information	2
1.3 Research Problems and Methodologies	5
1.3.1 Recommendation in Heterogeneous Networks	5
1.3.2 Out-of-town Region Recommendation	8
1.3.3 Requirement-aware POI Recommendation	9
1.4 Thesis Organization	10
2 Literature Review	12
2.1 Recommendation Systems	12
2.1.1 Collaborative Filtering	12
2.1.2 Content-based Recommendation	14
2.1.3 Hybrid Recommendation	15
2.2 Context-aware Recommendation	15
2.2.1 Recommendation Systems in Heterogeneous Networks	15
2.2.2 Item-relation-aware Recommendation	17
2.2.3 Requirement-aware Recommendation	19
2.3 POI Recommendation	20
2.3.1 Conventional POI Recommendation	20
2.3.2 Next POI Recommendation	22
2.3.3 Time-aware POI Recommendation	22
2.3.4 In-town/Out-town POI Recommendation	22

3	A General Recommendation Model for Heterogeneous Networks	24
3.1	Proposed Recommendation Model on Heterogeneous Networks	24
3.1.1	Heterogeneous Network Graph	24
3.1.2	Recommendation on Heterogeneous Network Graphs	26
3.2	Optimization Approach for Parameter Learning	30
3.2.1	Objective Function	30
3.2.2	Solving the Optimization Problem	31
3.2.3	Fast Learning Method	33
3.2.4	Approximation Algorithm	34
3.3	Enhanced HeteRS	36
3.4	Experiments	38
3.4.1	Dataset Description	38
3.4.2	Evaluation Metrics	40
3.4.3	Baseline Methods	40
3.4.4	Parameter Settings	42
3.4.5	Group-to-user Recommendation	42
3.4.6	Tag-to-group Recommendation	44
3.4.7	Event-to-user Recommendation	45
3.4.8	Movie-to-user Recommendation	46
3.4.9	Track-to-user Recommendation	47
3.4.10	Interpretation of Transition Parameters	47
3.4.11	Evaluating Fast Learning Method	49
3.4.12	Evaluating Approximation Method	50
3.4.13	Evaluating Enhanced HeteRS	53
3.5	Conclusion	55
4	A General Model for Out-of-town Region Recommendation	56
4.1	Observations	57
4.2	Region Recommendation	59
4.2.1	Proposed Model	59
4.2.2	Searching Maximum Region	61
4.3	Approximate Solution	64
4.3.1	Main Idea	65
4.3.2	Solving Approximate RegRS Problem	65
4.3.3	Approximate Algorithm	71
4.4	Experiments	71

4.4.1	Experimental Settings	71
4.4.2	Evaluation Metrics	72
4.4.3	Baseline Methods	73
4.4.4	Recommendation Results	74
4.4.5	Approximate Algorithm	77
4.5	Conclusion	78
5	Requirement-aware POI recommendation	79
5.1	Requirement-aware POI Recommendation Model	81
5.1.1	Preliminaries	82
5.1.2	Proposed Model	83
5.1.3	Incorporating geographical influence	89
5.2	Experiments	90
5.2.1	Dataset	91
5.2.2	Baseline Methods	92
5.2.3	Experimental Results	93
5.3	Conclusion	97
6	Conclusions and Future Work	98
6.1	Conclusions	98
6.2	Future Work	100
	References	101
	Author's Publications	117

List of Figures

1.1	An example of an event-based social network	6
3.1	a) Event-based Social Network Graph with nodes of users (U), events (E), groups (G), tags (T) and venues (V); b) Extension of EBSN Graph with session nodes (S).	25
3.2	An example of the efficient storage technique with $\xi = 0.1$: a) Original probability vector; b) The vector after removing small entries; c) The vector after reconstruction.	34
3.3	Performance of methods in group-to-user recommendation	43
3.4	Performance of methods in tag-to-group recommendation	44
3.5	Performance of methods in event-to-user recommendation	45
3.6	Performance of methods in movie-to-user recommendation	46
3.7	Performance of methods in track-to-user recommendation	47
3.8	Performance of fast learning method with different thresholds ξ and original learning process in tag-to-group recommendation problem on CA dataset.	50
3.9	Performance of Approximation HeteRS with different parameter ϵ on CA dataset	51
3.10	Comparison between Iterative and Approximation HeteRS in terms of effectiveness and efficiency on CA dataset	52
3.11	Performance of Global HeteRS and Enhanced HeteRS with different number of clusters.	53
4.1	Observations on out-of-town check-ins.	58
4.2	Fraction of regions with different number of visited POIs.	59
4.3	Reduction to GI problem.	63
4.4	The illustration of the sweeping algorithm.	64
4.5	Some observations	66
4.6	Dividing space by grid G_d	67

4.7	Special point and edge arrangements for cell C	69
4.8	Two possible positions of a rectangle c in case 2.	71
4.9	POI distribution of two datasets.	74
4.10	Performance of region recommendation methods on Foursquare dataset.	75
4.11	Performance of region recommendation methods on Twitter dataset.	76
5.1	The architectures of non-attention and attention-based models.	86
5.2	Attention layer	87
5.3	AttReqRS with geographical influence (negative POI k is not shown for saving space)	90
5.4	Performance of methods in requirement-aware POI recommendation	94
5.5	Performance of AttReqRS+G with different numbers of neighbor POIs.	95
5.6	Performance of recommendation methods on users with different num- bers of visited POIs.	96
5.7	Performance of recommendation methods on queries with different num- bers of words.	97

List of Tables

3.1	Meetup dataset	39
3.2	MovieLens and Yahoo! Music datasets	40
3.3	Transition parameters	48
3.4	Compression ratio	50
4.1	Performance comparison between RegRS and Appr-RegRS in terms of Hit, Quality and running time (in seconds).	77
5.1	Yelp dataset	91

Chapter 1

Introduction

1.1 Motivation

Recommendation systems have become an important research area over the last decade. There has been vast amount of research work on recommendation systems since the first papers on collaborative filtering published in 90s [1–3]. Recommendation systems have wide-range applications, which can be found in various domains such as products (Amazon), movies (Netflix) and music (Last.fm). The high demand of developing recommendation systems comes from the fact that users are usually overwhelmed by the huge amount of data, which makes them difficult in finding what they are interested in. Recent years have witnessed the rapid development of Web 2.0 services, such as social networks, blogs, video sharing sites, which makes the volume of data, most of which are user-generated, grow with much faster pace. Therefore, the importance of recommendation systems becomes higher nowadays.

Recommendation techniques can be classified into three categories, namely Collaborative Filtering (CF), content-based approaches and hybrid approaches (which is the combination of the previous two approaches). Among three types of techniques, CF receives more research interest than the other two techniques. CF techniques work based on the assumption that users with similar interests are likely to choose same items in the future. In other words, if a user A shares many common opinions (likes/dislikes or ratings) on same items with user B, user A tends to have B's opinion on an item X that is not decided by user A. Therefore, the main strategy of those techniques is to recommend items to users based on other like-minded users.

One noticeable characteristic of CF techniques is that they only consider two kinds of information, namely users and items, which are encoded as the user-item matrix or rating matrix. This, however, limits the ability of those methods a lot since rating data is not sufficient to make accurate recommendation as user rating behaviors may be also affected by other factors, which are called context. For example, when a user decides to attend an event in an event-based social network [4], besides the topic of the event, she may also consider other factors, such as time or location of the event, to make her decision. As another example, a user is more likely to buy a mouse than a keyboard, given that she already purchased a laptop. In these examples, “time”, “location” and “laptop” are context information that influence the user’s decision to select an item. Since context information is neglected by traditional CF techniques, they may fail to give satisfactory recommendations for users.

Recognizing this limitation, in this thesis, we focus on a special type of recommendation systems, namely context-aware recommendation systems, which take the context information into consideration when generating recommendation. The goal of context-aware recommendation systems is to recommend items that not only satisfy the target user’s interests but also match her context or current situation.

1.2 Context Information

There are many definitions of the term “context” introduced in the area of computer science. However, most of those definitions suffer from either generality (*i.e.*, self-referencing description) or incompleteness [5]. Among them, the definition proposed by Dey [6] is probably the most widely accepted one: “*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and the application, including the user and the applications themselves.*”. This definition has been adopted in many application domains, including context-aware recommendation systems [7, 8]. Researchers also have different opinions on how to categorize context information. For example, Schilit et al. [9] classify context into three categories: *computer context* (*e.g.*, network connectivity, communication costs, printers,

displays), *user context* (e.g., user profile, location, social situation) and *physical context* (e.g., temperature, noise levels). Chen et al. [10] extend this categorization by adding *time* as the fourth type of context. On the other hand, Schmidt et al. [11] propose their context categorization including *user*, *user's social environment*, *tasks*, *location*, *infrastructure*, *time* and *physical conditions*. Perhaps, the most comprehensive categorization of context is the one given by Zimmermann et al. [5], in which context information is divided into 4 main categories:

- *Individuality context* includes any information can be observed about entities. Individuality context is further divided into subcategories:
 - *natural entity context*, which includes the characteristics of not-human entities,
 - *human entity context*, which includes the characteristics of human beings,
 - *artificial entity context*, which includes the characteristics of entities created by human, such as buildings, computers,
 - *group entity context*, which captures the characteristics and interactions of groups of entities.
- *Time context* includes the temporal information,
- *Location context* includes the spatial information,
- *Activity context* can be described as goals, tasks and actions that the entity is involved,
- *Relations context* captures the semantic relations between entities (such as belong-to, parent-of).

We take Zimmermann's categorization as the true category structure of context in context-aware recommendation, since it covers most of pieces of context information which have been used in recommendation systems.

Recently, context-aware recommendation systems have received a lot of research interest (*e.g.*, [12–20]). Many types of context information have been exploited by recommendation systems, such as location context [17], time context [18] or relations context [19]. In this scope of the thesis, we do not have ambition to cover all types of context information that have been utilized in recommendation systems, but focus on some context categories, some of which have not received much interest from researchers.

- **Natural entity+Human Entity+Time+Location Context.** This set of context has been exploited in most of existing work on recommendation systems. It contains all the information that can be related to users (*e.g.* gender, age), items (*e.g.* location, price) or both (*e.g.* time when a user selects an item). They, together with rating data, form a complex system of heterogeneous data. We call this system a heterogeneous information network as it contains different types of entities (*e.g.*, users, items, locations, time) and their interactions (*e.g.*, users’ ratings on items, locations of items).
- **Group Entity Context.** When coming together, items may have influence on each other in a such way that the user’s interest on an item is changed in the presence of other item. As in above example, a user is unlikely to buy a keyboard but more likely to buy a mouse if she has purchased a laptop recently. This phenomenon has been captured in some e-commerce recommendation systems [21, 22]. In this thesis, we study this type of context in a different recommendation problem.
- **Activity Context.** When submitting recommendation requests, users usually have their own intentions or requirements. In this case, recommendation should be made to satisfy both users’ interests and their requirements. For example, when a user is looking for a place for a birthday party, the system should recommend restaurants or coffee shops, rather than museums or libraries. Or if a user is interested in romantic movies, it is not wise to recommend her horror

movies at the same time. This important factor is usually ignored by recommendation systems, which makes the recommended items less useful for users. Note that the requirement-aware recommendation is different from profile-based recommendation as requirements are only shown at the recommendation time (*e.g.*, looking for place for a party), while user profiles are more static and less frequently change.

1.3 Research Problems and Methodologies

Corresponding to types of context information described above, in this thesis, we study three research problems, namely recommendation in heterogeneous networks, out-of-town region recommendation and requirement-aware POI recommendation. Below, we will present three problems, and approaches to address them.

1.3.1 Recommendation in Heterogeneous Networks

Heterogeneous network can be found in many domains, including movies [23], restaurants [24], music [25], etc. In contrast to other networks, heterogeneous networks consist of multiple types of objects and relationships. For instance, Fig. 1.1 illustrates an example of a heterogeneous network, namely event-based social network (EBSN), which is a new type of social networks that connect people through events [4]. The heterogeneous network contains five types of entities: users, events, groups, tags and venues, together with their relations.

With the rich interaction information available, heterogeneous networks have received a lot of research interests in many topics, one of which is recommendation [24–26] since it has many real applications in heterogeneous networks. For example, in event-based social networks, several problems can be raised: (i) Which groups would a particular user like to join? (ii) Which tags might a group choose when constructing its profiles? (iii) Who will attend an upcoming event? Answering these questions is necessary in order to predict user activities when they participate in an event-based social network. As a matter of fact, these questions require us to design

Figure 1.1: An example of an event-based social network

recommendation systems for different tasks, which are recommend groups to users, tags to groups and events to users in this example.

Many recommendation methods have been developed for heterogeneous networks, which can be generally categorized into three types, *i.e.*, graph-based approaches (*e.g.*, [25, 27–29]), latent factor based approaches (*e.g.*, [13, 19, 26, 30, 31]) and topic model based approaches (*e.g.*, [32–34]). However, to our knowledge, none of those recommendation systems is able to explicitly determine the influence strength between different types of entities (to be introduced and analyzed in Section 2.2.1). This limits a lot the ability of those systems to exploit the variety of the data to improve the recommendation performance because the way each entity type behaves and interacts with others greatly affects the recommendation results. Another limitation of those systems is that some of them are designed for a particular problem or for a particular domain [32–34], which makes it difficult, even impossible, to apply them to other problems and other domains.

In this thesis, we propose a *general-purpose Heterogeneous graph-based Recommendation System* model (HeteRS), which can solve general tasks of recommendation in heterogeneous networks. Particularly, we construct a heterogeneous graph to model the interactions between multiple entities (*e.g.*, users, events, groups, and tags), where entities are represented as different types of nodes and the interactions between them are represented as different types of edges. We then convert the recommendation problem into a node proximity calculation problem on the heterogeneous graph. As our model makes no other assumptions except for knowing the topological structure of a heterogeneous network, it is thus applicable to any recommendation task.

The key challenge to evaluate the node proximity lies at that our heterogeneous graph contains multiple types of entities and they influence each other via different types of interactions. It is difficult to know the importance of these influences for proximity calculation. Moreover, the importance of them may vary from one recommendation problem to another even if they are on the same graph.

In our HeteRS, we employ multivariate Markov chain (MMC) to calculate the node proximity w.r.t. some query nodes, *e.g.* user for which we recommend, in the heterogeneous graph for recommendations. However, existing MMC based methods either (i) need to manually set the influence weights between different types of entities [29, 35, 36], which is tedious and makes these methods less attractive when multiple types of entities exist, as in our case; or (ii) learn the model parameters from transition sequences sampled from data [37–39], which are not available in recommendation problems. To overcome these problems, we propose an optimization framework to automatically learn the influence weights by using a ranking method. In particular, our learning scheme tries to find the optimal set of weights by maximizing the AUC measure¹ on training data, which helps our model to generate the appropriate ranking over recommended items.

¹Area Under the ROC Curve: a ranking-based measure which is the expected proportion of positives ranked before a uniformly drawn random negative.

1.3.2 Out-of-town Region Recommendation

Recent years have witnessed a rapid development of location-based social networks (LBSNs), such as Foursquare and Facebook Places. In these online services, users are able to share their locations and experiences, by checking in via mobile devices, when visiting points of interest (POIs), *e.g.*, restaurants, museums. As of December 2015, Foursquare had 55 million monthly active users and totally 7 billion check-ins. Being overwhelmed by the huge amount of data from LBSNs, users usually encounter difficulties in finding or searching interesting POIs. Motivated by the problem, recently, there is a growing interest in POI recommendation problem, such as [40–43]. Success in POI recommendation is important as it helps users explore new locations, which benefits both users and LBSN services.

POI recommendation is even more important when a user travels to a new place, *e.g.*, new city or country, where she has very little, or even no, knowledge about her destination. This leads to a new problem, namely out-of-town recommendation, which aims to find POIs that a given user may be interested in when she travels out of her hometown. Out-of-town recommendation has received little research interest. Existing work on this problem mainly focus on improving the recommendation accuracy on individual POIs [41, 44, 45]. However, we argue that it is often more beneficial to recommend a set of nearby locations to users rather than just individual locations. This is because, when traveling to new cities, users usually have very limited time to visit places. Investigations on Foursquare data show that a large proportion of users visited multiple POIs resided in small regions, and this proportion is higher when the visiting duration is limited (details of these findings are presented in Section 4.1). Hence, in this thesis, we introduce a new problem, namely Region Recommendation, to recommend out-of-town regions to users. Specifically, a region is preferred if it contains more POIs that the given user would like to visit. To the best of our knowledge, this is the first work on region recommendation for out-of-town users.

Previous work on POI search and recommendations finds that an interesting POI whose nearby POIs are also interesting to a user is preferable when compared to a POI

without interesting nearby POIs. In other words, nearby POIs reinforce each other to make them more attractive to users than separate ones. For example, when searching locations, Cao et al. [46] demonstrate that a relevant location is more interesting to users if it is surrounded by other relevant locations, and propose algorithms that propagate prestige scores among nearby spatial objects. In POI recommendation, Liu et al. [47] observe that nearby POIs tend to share more common visitors, and hence their proposed factorization model tries to learn similar preference for nearby POIs. Similarly, Li et al. [43] also exploit the neighborhood effect when recommending POIs to users. The signal of this effect is even stronger when users travel as they usually have limited time to visit POIs in a small region, making it beneficial for region recommendation. Inspired by these studies, in this thesis, we propose a general framework to recommend regions to users that takes the POI influences into consideration. Different from previous work, our model considers the neighborhood effect collectively on a group of POIs, rather than individual ones.

1.3.3 Requirement-aware POI Recommendation

Although there are many research studies on POI recommendation, most of them do not consider the user intent when making recommendation. When users request POI recommendation, they usually have their own intent or requirements, such as dining or hanging out. These requirements are important because they explicitly express users' preferences, and hence useful for recommendation. As a result, in this dissertation, we study the requirement-aware POI recommendation.

Before visiting a POI, a user may have many requirements for the POI. For example, when looking for a restaurant, the user wish that it has seafood menu, comfortable space, low food price and pleasing service, etc. Although users have many requirements for a venue, not all of them are equally important. Taking the example of searching restaurants above, some users may prefer places with comfortable space over food prices, while other users often choose venues because of its reasonable prices. Overall, a user has different requirements when visiting POIs, and she also has unequal levels of interests on requirements. As a result, the challenge is how we

can understand what a user wants the most, *i.e.* user intent, when she visits a POI. Although many context-aware recommendation models in literature can be applied for the requirement-aware POI recommendation problem, no model can exploit the user queries (*i.e.* requirements) adequately.

Recent years have witnessed the rapid development of deep learning based models, some of which have been applied for recommendation [48–51]. Among deep learning methods, attention methods have been successfully applied in many machine learning tasks such as machine translation [52] or image/video captioning [53–55]. Attention mechanism is based on assumption that human only focus on specific parts of their visual inputs to compute the adequate responses. This principle has a large impact on neural computation as we need to select the most pertinent piece of information, rather than using all available information, a large part of it is irrelevant to compute the neural response. This assumption is applicable in our problem, where users only focus on few important requirements in their queries. As a result, we develop an attention-based recommendation model, that utilizes the attention mechanism to process user queries to understand user intent more precisely. Despite their success in many applications, attention models are still new to recommendation systems and have not yet been exploited extensively for recommendation. To the best of our knowledge, the work from [56] is the first one proposing an attention model for recommendation. Different from this work, where attention model is used to learn the user attention preference on different items and different item components, we use attention model to explore user intent when visiting POIs. Moreover, our model is flexible to incorporate other types of information, such as POI information like geographical influence, to improve its recommendation performance.

1.4 Thesis Organization

In Chapter 2, we review previous work on context-aware recommendation. In Chapter 3, we present our model for recommending in heterogeneous networks. Next, we introduce our method for out-town region recommendation in Chapter 4. Chapter 5

presents the proposed model for requirement-aware POI recommendation. Finally, in Chapter 6, we conclude the thesis and discuss directions of future work.

Chapter 2

Literature Review

In this chapter, we first review some recommendation techniques. Next we focus on three types of recommendation systems that exploit contextual information, namely heterogeneous, bundle and requirement-aware recommendation systems. Since some of our work is related to POI recommendation, we also review this topic in this chapter.

2.1 Recommendation Systems

Recommendation Systems are a type of information filtering systems whose goal is to find and suggest items for users based on their preferences. Recommendation Systems have become more and more common in recent years, and are utilized in many areas: such as movies, music, news, and books. Recommendation techniques usually can be classified into one of three categories, namely Collaborative Filtering (CF), content-based approaches and hybrid approaches. In this section, we make a brief review on three types of recommendation techniques, and then focus on context-aware recommendation systems.

2.1.1 Collaborative Filtering

The central idea of CF is that if two users have similar rating behaviors, they are likely to rate same items in the future. Hence, CF recommends items to users based on the ratings of like-minded users, *i.e.*, users with similar preferences on past ratings. Many CF models have been proposed, which can be classified into two types: memory-based CF and model-based CF. They are different in the way to infer users' preferences.

2.1.1.1 Memory-based Collaborative Filtering

Memory-based CF takes the historical ratings of users (items) as user (item) preferences and makes the recommendation based on similar users (items). Two types of memory-based CF approaches are user-based and item-based CF.

Given a user u , User-based CF first computes the similarities between u and each of other users, and then generates ratings for all items that have not been rated by u based on the ratings and computed similarities of other users. Formally, let U and I be the set of users and items, respectively, and I_u be the set of items rated by user u . The rating of u for an item $i \in I \setminus I_u$ is predicted as follows:

$$\hat{r}_{u,i} = \frac{\sum_{u' \in U} w_{u,u'} \cdot r_{u',i}}{\sum_{u' \in U} w_{u,u'}}, \quad (2.1)$$

where $w_{u,u'}$ is the similarity score between two users u and u' . In other words, the predicted rating of u on i is the average of ratings on i from other users weighted by user similarities. The similarity score $w_{u,u'}$ can be computed using cosine similarity or Pearson correlations formulation.

On the other hand, Item-based CF [57] predicts user u 's rating on an item i based on her ratings on other similar items. The predicted rating is thus computed as follows:

$$\hat{r}_{u,i} = \frac{\sum_{i' \in I_u} w_{i,i'} \cdot r_{u,i'}}{\sum_{i' \in I_u} w_{i,i'}}, \quad (2.2)$$

where $w_{i,i'}$ is the similarity between two items i and i' , which is computed based on their ratings from all users.

2.1.1.2 Model-based Collaborative Filtering

This type of CF techniques builds models using data mining, machine learning algorithms based on training data. The main approach of model-based CF models is to uncover latent factors that explain observed ratings. Model-based CF models usually apply classification or clustering algorithms to identify users/items on training set. Many model-based CF techniques have been proposed, two of which are most widely used, namely matrix factorization and topic modeling.

- Matrix factorization (MF) [58, 59] represents users and items by vectors in a k -dimensional space by decomposing the user-item rating matrix R into $|U| \times k$ user matrix P and $|I| \times k$ item matrix Q . Each row of matrix P and Q represents k latent factors of one user and item, respectively. The predicted rating matrix $\hat{R}$ is approximated by the product of P and Q , *i.e.*, $\hat{R} \approx P \times Q^\top$. MF techniques usually achieve better recommendation accuracy than Memory-based CF techniques.
- Topic modeling [60–62] has received much research interest in recent years. In this type of techniques, each user u is represented by a multinomial distribution $\{P(z_k|u)\}_{k=1}^K$ over latent topics, each entry of which illustrates how much u is interested in a topic. Each topic z in turn is represented by a multinomial distribution $\{P(i_j|z)\}_{j=1}^{|I|}$ over items, illustrating how relevant the item i_j is to topic z . Then, the relevance between of an item i to a user u is computed as follows:

$$\hat{r}_{u,i} = \sum_{k=1}^K P(z_k|u)P(i|z_k). \quad (2.3)$$

2.1.2 Content-based Recommendation

Content-based recommendation techniques [63] utilize the user profiles or item description to determine the relevance between the user and the item. They first extract features from content of users and items by some information retrieval techniques. Next, content-based recommendation techniques learn the user preference from those features using machine learning algorithms, such as naive Bayes classifiers or decision trees. Finally, for each user, the system classifies candidate items into categories “likes” and “dislikes” based on learned user preference.

The advantage of content-based recommendation approaches is that they can overcome the cold-start problem, *i.e.*, they are able to make recommendation for new items that have just come to the system and do not have any rating from users. However, content-based recommendation techniques cannot recommend unexpected items to users. Moreover, they require enough information to train the classifier for each user, and their performance is limited by the features.

2.1.3 Hybrid Recommendation

Hybrid recommendation approaches combine different recommendation techniques so that they can exploit advantages and overcome limitations of individual techniques. Some hybrid recommendation techniques extend CF by incorporating content features [64,65]. Other techniques combine CF and content-based approaches, such as [66–68]. There are also hybrid models combining memory-based CF and model-based CF [69, 70]. Based on those work, hybrid recommendation approaches have reported their better performance over pure CF and content-based methods.

2.2 Context-aware Recommendation

As discussed in Section 1, we focus on three types of context information: heterogeneous networks, relations context and activity context. Correspondingly, we consider three types of context-aware recommendation systems: recommendation in heterogeneous networks, item-relation-aware recommendation and requirement-aware recommendation systems. In this section, we review existing work on each of these context-aware recommendation systems.

2.2.1 Recommendation Systems in Heterogeneous Networks

We review the existing work on heterogeneous networks based on the types of recommendation models. We also analyze the limitation of the existing methods.

Graph-based methods. This type of models represents users, items and other entities with their relations by a graph and then apply a graph mining technique. One of early methods is introduced by Konstas et al. [25], who apply random walk with restarts in a heterogeneous graph including users, music tracks and tags. On the other hand, Lee et al. [27] introduce new types of nodes that are combination of users and additional information, such as time and location, to enrich user-movie graph. Moreover, Wang et al. [71] design a user-location graph based on check-in data and user friendship, and then perform a color propagation algorithm on the graph to make recommendations. Lastly, Jiang et al. [29] focus on a special type of heterogeneous

social graphs, namely star-structured graphs, where user type is at the center and connected to other node types of the domain (such as web posts, videos); and their RW-based algorithm is proposed especially for star-structured graphs and cannot be applied to general heterogeneous graphs. Since the heterogeneous data and their relations are simply represented by a graph, graph-based methods are easily employed in various recommendation problems on various heterogeneous data. However, most of them adopt RWR as the recommendation algorithm. Since RWR considers all entities as the same type, and hence cannot model the influence strength between entity types. One exception from RWR-based methods is the method in [72] which is able to learn the influence weights between entity types; however, their method is not personalized, i.e. it estimates the global importance of every entity. As a result, in order to learn parameters, they need to annotate part of data as the groundtruth, which is very subjective and impracticable in large datasets. Finally, some work propose meta-path based methods on top of knowledge graphs for recommendation [24, 73]; however the disadvantage of those methods is that their performance are sensitive and heavily influenced by meta paths, which are required domain knowledge to construct.

Factorization models. They aim to learn user and item preferences by factorizing interaction matrices between users, items and other entities. Singh et al. [30] propose collective matrix factorization (MF) which factorizes multiple related matrices, including user-item and other matrices containing additional information, at the same time. Jamali et al. [19] extends collective MF to solve the imbalance problem between different domains. Subsequently, Agarwal et al. [74] introduce a regression-based latent factor model, which combines regression model and MF to integrate user and item attributes with user-item interactions for better preference prediction. Based on regression-based latent factor model, Zhang et al. [26] present a unified model to recommend groups to users in event-based social networks by exploiting social, spatial and textual information. Finally, Rendle et al. [31], [13] introduce Factorization Model that aims to learn the factorized interactions between every pair of entities in a heterogeneous domain. Similar to graph-based methods, models from this category are able to be utilized in many applications. However, because they often focus on

how to discover the interaction among entities, not entity types, it is difficult for those methods to understand the role of each entity type and explain the recommendation results. On the other hand, Luo et al. [75] take a different approach to exploiting heterogeneous data by constructing various meta-paths representing different semantic relations between users and items; then learning the importance weights for those paths. However, because those parameters are used as regularization terms but not directly used in the prediction function, they cannot identify the exact importance of different entities. Moreover, since there are no constraints for the parameters, they can be arbitrarily small or large, even negative, and therefore it is not easy to interpret them. Finally, they have to design some meta-paths, in which the accuracy of the model is highly dependent on.

Topic models. Approaches in this type extract user and item preferences as latent topics with the aid of extra information. Yuan et al. [32] incorporate users, tweets, time and location information into a probabilistic model, namely W4, to model users mobility behaviors. Kim et al. [33] present a model for recommending Digg articles by utilizing users, articles, scores and textual information. Zhao et al. [34] propose a topic model to capture the interaction of aspect, sentiment, category and spatial information, which can be used for POI recommendation. In contrast to two other types of recommendation models, topic-modeling methods are less flexible for general usage in different recommendation problems. This is because those models are designed based on assumptions of dependency between variables (represented by conditional probabilities), which are specific for a domain, and hence makes it difficult to reuse them in other domains.

2.2.2 Item-relation-aware Recommendation

In this section, we review some existing work on recommendation that consider the interaction between items. This type of recommendation systems can be divided into three subcategories: sequential recommendation, session-based recommendation and feature-based recommendation.

Sequential recommendation. Recommendation methods in this type aim to find next items that a user may choose given her most recent selected items. In one of early work, Zimdars et al. [76] introduce a sequential recommendation system based on Markov chains (MC). They use a common predictor, *e.g.*, decision tree, to learn the sequential patterns in user behaviors. Similarly, Mobasher et al. [77] apply pattern mining methods to discover sequential patterns of recommendation generating process. In [78], a recommendation system based on MC is introduced, where the maximum likelihood estimation is enhanced by using several heuristic approaches like skipping and clustering. The main limitation of those work is that they do not consider the users' individual preferences when extracting sequential patterns and predicting next selected items. To overcome this issue, Rendle et al. [79] introduce a factorized personalized MC model (FPMC) to recommend next basket of items for users. In particular, they construct a tensor of users, current items and next items, and then decompose it into three matrices, which represent users, current items and next items in a low-dimensional latent space. Based on FPMC model, Cheng et al. [80] develop a successive POI recommendation, which aims to recommend the next POI for users given their current location. Also in next POI recommendation, Feng et al. [81] apply the metric embedding method, which is firstly proposed for predicting next songs for users given their current playlists [82].

Session-based recommendation. This type of item-interaction-aware recommendation systems predicts the items based on other items in the same rating/purchasing session. McAuley et al. [22] consider two types of products, namely *substitutes* and *complements*. Substitutes are products that can be purchased instead of the other, while complements are products that can be purchased together. For example, it is more reasonable to recommend batteries, cases or charges to a user who has just bought a new phone, instead of recommending another phones. Sharing similar ideas, Zhu et al. [21] introduce a novel recommendation system, namely bundle recommendation, which recommends a package of items to users, in stead of individual items. In this work, items are selected such that they maximize the reward returned, where the reward is computed based on the user's interest on items and influence among

items in the package. On the other hand, Yang et al. [83] present a new type of recommendation methods, namely collaborative competitive filtering, which learns user and item preferences using context of user choice. In particular, users are given a set of items whenever accessing the system, and they choose one item that they like the most. The selection is the explicit feedback from users to express their interests, since we can consider that the selected items are preferred by users (*i.e.*, “positive” items) and unselected ones are not preferred by users (*i.e.*, “negative” items).

Feature-based recommendation. In this type of item-interaction-aware recommendation systems, items are recommended to users based on the relations between features of items. For example, based on the product categories, Sun et al. [84] classify a pair of products into *alternative* products (*e.g.*, Athletic Clothing and Fashion Clothing products) or *complementary* products (*e.g.*, Athletic Shoes and Heels), which shares the same idea with [22]. Similarly, Yang et al. [85] recommend locations to users based on the co-location relationship between recommended venues and venues visited by users.

2.2.3 Requirement-aware Recommendation

Requirement-aware recommendation is a relatively new research problem, and there is still very little existing work on it. To my best knowledge, there is only one research work from Yuan et al. [86] dealing with this problem. In this work, they aim to solve the requirement-aware POI recommendation problem, which recommends POIs to users given their requirements (represented by text). Two topic models, namely W^4 and EW^4 , are proposed, which models user mobility behavior from four factors: user, POI, time and text. Their experiments on Twitter show promising results over baseline methods, which proves the importance of considering user intention when making POI recommendation. In an earlier work, Weston et al. [87] introduce Latent Collaborative Retrieval model (LCR) to solve both retrieval and recommendation tasks simultaneously. Although it seems the model can be applied in requirement-aware recommendation problem if we consider queries as users’ intention, it requires queries

should be from a fixed set [88, 89], which is not applicable since user requirements can be arbitrary.

2.3 POI Recommendation

In this section, we review existing POI recommendation models. Models are grouped by their objective problems, namely conventional POI recommendation, next POI recommendation, time-aware POI recommendation and in-town/out-town POI recommendation.

2.3.1 Conventional POI Recommendation

Conventional POI recommendation is to *recommend new POIs (i.e., the user has never been to) that are likely to be visited by the user*. There are massive research work devoted for this problem. Here we further categorize POI recommendation models based on the information used in those models.

Geographical influence. In POI recommendation, because of the distance constraint, users may have certain geographical areas of activity where most of their visited POIs are located in. Therefore, several models attempt to capture geographical properties of users, to improve the recommendation. Some previous work use a one-dimensional distance distribution to model the geographical influence: Ye et al. [40] and Zhang et al. [90] apply a power-law distribution to model the probability of traveling a certain distance, and one-dimensional kernel density estimation (KDE) is introduced by Zhang et al. [91] to capture the distance distribution in a personalized manner. In addition, some methods directly model users active areas using two-dimensional distributions over latitude and longitude: Cheng et al. [92] assume that the distribution of users check-in POIs is multi-centered and propose a multi-center Gaussian model (MGM) to fit it; Zhang et al. [93] adopt a two-dimensional KDE to model the probability of visiting a POI, and the KDE model is further improved with adaptive bandwidth in their subsequent work [94]. Besides, Lian et al. [42] introduce activity area vectors and influence area vectors for augmenting Weighted

Matrix Factorization. Liu et al. [47] model geographical influence from a location perspective, via incorporating instance-level and region-level geographical neighborhood characteristics. Li et al. [43] consider the user preference not only on a candidate POI, but also on its neighborhood POIs. Hu et al. [95], Zhao et al. [34] and Liu et al. [96,97] build Bayesian models and integrate latent region in the generative process of the proposed probabilistic models.

Social influence. Social influence as commonly-used contextual information in conventional recommendation systems, is also extensively employed in POI recommendation. Ye et al. [40,98] propose a Friend-based Collaborative Filtering (CF) method to leverage social influence based on their common visited POIs. Zhang et al. [91,93,99] also take advantage of a similar CF method but with different definition of the similarity metric. In addition, Cheng et al. [92] place a social regularization term in Matrix Factorization framework, assuming that latent interests of friends tend to be similar. Li et al. [100] incorporate preference propagation on social network into Matrix Factorization. Noulas [101] et al. and Ying et al. [102] conduct random walk on social networks, which is taken as a part of user-POI graph in their model, for POI recommendation. Zhang et al. [94] aggregate social check-ins (check-ins made by users friends) as a power-law distribution to exploit social correlations between users.

Content information. In addition to geographical and social data, content information has been applied in building content-aware POI recommendation models in recent years. Hu et al. [103] incorporate topic distribution in their probabilistic model and optimize the parameters using text contents. Gao et al. [104] and Zhang et al. [99] learn sentiment indications from users tips, as the weights for user preference. Zhao et al. [34] capture the interactions of topics, aspects and sentiments using an unified probabilistic model. Lian et al. [105] propose a bag-of-word model and consider each word as a feature of user in recommendation. Bhargava et al. [106] leverage textual data in learning users activity and further build a tensor decomposition model. Besides textual content, other sorts of content such as categorical information have also been employed by some work [34,94,107].

2.3.2 Next POI Recommendation

Given a user and his/her current location, next POI recommendation aims at *recommending new POIs that are likely to be visited by the user in the next time interval (i.e., in the next 6 hours)*. Unlike conventional POI recommendation problem, in which most of the models are trying to integrate geographical and social influence, sequential influence is extensively studied and applied in next POI recommendation. Cheng et al. [80] embed personalized Markov Chain in modeling users transitions among POIs. Feng et al. [81] take transition probability between POIs as the Euclidean distance between their latent representation in a low-dimensional space, and propose a personalized ranking metric embedding method. Zhao et al. [108] and He et al. [109] both propose a ranking-based tensor factorization model for next POI recommendation.

2.3.3 Time-aware POI Recommendation

Considering that user preference varies with time, *i.e.*, a user will more likely visit a restaurant during lunch time and visit a bar at midnight, time-aware POI recommendation returns new POIs that are most likely to be visited at the specific time (*i.e.*, 5 p.m.) to users. Time-aware POI recommendation is first proposed by Yuan et al. [110], and a CF model with time-dependent similarity between users is introduced to solve the problem. Subsequently, Yuan et al. [111] propose a Geographical-Temporal influences Aware Graph method to further improve the recommendation results. Li et al. [43] augment their conventional POI recommendation model by including time-related latent factors, and apply tensor decomposition for time-aware recommendation. Deveaud et al. [112] devise a venue-centric method that models time-aware properties of POIs.

2.3.4 In-town/Out-town POI Recommendation

POI recommendation is even more important for users who are traveling out of his/her hometown city (*i.e.* out-of-town users) to satisfy their exploration purpose. However, previous work [113] shows that the performance of conventional POI recommendation

models on out-of-town users is worse than in-town users in a large margin. Therefore, recent work distinguish conventional POI recommendation into in-town POI recommendation and out-of-town recommendation, specializing their model or conducting experiments for both in-town and out-of-town scenarios. Ference et al. [113] fuse user preference, geographical and social information into a CF method, using different parameter settings to perform POI recommendation for both in-town and out-of-town users. Kurashima et al. [114] propose a LDA-based model and conduct experiments for both daily life (in-town) and traveling (out-of-town) scenarios. Yin et al. [41, 115] consider not only user interest, but also local preference and POI content, to overcome the cold-start problem for out-of-town users. Wang et al. [45] design a novel geographical sparse additive generative model and apply different latent topic distributions for native (in-town users) and tourist (out-of-town users).

Chapter 3

A General Recommendation Model for Heterogeneous Networks

In this chapter, we introduce our *general-purpose Heterogeneous graph-based Recommendation System* model (HeteRS) to solve general tasks of recommendation in heterogeneous networks. We first define a heterogeneous graph built from a heterogeneous social network. Then, we introduce how to accomplish multiple recommendations tasks based on the constructed graph. Next, since our model may encounter the efficiency problem, we design a fast learning method and an approximation algorithm to achieve better efficiency in the offline (learning parameters) and online (making recommendation) process, respectively. Finally, we propose an extension of our model which learns different influence weights for groups of users based on their behavior similarities to further improve the recommendation accuracy.

3.1 Proposed Recommendation Model on Heterogeneous Networks

3.1.1 Heterogeneous Network Graph

Definition 3.1 (Heterogeneous Network Graph) Let $\mathbb{T} = \{\mathbb{t}_1, \mathbb{t}_2, \dots, \mathbb{t}_{|\mathbb{T}|}\}$ be the set of entity types, $\mathbb{E}^{\mathbb{t}_i} = \{e_1^{\mathbb{t}_i}, e_2^{\mathbb{t}_i}, \dots, e_{|\mathbb{E}^{\mathbb{t}_i}|}^{\mathbb{t}_i}\}$ be the set of entities of type $\mathbb{t}_i$, and $\mathcal{R} = \{\langle \mathbb{t}_i, \mathbb{t}_j \rangle | \mathbb{t}_i, \mathbb{t}_j \in \mathbb{T}\}$ be the set of relation types¹. The **Heterogeneous Network Graph** is defined as a directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node

¹The relation between two types is two-way, i.e., $\mathcal{R}$ includes both $\langle \mathbb{t}_i, \mathbb{t}_j \rangle$ and $\langle \mathbb{t}_j, \mathbb{t}_i \rangle$. Later, we only list one of two relations for simplicity.

Figure 3.1: a) Event-based Social Network Graph with nodes of users (U), events (E), groups (G), tags (T) and venues (V); b) Extension of EBSN Graph with session nodes (S).

set $\mathcal{V} = \bigcup_{t \in \mathbb{T}} \mathbb{E}^t$ and the edge set $\mathcal{E} = \{\langle m, n \rangle | m \in \mathbb{E}^M \wedge n \in \mathbb{E}^N \wedge \langle M, N \rangle \in \mathcal{R} \wedge m \text{ has a relation with } n\}$. Let $\mathbf{A}_{NM} \in \mathcal{R}^{|\mathbb{E}^N| \times |\mathbb{E}^M|}$, ($\langle M, N \rangle \in \mathcal{R}$), denote the adjacency matrix representing the relations (interactions) from nodes of type M to the ones of type N . Then $\mathbf{A}_{NM}(n, m) = 1$, if $m \in \mathbb{E}^M \wedge n \in \mathbb{E}^N \wedge \langle m, n \rangle \in \mathcal{E}$, and 0 otherwise².

Note that adjacency matrices $\mathbf{A}_{NM}$ (and $\mathbf{A}_{MN}$) are defined only for pairs of types M and N that have a relation, i.e., $\langle M, N \rangle \in \mathcal{R}$.

Definition 3.1 is the general definition for heterogeneous network (HN) graphs. Based on def. 3.1 we can instantiate an HN graph for a particular heterogeneous network. For example, the Event-based Social Network Graph can be instantiated as follows:

Definition 3.2 (Event-based Social Network Graph) *The **Event-based Social Network Graph** is an HN graph with $\mathbb{T} = \{U, E, G, T, V\}$, which consists of entity types of users, events, groups, tags and venues, respectively; and $\mathcal{R} = \{\langle U, E \rangle, \langle E, G \rangle, \langle E, V \rangle, \langle U, G \rangle, \langle U, T \rangle, \langle G, T \rangle\}$.*

Figure 3.1a illustrates the structure of the EBSN graph. The EBSN graph consists of the explicit information extracted from the event-based social network data (which are shown in the example in Fig. 1.1); however, after analyzing the dataset, we found

²Here, $\mathbf{A}_{NM}(n, m)$ means the entry at n -th row and m -th column of $\mathbf{A}_{NM}$.

that user behaviors follow some hidden temporal patterns. Particularly, we observe that users usually take part in events with a weekly periodic schedule. Moreover, users tend to join two events of their interest in a short time period. To capture these implicit patterns of user behaviors in event-based social networks, we extend the graph by adding a new type of nodes, namely session nodes, and changing the edge weights based on event time. The extended EBSN graph with session nodes are shown in Fig. 3.1b. Details of these modifications can be found in [116].

3.1.2 Recommendation on Heterogeneous Network Graphs

We next describe how to perform a recommendation task on our graph. Our idea is to transform the recommendation problem into node proximity calculation problems w.r.t. some query nodes, and then use multivariate Markov chain to solve it. We choose the task of recommending groups to users in an EBSN as the example to introduce our method. For recommendation problems in other HN graphs, the method is applied in the similar way.

Definition 3.3 (Group-to-user recommendation) *Given a user u , **group-to-user recommendation** is the problem to find groups that u is likely to join.*

Now we describe how to use the EBSN graph to recommend groups for a user. As our method is based on MMC, we first define the transition matrix.

Definition 3.4 (Transition matrix) *Transition matrix $\mathbf{P}_{NM}$ ($\langle M, N \rangle \in \mathcal{R}$) is obtained by normalizing adjacency matrices $\mathbf{A}_{NM}$ by columns. $\mathbf{P}_{NM}$ handles dangling nodes, i.e., if there exists an entity m such that $\sum_n \mathbf{A}_{NM}(n, m) = 0$, then $\mathbf{P}_{NM}(n, m) = 1/|N|, \forall n$.*

For the EBSN graph, there are multiple transition matrices, each of which corresponds to a relation from one to another type of entities. Next, we consider how to construct query vector given a user.

Definition 3.5 (User query vector) *Given a user u , we define the **user query vector** as $\mathbf{q}_u \in \mathbb{R}^{|U|}$, where $\mathbf{q}_u(i) = 1$ if $i = u$, 0 otherwise.*

With the transition matrices and user query vector, we are able to simulate a random walk process by using MMC on the heterogeneous graph to calculate the proximity of nodes. We establish the following equations:

$$\mathbf{u}^{(t+1)} = \alpha_{UE}\mathbf{P}_{UE}\mathbf{e}^{(t)} + \alpha_{UG}\mathbf{P}_{UG}\mathbf{g}^{(t)} \quad (3.1)$$

$$+ \alpha_{UT}\mathbf{P}_{UT}\mathbf{t}^{(t)} + (1 - \alpha_{UE} - \alpha_{UG} - \alpha_{UT})\mathbf{q}_u$$

$$\mathbf{e}^{(t+1)} = \alpha_{EU}\mathbf{P}_{EU}\mathbf{u}^{(t)} + \alpha_{EG}\mathbf{P}_{EG}\mathbf{g}^{(t)} \quad (3.2)$$

$$+ \alpha_{EV}\mathbf{P}_{EV}\mathbf{v}^{(t)} + (1 - \alpha_{EU} - \alpha_{EG} - \alpha_{EV})\mathbf{P}_{ES}\mathbf{s}^{(t)}$$

$$\mathbf{g}^{(t+1)} = \alpha_{GE}\mathbf{P}_{GE}\mathbf{e}^{(t)} + \alpha_{GU}\mathbf{P}_{GU}\mathbf{u}^{(t)} \quad (3.3)$$

$$+ (1 - \alpha_{GE} - \alpha_{GU})\mathbf{P}_{GT}\mathbf{t}^{(t)}$$

$$\mathbf{t}^{(t+1)} = \alpha_{TU}\mathbf{P}_{TU}\mathbf{u}^{(t)} + (1 - \alpha_{TU})\mathbf{P}_{TG}\mathbf{g}^{(t)} \quad (3.4)$$

$$\mathbf{s}^{(t+1)} = \mathbf{P}_{SE}\mathbf{e}^{(t)} \quad (3.5)$$

$$\mathbf{v}^{(t+1)} = \mathbf{P}_{VE}\mathbf{e}^{(t)} \quad (3.6)$$

where

- $\mathbf{u}^{(t)}, \mathbf{e}^{(t)}, \mathbf{g}^{(t)}, \mathbf{t}^{(t)}, \mathbf{s}^{(t)}, \mathbf{v}^{(t)}$ are distribution probability vectors representing the probabilities that users, events, groups, sessions and venues are visited at time t , respectively.
- α_{NM} ($\langle M, N \rangle \in \mathcal{R}$, $\alpha_{NM} > 0$ and $\sum_M \alpha_{NM} \leq 1$) denotes the transition weight (or influence weight) from nodes of type M to nodes of type N .

Equations (3.1)-(3.6) model how the probabilities change for different types of nodes after each step of random walk transition on our EBSN graph. We can see from the equations that α_{NM} explicitly controls how much probability (influence) one type of nodes receive from the other types of nodes. For example, in Eq. (3.1), user nodes receive α_{UE} probability from event nodes, and α_{UG} probability from group nodes, α_{UT} probability from tag nodes and $(1 - \alpha_{UE} - \alpha_{UG} - \alpha_{UT})$ from query user node. The explicit controllability benefits from the fact that $\mathbf{P}_{UE}$, $\mathbf{P}_{UG}$ and $\mathbf{P}_{UT}$ are transition matrices and $\mathbf{e}^{(t)}$, $\mathbf{g}^{(t)}$ and $\mathbf{t}^{(t)}$ are probability distribution vectors, and as a result

their products are still probability distribution vectors. This explicit controllability is not only useful to explain the model, but important to reveal the roles of different entities in different recommendation tasks after learning α_{NM} . However, if we project the graph into homogeneous graph and apply traditional univariate random walk, we cannot obtain the explicit controllability.

After solving the stationary probability vectors from Eqs. (3.1)-(3.6), the recommendations can be made by ranking the groups based on $\mathbf{g}$. We note that, without considering the query inputs, the model degenerates into the original MMC in [117]. The query vectors are introduced into the model to produce “personalized” results, the idea of which is similar to RWR. When considering tag-to-group recommendation, we incorporate the group query vector $\mathbf{q}_g$ into Eq. (3.3) and remove the user query vector $\mathbf{q}_u$ from Eq. (3.1). For event-to-user recommendation, we can perform similar modifications to incorporate query vectors $\mathbf{q}_u, \mathbf{q}_g, \mathbf{q}_v, \mathbf{q}_s$ into Eqs. (3.1)-(3.6).

Theorem 3.1 *If the EBSN graph is connected, probability vectors in Eqs. (3.1)-(3.6) uniquely converge to stationary vectors as t goes to infinity.*

Proof: First, we rewrite Eqs. (3.1)-(3.6) as follows:

$$\mathbf{r}^{(t+1)} = \mathbf{M}\mathbf{r}^{(t)} + \mathbf{q} \quad (3.7)$$

where:

$$\mathbf{r} = (\mathbf{u}^\top, \mathbf{e}^\top, \mathbf{g}^\top, \mathbf{t}^\top, \mathbf{s}^\top, \mathbf{v}^\top)^\top \quad (3.8)$$

$$\mathbf{M} = \begin{pmatrix} \mathbf{0} & \alpha_{EU}\mathbf{P}_{EU} & \cdots & \mathbf{0} \\ \alpha_{UE}\mathbf{P}_{UE} & \mathbf{0} & \cdots & \alpha_{VE}\mathbf{P}_{VE} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{P}_{EV} & \cdots & \mathbf{0} \end{pmatrix} \quad (3.9)$$

$$\mathbf{q} = (\alpha_{UU}\mathbf{q}_u^\top, \mathbf{0}^\top, \mathbf{0}^\top, \mathbf{0}^\top, \mathbf{0}^\top, \mathbf{0}^\top)^\top \quad (3.10)$$

$$\alpha_{UU} = 1 - \alpha_{EU} - \alpha_{GU} - \alpha_{TU}$$

$$\alpha_{SE} = 1 - \alpha_{UE} - \alpha_{GE} - \alpha_{VE}$$

$$\alpha_{TG} = 1 - \alpha_{EG} - \alpha_{UG}$$

$$\alpha_{GT} = 1 - \alpha_{UT}$$

Note that $\mathbf{M}$ is not a stochastic matrix. Let Λ be the matrix containing parameters from $\mathbf{M}$:

$$\Lambda = \begin{pmatrix} 0 & \alpha_{EU} & \cdots & 0 \\ \alpha_{UE} & 0 & \cdots & \alpha_{VE} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 \end{pmatrix}$$

We can see that Λ is a sub-stochastic matrix, and thus the *spectral radius* (maximum of the absolute eigenvalues) of Λ is strictly smaller than 1, denoted by $\rho(\Lambda) = |\lambda| < 1$. We can also know $\lambda > 0$ because Λ is a nonnegative and irreducible matrix (there is always a path from one type to another type of entities, so Λ is irreducible). Then by Perron-Frobenius Theorem, there exists a positive vector $\mathbf{z} = (z_U, z_E, z_G, z_T, z_S, z_V)^\top$ such that $\mathbf{z}^\top \Lambda = \lambda \mathbf{z}^\top$. We note that $\mathbf{1}_{|N|}^\top \mathbf{P}_{MN} = \mathbf{1}_{|M|}^\top$ where $\mathbf{1}_{|M|}$ is the vector with size $|M|$ of all ones. Then it is easy to show that $(z_U \mathbf{1}_{|U|}^\top, z_E \mathbf{1}_{|E|}^\top, \dots, z_V \mathbf{1}_{|V|}^\top) \mathbf{M} = \lambda (z_U \mathbf{1}_{|U|}^\top, z_E \mathbf{1}_{|E|}^\top, \dots, z_V \mathbf{1}_{|V|}^\top)$, hence λ is an eigenvalue of $\mathbf{M}$ and its corresponding eigenvector is positive. Since the EBSN graph is connected, $\mathbf{M}$ is a non-negative and irreducible matrix. Based on Perron-Frobenius Theorem, we know that only eigenvectors associated with the spectral radius of a nonnegative and irreducible matrix are positive, and thus we have $\rho(\mathbf{M}) = \lambda$.

Suppose $\mathbf{r}^{(0)} = \pi$, we have $\mathbf{r}^{(1)} = \mathbf{M}\pi + \mathbf{q}$, $\mathbf{r}^{(2)} = \mathbf{M}^2\pi + \mathbf{M}\mathbf{q} + \mathbf{q}$, $\dots$, $\mathbf{r}^{(t)} = \mathbf{M}^t\pi + \sum_{k=0}^{t-1} \mathbf{M}^k \mathbf{q}$. Since $\rho(M) = \lambda < 1$, we have $\lim_{t \rightarrow \infty} \mathbf{M}^t = \mathbf{0}$ and $\lim_{t \rightarrow \infty} \sum_{k=0}^{t-1} \mathbf{M}^k = (\mathbf{I} - \mathbf{M})^{-1}$. So $\mathbf{r}^{(t)}$ finally converges to $\mathbf{r}^* = (\mathbf{I} - \mathbf{M})^{-1} \mathbf{q}$.

Generally, our constructed graph may not be connected. However, there usually exist dangling nodes in our graph, for example, considering the user-group interaction, if a user does not join any group, she is then a dangling node. According to definition 3.4, we consider under this circumstance that she is connected to all group nodes. This operator for dangling nodes usually makes our EBSN graph be connected. Therefore we can obtain the stationary probability vectors by iteratively running Eqs. (3.1)-(3.6) until it converges.

3.2 Optimization Approach for Parameter Learning

From Eqs. (3.1)-(3.6), we can see the transition weights α_{NM} play an important role in producing the final solution, as they control how much probability one type of entities receives from others. In this section, we consider to learn them from training data. To this end, we first design an objective function on α_{NM} , and then propose a learning algorithm to optimize it. Again, we use group-to-user recommendation task as the example to illustrate our learning method; for other recommendation problems the learning process is similar.

3.2.1 Objective Function

We follow the Bayesian Personalized Ranking (BPR) optimization framework [118] to construct our objective function. In group-to-user recommendation, given a user, we consider groups which she already joined as positive groups, denoted by the set PG , while the ones that the user did not join as negative groups, denoted by NG . In other words, the whole group set consists of two parts: $G = PG \cup NG$. Then the appropriate parameters α_{NM} of Eqs. (3.1)-(3.6) should rank all the positive groups higher than negative groups, *i.e.*, positive groups should have higher probability than negative ones. To model this, we design the following AUC (Area Under the ROC Curve) objective to be maximized, where we assume there are m instances $\{< u_k, PG_k >\}_{k=1}^m$:

$$\max_{\alpha} Obj(\alpha) = \sum_{k=1}^m \frac{\sum_{i \in PG_k} \sum_{j \in NG_k} \mathbb{1}(\mathbf{g}(i) - \mathbf{g}(j))}{|PG_k| |NG_k|}, \quad (3.11)$$

where $NG_k = G - PG_k$ and $\mathbb{1}(\cdot)$ is an indicator function that equals to 1 if $\mathbf{g}(i) > \mathbf{g}(j)$, and 0 otherwise. As the indicator function $\mathbb{1}(\cdot)$ is not differentiable, it is usually approximated by sigmoid function:

$$\sigma(x; \beta) = \frac{1}{1 + e^{-\beta x}}, \quad (3.12)$$

where the parameter β controls the approximate error, and is set empirically during the training process. Substituting it into Eq. (3.11) and considering in log form as in [26], we obtain a new objective function as follows:

$$\max_{\alpha} Obj(\alpha) = \sum_{k=1}^m \frac{\sum_{i \in PG_k} \sum_{j \in NG_k} \ln \sigma(\mathbf{g}(i) - \mathbf{g}(j))}{|PG_k| |NG_k|}. \quad (3.13)$$

3.2.2 Solving the Optimization Problem

To find parameters α that maximize the objective function in Eq. (3.13), we adopt stochastic gradient descent (SGD) algorithm. In SGD, the parameters are step by step updated based on one training instance, instead of all training data as in gradient descent. Therefore, SGD is more suitable when the training data is large. Specifically, for each training instance, its derivative is calculated and the parameters α are updated by moving along the ascending direction of the gradient as follows:

$$\alpha \leftarrow \alpha + lr \frac{\partial Obj_k(\alpha)}{\partial \alpha}, \quad (3.14)$$

where $Obj_k(\alpha)$ is the objective function for k -th training instance:

$$Obj_k(\alpha) = \frac{\sum_{i \in PG_k} \sum_{j \in NG_k} \ln \sigma(\mathbf{g}(i) - \mathbf{g}(j))}{|PG_k| |NG_k|}. \quad (3.15)$$

Here $\mathbf{g}(i)$ denotes the ranking score of the i -th group. Accordingly, we have the partial derivatives of $Obj_k(\cdot)$ w.r.t. α as:

$$\frac{\partial Obj_k(\alpha)}{\partial \alpha} = \frac{\sum_{i \in PG_k} \sum_{j \in NG_k} \frac{\partial \ln \sigma(\delta_{ij})}{\partial \delta_{ij}} (\frac{\partial \mathbf{g}(i)}{\partial \alpha} - \frac{\partial \mathbf{g}(j)}{\partial \alpha})}{|PG_k| |NG_k|}, \quad (3.16)$$

where $\delta_{ij} = \mathbf{g}(i) - \mathbf{g}(j)$. From Eq. (3.12), we easily have $\partial \ln \sigma(\delta_{ij}) / \partial \delta_{ij} = \beta(1 - \sigma(\delta_{ij}))$. In the implementation, we set $\partial \ln \sigma(\delta_{ij}) / \partial \delta_{ij} = (1 - \sigma(\delta_{ij}))$ because β can be integrated into learning rate lr .

The remaining issue is how to compute the derivative $\partial \mathbf{g}(i) / \partial \alpha$. Taking the

derivatives w.r.t. each parameter α_{NM} on both sides of Eq. (3.3), we can get:

$$\frac{\partial \mathbf{g}}{\partial \alpha_{GE}} = \mathbf{P}_{GE} \mathbf{e} - \mathbf{P}_{GT} \mathbf{t} \quad (3.17)$$

$$\begin{aligned} \frac{\partial \mathbf{g}}{\partial \alpha_{EU}} &= \alpha_{GE} \mathbf{P}_{GE} \frac{\partial \mathbf{e}}{\partial \alpha_{EU}} \\ &= \alpha_{GE} \mathbf{P}_{GE} (\mathbf{P}_{EU} \mathbf{u} - \mathbf{P}_{ES} \mathbf{s}) \\ &\propto \mathbf{P}_{GE} (\mathbf{P}_{EU} \mathbf{u} - \mathbf{P}_{ES} \mathbf{s}) \end{aligned} \quad (3.18)$$

$$\begin{aligned} \frac{\partial \mathbf{g}}{\partial \alpha_{UE}} &= \alpha_{GU} \mathbf{P}_{GU} \frac{\partial \mathbf{u}}{\partial \alpha_{UE}} \\ &\propto \mathbf{P}_{GU} (\mathbf{P}_{UE} \mathbf{e} - \mathbf{q}_u) \end{aligned} \quad (3.19)$$

$$\begin{aligned} \frac{\partial \mathbf{g}}{\partial \alpha_{TU}} &= \alpha_{GT} \mathbf{P}_{GT} \frac{\partial \mathbf{t}}{\partial \alpha_{TU}} \\ &\propto \mathbf{P}_{GT} (\mathbf{P}_{TU} \mathbf{u} - \mathbf{P}_{TG} \mathbf{g}) \end{aligned} \quad (3.20)$$

Here, we only show derivatives $\partial \mathbf{g} / \partial \boldsymbol{\alpha}$ w.r.t. some parameters α_{NM} ; the remaining ones can be derived in the similar way. Note that, by definition, parameters $\boldsymbol{\alpha}$ must be positive, so we set a lower bound γ for all transition parameters $\boldsymbol{\alpha}$. Whenever a parameter α_{NM} becomes smaller than γ , which is 0.01 in our implementation, its value is set to γ . Moreover, when the sum of all transition parameters α_{NM} to type N is not 1, we normalize those parameters so that their sum equals to 1.

Based on the gradients derived above, we can learn the optimal parameters $\boldsymbol{\alpha}$, and the learning scheme is summarized in Algorithm 1.

Algorithm 1: Learning process

Input: m training instances, and learning rate lr

Output: optimal $\boldsymbol{\alpha}$

begin

```

1   t = 0 ;
2   Initialize  $\boldsymbol{\alpha}^{(0)}$ ;
3   while  $Obj(\boldsymbol{\alpha})$  has not converged do
4       Randomly shuffle the  $m$  training instances;
5       foreach training instance  $k$  do
6           Compute stationary vectors  $\mathbf{u}, \mathbf{e}, \mathbf{g}, \mathbf{t}, \mathbf{s}, \mathbf{v}$  by iteratively executing
              Eqs. (3.1)-(3.6);
7           Compute  $\partial \mathbf{g} / \partial \boldsymbol{\alpha}$  based on Eqs. (3.17)-(3.20);
8           Update  $\boldsymbol{\alpha} : \boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + lr \frac{\partial Obj_k(\boldsymbol{\alpha}^{(t)})}{\partial \boldsymbol{\alpha}}$ ;
9       t = t+1;
    
```

3.2.3 Fast Learning Method

We can see from Algorithm 1 that most of the overhead of the learning process comes from computing stationary vectors $\mathbf{u}, \mathbf{e}, \mathbf{g}, \mathbf{t}, \mathbf{s}, \mathbf{v}$ based on MMC for every training instance (line 6). The time complexity of one MMC computation is $O(tn)$, where n is number of edges (represented by nonzero values in transition matrices) and t is number of iterations for executing Eqs. (3.1)-(3.6) until probability vectors converge. Obviously, when the graph becomes large, the computation is expensive. In this section, we propose a fast learning method to save the learning time by reducing the number of iterations for computing MMC.

Our idea comes from the fact that the learning process comprises of multiple training epochs (shown as lines 3-5 in Algorithm 1) and each of training instances is visited in each epoch. Since the parameter α has changed between two successive epochs, this leads to the change of MMC Eqs. (3.1)-(3.6). However, we can assume that this change is so small that if we use the stationary vectors computed from the previous epoch as initial vectors for new MMC equations of the same training instance, the computation stops quickly, *i.e.*, after just few iterations. In other words, we provide a “better” set of initial vectors, which are “closer” to final result vectors, than random initial vectors to execute MMC; and hence reduce the computation time. Overall, *our solution is to store and update initial vectors for each of the training instance every time we encounter them during the learning process.*

However, storing the initial vectors of all training instances is space-consuming, where the space complexity is $O(N|\mathcal{V}|)$, in which N is number of training instances and $|\mathcal{V}|$ is number of entities (or nodes in the graph). To overcome this issue, we propose a storing technique to maintain the set of initial vectors, which is illustrated in Fig. 3.2. Our idea is that, for each initial vector, we do not need store the whole vector but only keep entries with values greater than a threshold ξ and discard the remaining part (Fig. 3.2b). In other words, we maintain each vector in a compressed form, in which only large entries are kept. To reconstruct the whole vector for the next MMC computation, the missing elements are generated uniformly (Fig. 3.2c).

Figure 3.2: An example of the efficient storage technique with $\xi = 0.1$: a) Original probability vector; b) The vector after removing small entries; c) The vector after reconstruction.

The intuition behind this idea is that probability vectors are normally very skew, *i.e.*, very few entries have large values while most of them have small values, so keeping only large entries saves the storage space significantly. Moreover, the changes on small entries make the reconstructed vectors just slightly different from the original ones. Therefore, we can discard small entries and reconstruct them uniformly without sacrificing much the efficiency of our fast learning strategy.

3.2.4 Approximation Algorithm

In this section, we present a solution to accelerate our proposed model for the online recommendation phase. Although an efficient strategy is proposed in Section 3.2.3 to execute MMC in the learning phase, we cannot use it in the recommendation step, because of two reasons: (i) In some problems, such as event-to-user recommendation, query objects (*e.g.*, events) are always new, *i.e.*, have never been seen in the training phase, and hence we do not have initial vectors of these instances. (ii) Saving initial vectors for all objects, even in compressed form, takes space. Therefore, we propose an approximation algorithm for HeteRS, which could yield much better efficiency than the original HeteRS without sacrificing much accuracy. Note that the approximation algorithm still does not need any additional storage.

In Theorem 3.1, we prove that $\mathbf{r}^{(t)} = \mathbf{M}^t \pi + \sum_{k=0}^{t-1} \mathbf{M}^k \mathbf{q} = \sum_{k=0}^{t-1} \mathbf{M}^k \mathbf{q}$. In other words,

$$\mathbf{r} = (\mathbf{I} + \mathbf{M} + \mathbf{M}^2 + \mathbf{M}^3 + \dots) \mathbf{q} \quad (3.21)$$

Equation (3.21) gives us an idea to design an approximation algorithm. Since $\mathbf{p}^{(k)} = \mathbf{M}^k \mathbf{q}$ represents the preferences of all nodes in the graph after k steps of preference propagation starting from query nodes, and $\mathbf{p}^{(k)}$ can be computed directly from $\mathbf{p}^{(k-1)}$ ($\mathbf{p}^{(k)} = \mathbf{M} \mathbf{p}^{(k-1)}$), we may simulate the process of propagating preferences over our constructed graph to estimate $\mathbf{r}$, which is shown in Algorithm 2.

Particularly, we maintain a queue for nodes to be visited, and it initially contains query nodes (line 4). Then, each time a node i is taken from the queue, we send to each of i 's neighbors j an amount of preference $\mathbf{M}(j, i) \times \mathbf{p}(i)$, where $\mathbf{p}(i)$ is current preference of i (line 10-11), and put j into the queue if it is not in (line 13-14). Each time a node j receives preference from other node, we update the accumulating preference vector $\mathbf{r}$ in the corresponding position so that $\mathbf{r}(j)$ contains the total amount of preference that j receives during the process (line 12). When the preference of a node is below a threshold ϵ , we ignore this node and do not send any value to its neighbors (line 8). The algorithm is terminated when there is no any node to be visited, *i.e.*, the queue is empty, and $\mathbf{r}$ is returned as the result.

Algorithm 2: Approximation algorithm

Input: Matrix $\mathbf{M}$ and query nodes $\{q_i\}$
Output: Score vector $\mathbf{r}$
begin
 1 Initialize $\mathbf{q}$ for query nodes $\{q_i\}$ as in Eq. (3.10);
 2 $\mathbf{r} = \mathbf{q}$; $\backslash\backslash$ accumulating preference vector;
 3 $\mathbf{p} = \mathbf{q}$; $\backslash\backslash$ current preference vector;
 4 $\mathbf{Q} := \{q_i\}$; $\backslash\backslash$ queue of nodes to be visited;
 5 **while** $\mathbf{Q}$ is not empty **do**
 6 $i = \mathbf{Q}.\text{pop}()$;
 7 **if** $\mathbf{p}(i) < \epsilon$ **then**
 8 $\backslash\backslash$ continue;
 9 **foreach** j of i 's neighbors **do**
 10 $w = \mathbf{M}(j, i) \times \mathbf{p}(i)$;
 11 $\mathbf{p}(j) = \mathbf{p}(j) + w$;
 12 $\mathbf{r}(j) = \mathbf{r}(j) + w$;
 13 **if** $j \notin \mathbf{Q}$ **then**
 14 $\mathbf{Q}.\text{push}(j)$;
 15 $\mathbf{p}(i) = 0$;
 16 **return** $\mathbf{r}$;

Compared to the iterative algorithm, the approximation one is more efficient because iterative method takes into account all nodes, most of which are irrelevant nodes, while approximation algorithm exploits local propagation, and it rarely goes too far from query nodes. Through our experiments, it is shown that the approximation algorithm is significantly faster than iterative method while achieving similar accuracy.

Note that the approximation algorithm cannot be used in the learning process to quickly execute MMC (line 6 in Algorithm 1). This is because its output vectors $\mathbf{u}, \mathbf{e}, \mathbf{g}, \mathbf{t}, \mathbf{s}, \mathbf{v}$, although preserve the ranking between entities, are much different from exact vectors in values. This leads to derivatives in Eqs. (3.17)-(3.20), and thus the gradient in Eq. (3.14), are computed incorrectly. As a result, parameters α are updated by moving along wrong directions, and hence never reach the optimal point.

3.3 Enhanced HeteRS

In previous sections, we presented HeteRS as a general recommendation model for heterogeneous networks. We introduce a learning scheme to learn the parameters for our model in different problems, which is based on feedback of all the users, and use the learned model to generate recommendation results for all the users.

However, users may have different behavior patterns that cannot be adequately captured by a single model. For example, when deciding movies to watch, some users may choose movies based on their genres, while other users select movies based on their interesting directors or actors/actresses. Obviously, our model with one global parameter set is not able to capture different behavior patterns at the same time.

To overcome this issue, we extend our model to model user behaviors in a finer granularity level. Instead of learning one recommendation model for all the users, we plan to build different recommendation models for different users. A straightforward way is that, for each user, we learn a distinct parameter set, as described in Section 3.2, based on only her own feedback data. However, users usually have very little data, which makes it impossible to build a proper model for each of them.

Nevertheless, we can resolve this difficulty by dividing users into multiple groups, and build a model for each group. This is because although users may have different behaviors, some of them have similar preferences or interests. For example, some users love action movies and always watch movies from this genre regardless their directors or actors; and users who are fans of a popular actress always follow her movies. As a result, those users will have similar parameter sets as they share similar behaviors, and therefore we can learn one model for each set of similar users. This not only solves the problem of lack of training data for each model, but also reduces the complexity as the number of models is much smaller than the number of users.

The next question is how to cluster users into groups. As we want to group users with similar interests, it is natural to cluster users based on recommended items. For example, in movie-to-user recommendation, we group users based on their movies. We use the traditional hierarchical clustering method to cluster users, and similarity between two users is computed by using Jaccard metric $sim(u_1, u_2) = \frac{|M(u_1) \cap M(u_2)|}{|M(u_1) \cup M(u_2)|}$, where $M(u_i)$ is the set of movies of user u_i . For group-to-user recommendation, the user similarity estimation is difficult since the user-group data is too sparse and there are many users without group information. The solution for these problems will be discussed in Section 3.4.13.

The number of clusters is important to the performance of Enhanced HeteRS and needs to be carefully decided. If the number of clusters is too small, it cannot distinguish much between different users, hence the improvement may not be significant. On the other hand, when there are too many clusters, the number of users in each cluster is small, and therefore there will not be enough training data to learn parameters for each model. We will examine the performance of Enhanced HeteRS with different number of clusters in Section 3.4.13 .

After obtaining recommendation models for all user clusters, given a user, recommendation scores of all items are generated by the model corresponding to the cluster of the user.

3.4 Experiments

3.4.1 Dataset Description

In our experiments, we use three real-world datasets. The first one is the Meetup dataset that is used for group-to-user, tag-to-group and event-to-user recommendation on EBSN in our previous work [116]. Moreover, to demonstrate our model’s ability in other heterogeneous networks, we use two more datasets, namely MovieLens + IMDb/Rotten Tomatoes and Yahoo! Music. With the two additional datasets, many recommendation problems can be introduced for evaluation. We choose movie-to-user and track-to-user recommendation on the two datasets, respectively. Other problems (for example, album-to-user recommendation) are not much different, and our model can be easily applied to solve them without much modification.

3.4.1.1 Meetup Dataset

We crawl meetup.com to construct two datasets for New York City (NYC) and state of California (CA) respectively during the period from Jan. 1st to Dec. 31st 2012. For preprocessing, we keep users who joined at least 5 events, events with at least 5 participants, and groups with more than 20 events. The dataset statistics for both regions are summarized in Table 3.1.

Subsequently, we randomly select 20% groups of each user and 20% tags of each group as test sets for the tasks of group-to-user and tag-to-group recommendation, respectively. The remaining data will be used to form our EBSN graph. To learn the parameters for HeteRS in group-to-user recommendation problem, we further select 10% groups from each user as tuning set and remove corresponding user-group edges from the EBSN graph. The tuning set for tag-to-group recommendation is built in the same way. For event-to-user recommendation, the construction is different because events are time related. In particular, we remove events in the last 3 months of training set, *i.e.*, Oct. 1st - Dec. 31st, to build the tuning set. For the test set, we extract from the data events held after training set’s last timestamp (Dec 31st 2012) and joined by at least 5 users in the training set. We use events in first month (1st-month test data)

Table 3.1: Meetup dataset

Dataset	NYC	CA
# Events	9,549	15,588
# Users	46,895	59,989
# Groups	398	631
# Tags	15,819	21,228
# Venues	2,396	4,507
Avg. Events/month	795.75	1,299
Avg. Participants/Event	17.65	13.68
Avg. Events/User	3.59	3.55
Avg. Members/Group	382.71	256.76
Avg. Groups/User	3.16	2.74
Avg. Tags/Group	10.62	10.17
Avg. Tags/User	14.44	18.42

to compare the performances of different event-to-user recommendation methods, and use test data sets in different months, *e.g.*, 1st-month, 3rd-month and 6th-month test data, to illustrate the effect of temporal factor on user’s event participation behaviors.

3.4.1.2 MovieLens + IMDb/Rotten Tomatoes Dataset

This is the MovieLens dataset from GroupLens research group³ extended by using additional information from IMDb⁴ and Rotten Tomatoes⁵ websites. The dataset can be found in [119]. The statistics for the MovieLens dataset are shown in Table 3.2.

3.4.1.3 Yahoo! Music Dataset

This is the music dataset provided as part of Yahoo! Webscope program⁶ and used in 2011 KDD Cup [120]. From the raw dataset, we randomly select 10K users and all other entities related to users. Some statistics for the Yahoo! Music dataset can be found in Table 3.2.

³<http://www.grouplens.org>

⁴<http://www.imdb.com>

⁵<http://www.rottentomatoes.com>

⁶http://research.yahoo.com/Academic_Relations

Table 3.2: MovieLens and Yahoo! Music datasets

MOVIELENS DATASET		YAHOO! MUSIC DATASET	
# Users	2,113	# Users	10,000
# Movies	10,197	# Tracks	216,240
# Genres	20	# Albums	51,884
# Directors	4,060	# Artists	18,270
# Tags	13,222	# Genres	537
# Actors	95,321	Avg. Tracks/User	246.48
Avg. Actors/Movie	22.78	Avg. Albums/User	107.09
Avg. Tags/User	22.70	Avg. Genres/User	20.10
Avg. Tags/Movie	8.12	Avg. Tracks/Album	6.45
Avg. Movies/User	404.92	Avg. Tracks/Artist	16.25
Avg. Users/Movie	84.64	Avg. Genres/Track	2.55

For the MovieLens and Yahoo! Music datasets, the process of creating training and testing data is similar to that of Meetup dataset.

3.4.2 Evaluation Metrics

We use two metrics: precision and recall, denoted by $p@N$ and $r@N$, respectively, where N is number of recommendation results. In particular, the precision and recall of each test case are computed and the overall precision and recall are obtained by averaging these scores of all test cases.

3.4.3 Baseline Methods

Besides common baseline methods, we compare HeterS with baseline methods developed for each of the recommendation problems. We choose the group-to-user recommendation task to describe baseline methods, their implementation in other problems (except event-to-user recommendation) are similar.

- **CF**: This is user-based Collaborative Filtering method. For group-to-user recommendation, the user similarity is calculated based on the matrix $\mathbf{A}_{GU}$. In event-to-user recommendation, top- N most similar users to the event's seed users are returned as results.

- **BPR**: The Bayesian Personalized Ranking [118] based matrix factorization method is performed on matrix $\mathbf{A}_{GU}$ and recommendations are made based on the factorization results. In event-to-user recommendation, BPR factorizes user-user matrix and computes user similarity based on their latent features.
- **RWR**: RWR is run on a group-group interaction graph weighted by number of common users, where the groups of each test user are treated as query nodes. In event-to-user recommendation, RWR runs on weighted user-user graphs, where seed users are query nodes.
- **tRWR**: This event-to-user recommendation baseline is a variation of RWR where the event time information is incorporated into edges [116].
- **Hete-CF**: This is an MF-based method, in which weights learned from meta-paths are used as regularization terms [75]⁷. We choose meta-paths such that every entity type is included in at least one path.
- **PTARMIGAN**: This is the method proposed by Zhang et al. [26] for group-to-user recommendation.
- **PIT**: This is the method proposed by Liu et al. [121] for tag-to-group recommendation.

Finally, we consider two other methods derived from our proposed model:

- **full_RWR**: This method performs RWR on our constructed heterogeneous graph, and it treats the different types of edges (and nodes) in the same way.
- **uni_HeteRS**: This method is HeteRS without parameter learning part. In this method, parameters in the same equation, *i.e.* $\{\alpha_{Ni}\}_{\forall i \in \{U, E, G, T, V, S\}}$, are assigned with values uniformly. We compare with this variation of our method to evaluate the effectiveness of our learning process.

⁷The code of Hete-CF is available from the paper [75]

There are other methods proposed for heterogeneous data [19, 29, 72] which are presented in Section 2. However, most of them cannot be used in our experiments for different reasons. For example, method in [29] is only for star-structured graphs and cannot be applied in general heterogeneous graphs; the method proposed by Zhang et al. [72] needs annotated data, which is not available in our datasets. Only the method in [19], namely HeteroMF, is suitable for our experiments. However, its time complexity of its training process is very high that it could not finish in several days on our datasets. Therefore, we do not include it in our experiments.

3.4.4 Parameter Settings

There are three parameters in HeteRS: time-decay η , error approximation β and learning rate lr . Parameter β in Eq. (3.12) controls the error of approximating $\mathbb{1}(\cdot)$. The value of β should be chosen carefully, because inappropriate β can lead to the failure to optimize the AUC [116, 122]. By using tuning set, we set β to 10^3 for event-to-user and group-to-user recommendation, set β to 10^4 for tag-to-group and movie-to-user recommendation, and set β to 10^6 for track-to-user recommendation. Learning rate lr relies on β , and is used to balance between convergence and the speed of learning. We set lr to 0.1 for event-to-user and group-to-user recommendation and to 1 for remaining recommendation tasks. For each task, we train our model with 50 epochs, as we observe that after 30-35 epochs the objective function becomes stable.

All the parameters of baseline methods are empirically set to optimal values on the tuning sets.

3.4.5 Group-to-user Recommendation

Figure 3.3 shows the results on group-to-user recommendation problem. In this problem, the input of each test is a user. We observe that our method HeteRS outperforms PTARMIGAN, which also exploits heterogeneous information, significantly on both datasets. For example, HeteRS outperforms PTARMIGAN by 34% and 39% for p@5 and r@5, respectively, on CA dataset, and the improvement is statistically significant

Figure 3.3: Performance of methods in group-to-user recommendation

(p -value < 0.01). This may be because HeterRS represents the interactions between different entities better than the linear latent factor model in PTARMIGAN. Moreover, HeterRS is significantly better than full_RWR on both datasets, which demonstrates the effectiveness of our model on heterogeneous graphs. On the other hand, HeterRS always achieves better results than uni_HeterRS, which means that our learning scheme is necessary for our model to obtain the better performance. Finally, we can see that full_RWR outperforms RWR on both datasets, because the former is performed on a heterogeneous graph. The performance of CF and BPR is also poor, and comparing all approaches we find the methods that can take advantage of additional information, *e.g.*, tags and events, are able to produce better results than others. However, there is one exceptional case where the accuracy of Hete-CF is extremely worse. This results from the fact that Hete-CF performs the full-matrix factorization, which has proves to very ineffective for sparse and implicit feedback data ([43, 118]) as in the group-to-user recommendation problem.

Figure 3.4: Performance of methods in tag-to-group recommendation

3.4.6 Tag-to-group Recommendation

This task takes a group as input and returns the most likely tags the group may use. Figure 3.4 shows the results of different methods for this recommendation problem. From the figure, we observe that HeteRS outperforms baseline methods, by at least 90% on both datasets in terms of $p@5$ and the p -value of t-test is smaller than 0.01. This could be because HeteRS benefits from exploiting additional information including tags from group members or from participants of group events, while the other baseline methods except for full_RWR do not. This also explains why full_RWR yields good performance compared to the other baselines since full_RWR also uses all the different types of information to make recommendations. We also observe that PIT performs worse than other baselines. It could be because groups in Meetup are usually large, in which most users have limited historical data (*e.g.*, only participate in one group). Therefore, the impacts learned from PIT for these users are not reliable, which results in the poor performance of PIT. Once again, the accuracy of Hete-CF

Figure 3.5: Performance of methods in event-to-user recommendation

is the worst among all methods due to the ineffectiveness of full-matrix factorization on implicit feedback data. Finally, the results of uni_HeteRS are significantly worse than those of HeteRS and even worse than the baseline method full_RWR. This means that if we do not carefully set parameters for our model, its performance could be diminished dramatically.

3.4.7 Event-to-user Recommendation

The seed users, group, venue and day of the week of each test event are taken as inputs, and users who are interested in the event are targets.

The results are shown in Fig. 3.5. In this experiment, the number of seed users is set to 5. We observe from this figure that, the baseline method RWR always has better performance than CF and BPR. On the other hand, tRWR, full_RWR, uni_HeteRS and HeteRS always outperform CF, BPR and RWR, because they take the events' importance into account while the others do not. In particular, HeteRS outperforms RWR by 23% and 16% on NYC and CA data, respectively, in terms of $p@5$.

Figure 3.6: Performance of methods in movie-to-user recommendation

Moreover, HeteRS always achieves better results than tRWR, which cannot exploit the periodical behavior patterns of users as HeteRS does. The effect of this factor is analyzed in more detail in Section 3.4.10. In this experiment, full_RWR performs well, which again confirms the benefits of using additional information to produce better recommendation results. On NYC dataset, HeteRS outperforms the second best method full_RWR with p -value < 0.05 , which is statistically significant. However, the difference between two methods on CA dataset is less significant, which could be because user behaviors in CA are less periodical than in NYC (more details in Section 3.4.10). The baseline Hete-CF cannot be used in this problem since recommended events are totally new, *i.e.*, not in the training data, and hence we do not have latent factors for those events in order to make recommendation.

3.4.8 Movie-to-user Recommendation

The movie-to-user recommendation problem is to find movies that a given user would be interested in. Figure 3.6 shows the experimental results of different movie-to-user recommendation methods on MovieLens + IMDb/Rotten Tomatoes dataset. We observe that HeteRS outperforms other methods, with the relative improvement is at least 18% and 24% in terms of $p@5$ and $r@5$, respectively. Again, uni_HeteRS performs much worse than HeteRS, which proves that the parameter learning process is necessary for our model.

Figure 3.7: Performance of methods in track-to-user recommendation

3.4.9 Track-to-user Recommendation

The track-to-user recommendation problem is to recommend music tracks to a given user. The experimental results of methods in this problem are shown in Figure 3.7. Similar to other problems, HeteRS still outperforms the chosen baseline methods in this problem. Although uni_HeteRS is significantly better than other methods, it is still worse than HeteRS by 16% in terms of both $p@5$ and $r@5$. Note that we cannot report the results of Hete-CF because the method has a high time complexity and could not finish training in reasonable time for large dataset like Yahoo! Music.

3.4.10 Interpretation of Transition Parameters

In all these experiments, HeteRS works better than uni_HeteRS. This demonstrates that our parameter learning process is necessary and useful. In this subsection, we investigate the transition parameters α_{NM} to examine the roles of different types of entities in our recommendation problems. Due to the space limitation, we only show the values of learned parameters for event-to-user recommendation problem on both NYC and CA datasets and for group-to-user recommendation problem on NYC dataset in Table 3.3.

From Table 3.3, for event-to-user recommendation problem on NYC, we observe that the value of α_{ES} is the largest, 0.44, among those of transition parameters from other types to event type E, which means that users in NYC tend to join events

Table 3.3: Transition parameters

α_{NM}		Recommendation problems		
N	M	Event-to-user NYC	Event-to-user CA	Group-to-user NYC
E	G	0.34	0.44	0.38
E	S	0.44	0.37	0.25
E	U	0.11	0.10	0.12
E	V	0.11	0.09	0.25
G	E	0.01	0.01	0.27
G	T	0.01	0.01	0.31
G	U	0.01	0.01	0.42
G	q_g	0.97	0.97	N/A
S	E	0.88	0.73	N/A
S	q_s	0.12	0.27	N/A
T	G	0.72	0.83	0.65
T	U	0.28	0.17	0.35
U	E	0.33	0.37	0.11
U	G	0.50	0.43	0.11
U	T	0.16	0.19	0.05
U	q_u	0.01	0.01	0.73
V	E	0.99	0.80	N/A
V	q_v	0.01	0.20	N/A

periodically. On the other hand, parameter α_{ES} has smaller value in CA dataset, which indicates that the user activities in CA is less periodical than in NYC. This could be the reason why the improvement we achieved in event-to-user recommendation on CA dataset is less significant than on NYC dataset. On both datasets, the location information plays a minor role as the weight α_{EV} is small, 0.11 for NYC and 0.09 for CA dataset. By analyzing data, we observe that two events held in the same venue always belong to the same group. This means that the role of locations is already contained by that of groups. Moreover, most of the new events take place in new venues, and this makes the location information less useful. For group-to-user recommendation, we can see that the weights for nodes of type G largely come from those of U. Hence, we mostly rely on similar users when recommending groups to a target user. Note that the value of α_{GT} is high, *i.e.*, tags have a large role in group-

to-user recommendation. Also, we can see that the weight of T is mainly contributed by both G (α_{TG} is 0.65) and U (α_{TU} is 0.35). These observations are because users usually choose groups with the same topics (represented by tags) as their current groups or their interests.

3.4.11 Evaluating Fast Learning Method

In this section, we examine the efficiency of the fast learning method proposed in Section 3.2.3.

The fast learning method has one parameter: the storage threshold ξ , which determines the lower bound of entries in the initial vectors to be kept in compressed form. There is a trade-off between the time cost and space cost when identifying the value of ξ : if parameter ξ is set small, many vector entries need to be stored (space cost is worse), but the reconstructed vectors are close to the original ones, therefore it needs fewer iterations to calculate MMC than the case of large ξ (time cost is better). Figure 3.8a shows the comparison results of total running time between original learning process and the fast learning method with different thresholds ξ in the tag-to-group recommendation on CA dataset. Note that we observe similar results in other recommendation problems and on other datasets. As we can see, when the number of epochs (*i.e.*, one pass through the training data) increases, the fast learning method scales much better than the original learning process. Moreover, when we set the threshold ξ smaller, the total time of the fast learning method decreases as expected. To investigate deeper the effect of threshold ξ on the performance of fast learning method, we compute the average number of iterations needed to execute MMC in each epoch, and show in Fig. 3.8b. It can be seen that, after 20 epochs, the fast learning method requires fewer iterations, approximately 6 iterations with $\xi = 10^{-3}$, than original learning process, which needs about 15 iterations to compute MMC. Furthermore, the average number of iterations of fast learning method decreases when parameter ξ is smaller.

Finally, we evaluate the space efficiency of the fast learning method. For each training instance, we compute its compression ratio, which is the proportion of en-

Figure 3.8: Performance of fast learning method with different thresholds ξ and original learning process in tag-to-group recommendation problem on CA dataset.

Table 3.4: Compression ratio

ξ	Epoch				
	10	25	50	75	100
10^{-3}	0.20%	0.19%	0.19%	0.19%	0.18%
10^{-4}	1.87%	1.49%	1.49%	1.60%	1.60%
10^{-5}	18.25%	16.16%	15.17%	13.86%	12.78%

tries of initial vectors being kept in the compressed form; then report the average compression ratio of all training instances in each epoch in Table 3.4. We can see from the table that the number of vector entries stored by the fast learning method is considerably small, only accounting for 0.2% and 1.5% of whole vectors when ξ is 10^{-3} and 10^{-4} , respectively. This demonstrates the skewness of initial vectors, where there are very few entries with large values. When ξ is set smaller, 10^{-5} , the space cost, which is about 15%, is less efficient, but we can save more computation time as shown in Fig. 3.8a. Overall, our fast learning method significantly improves the efficiency of the original learning process while requiring minor storage space.

3.4.12 Evaluating Approximation Method

In Section 3.2.4, we propose an approximation method for HeteRS. We evaluate the efficiency of HeteRS with approximation method (Approximation HeteRS) by comparing with the one using iterative method (Iterative HeteRS) in this section. Note

Figure 3.9: Performance of Approximation HeteRS with different parameter ϵ on CA dataset

that in previous experiments, we use iterative method for HeteRS.

3.4.12.1 Parameter Setting

The approximation method has only one parameter: the preference threshold ϵ , which is the minimum preference a node needs to propagate to its adjacent nodes. If a node has preference smaller than ϵ , it will be ignored and no preferences are sent to its neighbors. There is a trade-off between the accuracy and the running time when setting ϵ : if we set ϵ large, the preference propagation process stops early, and only a small fraction of nodes get values; hence, the running time is short but the accuracy may be low. Figure 3.9 shows the performance of Approximation HeteRS in terms of p@5, r@5 and total running time in three recommendation problems on CA dataset with different preference threshold ϵ . We can see that when ϵ becomes larger, the running time is shorter, while the accuracy decreases. The precision and recall of Approximation HeteRS become stable when ϵ is smaller than 10^{-3} and 10^{-2} in tag-to-group and event-to-user recommendation, respectively. The accuracy of Approximation HeteRS in group-to-user recommendation still increases when ϵ is below 10^{-3} ; however, it suffers from significant increase in running time. In the rest of the experiments, we set ϵ to 10^{-3} because Approximation HeteRS balances well between accuracy and running time.

Figure 3.10: Comparison between Iterative and Approximation HeteRS in terms of effectiveness and efficiency on CA dataset

3.4.12.2 Experiment for Scalability

To evaluate the scalability of the proposed algorithms with the size of EBSN graph, we use training sets of different sizes, *i.e.*, 6-month, 12-month and 18-month training sets that include events in last 6, 12 and 18 months respectively. Figure 3.10 shows the results of both Iterative and Approximation HeteRS in terms of p@5 and total running time on CA of different sizes. The total number of nodes/edges of EBSN graphs corresponding to 6-month, 12-month and 18-month training sets is 48K/1M, 102K/2.4M and 136K/3.3M, respectively. From Fig. 3.10, it is observed that Approximation HeteRS is always at least 3 times faster than Iterative HeteRS in all recommendation problems, and the improvement is almost an order of magnitude on the largest EBSN graph. When the size of training data, *i.e.*, the EBSN graph, becomes larger, the computation time of both algorithms scales well, but Approximation HeteRS scales much better. We also observe that Approximation HeteRS achieves similar accuracy as Iterative HeteRS, and it even produces better results in tag-to-group recommendation problem. This improvement probably comes from the locality property of our approximation method: it rarely reaches nodes that are far from query node, hence reducing the number of irrelevant nodes in final results. Overall, the proposed approximation method greatly improves the efficiency of HeteRS while achieving similar accuracy as Iterative HeteRS.

Compared to Random Walk-based methods, *e.g.*, fullRWR, uniHeteRS, the Approximation HeteRS achieves similar efficiency improvement as the running time of

Figure 3.11: Performance of Global HeteRS and Enhanced HeteRS with different number of clusters.

those methods are roughly equal to that of Iterative HeteRS. Other baselines, *e.g.*, CF, BPR and RWR, always have shortest response time; their accuracy, however, is considerably worse, as shown in Figs. 3.3-3.5. Due to the space limitation, we do not show the comparison results of Approximation HeteRS with other methods.

3.4.13 Evaluating Enhanced HeteRS

In this section, we evaluate the recommendation of Enhanced HeteRS, which is proposed in Section 3.3. We compare Enhanced HeteRS models to Global HeteRS, which only has one parameter set, in two recommendation problems: movie-to-user and group-to-user recommendation.

As discussed in Section 3.3, in movie-to-user recommendation, we compute the similarity between two users based on their movies. In group-to-user recommendation, however, we cannot compute similarity between users based on their common groups because users usually join very few groups, averagely 2 to 3 groups (see Table 3.1), which makes it unreliable to compute similarity based on their groups. Worse still, for some users, we do not have their group information. To overcome this situation, we cluster users based on their similar events, because of two reasons: (i) each user joins at least one event, (ii) from [4] we know that there is a strong correlation between online and offline activities of users in event-based social networks, *i.e.*, users who participate in similar events also join similar groups. However, the user-event data is still very sparse as users join just 3 events averagely (Table 3.1). Moreover, the

number of users is enormous, *e.g.*, nearly 60K users in CA dataset (Table 3.1), which makes the clustering process inefficient. To solve these two issues, we extract and cluster users who have joined at least 10 events; and the remaining users are assigned to the closest cluster.

Figure 3.11 shows the comparison results in terms of p@5 between Global HeteRS and Enhanced HeteRS with different number of clusters. We can see that in both recommendation problems, Enhanced HeteRS achieves different performance results when we vary the number of clusters c . Moreover, Enhanced HeteRS always obtains better accuracy than Global HeteRS in both problems. Particularly, in movie-to-user recommendation, starting c from 2, the performance of Enhanced HeteRS rises up and peaks when $c = 20$, at which it outperforms Global HeteRS by 9.6%, before suffering from a gradual decrease. Similarly, Enhanced HeteRS achieves best performance when $c = 10$ in group-to-user recommendation. This is because when the number of clusters is small, Enhanced HeteRS cannot distinguish users with different behavior patterns very well, and hence its improvement over Global HeteRS is insignificant. On the other hand, when the number of clusters is large, there is not enough training data for each model, resulting in the decrease of overall performance.

We also conducted the experiments of Enhanced HeteRS in tag-to-group recommendation problem, but it did not achieve improvement over Global HeteRS. By investigating the data, we find that the cluster structure of groups is abnormal, *i.e.*, only one or two clusters contain many groups while other clusters (outliers) have only one group. This is because although groups have different topics, they usually use same tags with general meanings, such as “Friends” or “Social Networking”, and are put in one cluster. On the other hand, many groups have very few or no tags, and thus they have very small or zero similarities with other groups and they form separate clusters. For event-to-user recommendation, we cannot apply Enhanced HeteRS because target events are always new, *i.e.*, they have never been seen in the training data and thus no models are built for them.

3.5 Conclusion

In this chapter, we introduce a general model to solve recommendation problems in heterogeneous networks. First, we transform the recommendation problems into node proximity calculation problem w.r.t. query inputs on a heterogeneous graph, and employ multivariate Markov chain to solve it. Then, we propose an optimization scheme to learn transition parameters for our model. The learned parameters not only enable our model to achieve superior performance, but also help us to understand roles of different types of entities in each recommendation problem. The experimental results demonstrate the proposed model outperforms the baseline methods considered in the experiments in all recommendation tasks.

Chapter 4

A General Model for Out-of-town Region Recommendation

Out-of-town recommendation is a special sub-problem of POI recommendation, which aims to recommend locations to users who travel out their hometowns. All the previous works on out-of-town recommendation focus on improving the accuracy of recommending individual POIs to users. However, as discussed in Section 1.3.2, it is more useful to recommend a region with several POIs than just singular POIs when users travel out their hometowns. Therefore, in this thesis, we propose a novel region recommendation problem that aims to find regions interesting to a user.

There are two challenges to solve the region recommendation problem. The first challenge is how to measure the attractiveness of a region to a user. One intuitive way is to compute the user’s attractiveness score of each POI in the region and then aggregate them as the score of the region. However, this approach considers each POI independently and ignores the influence between POIs. In contrast, the user’s decision to visit a place is also affected by how much she is interested in nearby POIs [43,46,47]. In other words, nearby POIs can reinforce each other to attract users. For example, when choosing hotels to stay, travelers usually prefer the ones that are near their interesting places, such as cinemas, restaurants and tourist attractions (museums, theaters, etc.). Considering POIs individually neglects such influences between POIs. As a result, it is necessary to consider the POI influences when searching the optimal region. As a result, in this thesis, we propose a general procedure to estimate the region’s attractiveness with the POI interaction being taken into account.

The second challenge is the efficiency of searching the optimal region. Given a set of POIs in a 2-dimension space, there are *infinite* number of ways to place a region of a given size. Obviously, we cannot check all the cases to find the optimal region, and hence a more efficient approach is required to find the result region quickly. In this thesis, based on the sweeping algorithm [123, 124], we introduce a simple but efficient method to find the optimal region. Specifically, we convert the original problem into a geometric intersection problem, and then apply the sweeping algorithm. As a result, the region searching problem can be solved in $\mathcal{O}((N + M) \log N)$, where N and M are the number of POIs and their pair-wise interactions, respectively. Moreover, to achieve better efficiency, a constant-bounded approximate algorithm is developed for the region search problem.

4.1 Observations

First of all, we present some observations from real-world Foursquare dataset [125] that motivate the region recommendation problem. We first infer home locations of all users, and keep users and locations located in the US. For each user, the region within $r = 100km$ around her home location is considered her hometown. Details of the preprocessing step are described in Section 4.4.1.

First, we investigate the user's distribution of out-of-town check-ins as follows. Given a length of d kilometers and a number n , we compute the fraction of users who have one region of size $d \times d$ with at least n visited POIs, and then plot Fig. 4.1a. The figure shows that a large proportion of users have multiple check-ins in a small region. For example, more than 60% of users have visited at least 2 POIs in a $1km \times 1km$ region. When the size of region grows, the fraction of those users obviously increases, which is at nearly 80% when $d = 5km$. For larger value of n , the user ratio is decreased but still at the high number. For example, there are still 25% of users who have at least 4 visited POIs located in an $1km \times 1km$ region. This implies that users tend to go to locations within a small region when they go outside their hometowns. To investigate the reason, we compute the sizes of regions containing check-ins of same

(a) Fraction of users having a $d \times d$ region with at least n POIs. (b) CDF of region sizes made by check-ins in one and two days, respectively.

Figure 4.1: Observations on out-of-town check-ins.

users in t consecutive days and plot their cumulative distribution functions (CDF) for $t = 1$ and $t = 2$ in Fig. 4.1b. The figure shows that 10% of one-day check-ins occur in a region smaller than $1\text{km} \times 1\text{km}$, and this number is about 45% when $d = 10\text{km}$. When $t = 2$, the CDF line is lower, which means that users tend to visit larger regions when the duration is longer. This confirms our belief that users often visit small regions due to limited time. Overall, these observations motivate us to consider the problem of recommending regions to users.

Definition 4.1 *Given a user, a set of POIs in spatial space and a query size of $d \times d$, the **Region Recommendation (RegRS) Problem** aims to find a $d \times d$ region that the user is most likely to visit. Here, we consider a region is visited by a user when she visits at least one POI in that region.*

Instead of recommending individual POIs, the region recommendation problem aims to recommend a group of POIs that are located within a region.

In previous work [47, 126], regions are predetermined by using some spatial partitioning algorithms. Although our recommendation model can be easily applied for such set of fixed regions, we do not divide the space into fixed regions but instead consider any rectangle spatial area as a region. This is more general and flexible as we are able to find meaningful and crucial regions that can lie inside, contain or intersect those fixed regions. Moreover, we assume that a region has the square shape $d \times d$ for

Figure 4.2: Fraction of regions with different number of visited POIs.

simplicity. One can define a rectangle of size $a \times b$ as a region. However, it does not lose the generality because we can easily scale the space to make the region become a square. Also, the size of the region is given by the user, and different users can issue different query sizes depending on their distance and time constraints.

4.2 Region Recommendation

4.2.1 Proposed Model

In this section, we present our model for recommending regions to users. Consider a region R containing multiple POIs: $R = \{p_i\}$. Let $S_{i_1 \dots i_k}$ be the satisfaction of user u when she visits POIs $p_{i_1} \dots p_{i_k}$ in R . Let $P_u(p_{i_1}, \dots, p_{i_k} | k)$ be the probability that user u visits POIs $p_{i_1}, \dots, p_{i_k}$ in R given she will visit k distinct POIs. For convenience, we drop the notation u unless necessary, *i.e.*, we simply use $P(p_{i_1}, \dots, p_{i_k} | k)$. Then, the total expected satisfaction of u , S_{total} , when she visits the region R can be written as

$$\begin{aligned}
 S_{total} = & \left(\sum_{p_{i_1} \in R} S_{i_1} P(p_{i_1} | k=1) \right) P(k=1) \\
 & + \left(\sum_{p_{i_1}, p_{i_2} \in R} S_{i_1 i_2} P(p_{i_1}, p_{i_2} | k=2) \right) P(k=2) + \dots \\
 & + \left(\sum_{p_{i_1}, \dots, p_{i_{|R|}} \in R} S_{i_1 \dots i_{|R|}} P(p_{i_1}, \dots, p_{i_{|R|}} | k=|R|) \right) P(k=|R|).
 \end{aligned} \tag{4.1}$$

Equation (4.1) covers all the cases where user u may visit from 1 to $|R|$ POIs in R . However, we observe that, although a user may have a region with many visited POIs

(see Section 4.1), most of her visited regions have only 1 or 2 POIs. To demonstrate this, given a user, we generate randomly some regions with the size of $4km \times 4km$ such that each region covers at least one of her visited POIs. Then, we compute the proportion of regions with different number of visited POIs, and plot the distribution in Fig. 4.2. From the figure, we can see that most of the regions have less than 3 POIs visited by a user. Therefore, it is sufficient to consider only the first- and second-order terms, and ignore other higher order terms of S_{total} :

$$S_{total} \approx \left(\sum_{p_{i_1} \in R} S_{i_1} P(p_{i_1} | k = 1) \right) P(k = 1) + \left(\sum_{p_{i_1}, p_{i_2} \in R} S_{i_1 i_2} P(p_{i_1}, p_{i_2} | k = 2) \right) P(k = 2). \quad (4.2)$$

Next, we will define two probabilities: $P(p_{i_1} | k = 1)$ and $P(p_{i_1}, p_{i_2} | k = 2)$. The probability $P(p_{i_1} | k = 1)$ represents the probability that user u visits p_{i_1} , *i.e.*, $P(p_{i_1} | k = 1) = P(p_{i_1} | u)$, which can be computed by using any recommendation model such as memory-based collaborative filtering [17, 40] or matrix factorization [42, 43, 47]. For the probability $P(p_{i_1}, p_{i_2} | k = 2)$, one simple way is to define this probability as $P(p_{i_1}, p_{i_2} | k = 2) = P(p_{i_1} | u) \cdot P(p_{i_2} | u)$, assuming that the user's decisions on visiting different POIs are independent. However, as discussed above, the user's interest on a POI could be affected by other nearby POIs, which means that the influences between POIs cannot be ignored. Hence, in our model, we define the probability as

$$P(p_{i_1}, p_{i_2} | k = 2) = \frac{1}{2} \cdot P(p_{i_2} | p_{i_1}) \cdot P(p_{i_1} | u) + \frac{1}{2} \cdot P(p_{i_1} | p_{i_2}) \cdot P(p_{i_2} | u), \quad (4.3)$$

where $P(p_{i_1} | p_{i_2})$ is the transition probability from p_{i_2} to p_{i_1} , which can be computed from historical check-in data as follows:

$$P(p_{i_1} | p_{i_2}) = \frac{\text{\#users who visited both } p_{i_1} \text{ and } p_{i_2}}{\text{\#users who visited } p_{i_2}}. \quad (4.4)$$

Probabilities $P(p_{i_1} | k = 1)$ and $P(p_{i_1}, p_{i_2} | k = 2)$ are defined simply based on users' check-in history. However, more sophisticated definitions of those probabilities can be given by integrating additional information. For example, we can boost up (or diminish) the probability of visiting two POIs if we know they are complementary

(or substitutable) for each other [22]. Moreover, other information, *e.g.*, categorical [94] or textual information [104], can be utilized to compute the visiting probability. In this thesis, we keep the simple definitions for those probabilities and leave any enhancement for future work.

Next, we define the satisfaction $S_{i_1 i_2, \dots, i_k}$. In general, the more POIs a user visited, the more satisfaction the user obtains. Hence, we set $S_{i_1} = 1$, $S_{i_1 i_2} = 2$, ..., and $S_{i_1 i_2, \dots, i_k} = k$ in this paper. We note that other satisfaction function can be defined in terms of different application needs. As a result, the expected satisfaction S_{total} can be rewritten as

$$S_{total} \approx \left(\beta \cdot \sum_{p_i \in R} P(p_i|u) + \sum_{\langle p_i, p_j \rangle \in O_R} P(p_i|p_j) \cdot P(p_j|u) \right) P(k=2), \quad (4.5)$$

where O_R is all the ordered pairs $\langle p_i, p_j \rangle$ in R , and $\beta = \frac{P(k=1)}{P(k=2)}$ is a user-dependent parameter. However, it is difficult to set β for each user, and hence, in our model, we set β to 1 for all users, since it gives our model the best performance.

Finally, since the goal of RegRS problem is to find the region that gives the maximum satisfaction to a user, it can be formulated as the following objective function:

$$R^* = \arg \max_{R \in \mathcal{R}} \sum_{p_i \in R} P(p_i|u) + \sum_{\langle p_i, p_j \rangle \in O_R} P(p_i|p_j) \cdot P(p_j|u). \quad (4.6)$$

Here, $\mathcal{R}$ is the set of all possible regions, and the term $P(k=2)$ is dropped as it is a constant given a user.

4.2.2 Searching Maximum Region

The goal of RegRS problem is to find the region R^* according to Eq. (4.6). In other words, we aim to locate the position of the maximum region, which satisfies the objective function Eq. (4.6). However, there are *infinite* possible ways to place such region in the space. Therefore it is impossible to check every region to find the solution.

In this section, we present our algorithm, namely RegRS algorithm, to efficiently find the maximum region. First, we define some notations. Besides N points $\mathcal{P} =$

$\{p_i \mid 1 \leq i \leq N\}$, each of which represents a POI in $\mathbb{R}^2$, we define a set of M edges $\mathcal{E} = \{e_{ij}, 1 \leq i, j \leq N\}$, where each of edges connects two points $\langle p_i, p_j \rangle$ in P . Each point p_i and edge e_{ij} have positive weights $w_i = P(p_i|u)$ and $l_{ij} = q_{ij} \cdot w_i + q_{ji} \cdot w_j$, respectively, where $q_{ij} = P(p_j|p_i)$. Our problem is to find the maximum rectangle R^* of given size $d \times d$ that maximizes the objective function Eq. (4.6), which can be rewritten as:

$$s(R) = \sum_{p_i \in R} w_i + \sum_{\substack{p_i, p_j \in R, \\ e_{ij} \in \mathcal{E}}} l_{ij} \quad (4.7)$$

The main idea of the solution is to reduce the original problem to the geometric intersection problem, and then a space *sweeping* algorithm is performed to solve the new problem. Our method is based on the technique to solve the maximum object enclosing rectangles problem [123, 124].

4.2.2.1 Geometric intersection problem

The first step of the algorithm is to reduce the RegRS problem to the geometric intersection (GI) problem [123, 124]. The reduction is performed as follows: for each point $p_i \in P$, we draw a $d \times d$ rectangle centered at p_i , and denoted by r_{p_i} . The N constructed rectangles $\{r_{p_i}\}$ intersect each other, and thus create multiple disjoint regions, each region is the intersection of a unique set of rectangles. For each disjoint region D , let D_R the set of points p_i , whose rectangles r_{p_i} form the region D . For example, in Fig. 4.3, the crossed area is a disjoint region created by rectangles r_{p_1} , r_{p_2} and r_{p_3} , and hence $D_R = \{p_1, p_2, p_3\}$. Given any point o in a disjoint region D , the $d \times d$ rectangle r_o centered at o will cover all the points in D_R . This is because if a point o is covered by a rectangle r_{p_i} , then r_o also covers p_i . Let the weight of D_R be calculated by Eq. (4.7), where $R = D_R$. As a result, the goal of the GI problem is to find the disjoint region D_R^* with the maximum weight. Then, any point o^* in the D_R^* can be returned as the center of the maximum region R^* for the RegRS problem.

The reduction gives us a new way to solve the RegRS problem. Finding maximum disjoint region D_R^* can be done by extending sweeping algorithm [123, 124].

Figure 4.3: Reduction to GI problem.

4.2.2.2 Sweeping algorithm

The main idea of the algorithm is to use a horizontal line l to scan the space from top to bottom. The process begins when the sweeping line l is at the upmost line and starts sweeping down. When l encounters and goes inside a rectangle r_{p_i} , r_{p_i} cuts l at two intersection points, l_i and r_i , by its two vertical edges, and r_{p_i} is said to be active. When l exits r_{p_i} , two corresponding intersection points are removed, and r_{p_i} becomes inactive. At any moment, if there are currently k active rectangles, *i.e.*, l is inside k rectangles, there are $2k$ intersection points on l , and those points create $2k - 1$ disjoint intervals on l . Figure 4.4a illustrates one moment when the sweeping line lies across 4 rectangles, and 7 intervals, I_1 to I_7 , are created accordingly.

An interval I_k is said to be covered by an active rectangle r_{p_i} if it is inside the window $[l_i, r_i]$ created by r_{p_i} . Then we can define weight $s(\mathcal{T}_k)$ for an interval I_k , where $s(\mathcal{T}_k)$ is computed by Eq. (4.7), and $\mathcal{T}_k$ is the set of points whose rectangles cover I_k . When the sweeping line l encounters a rectangle r_{p_i} (Fig. 4.4b), not only new intervals are created, but also weights of all intervals covered by r_{p_i} are increased. The update includes two steps corresponding to two terms of Eq. (4.7). First, all the intervals covered by r_{p_i} (*i.e.*, the ones inside the window $[l_i, r_i]$) are added up by the weight w_i of p_i . Second, if p_i links to any point p_j (*i.e.*, $e_{ij} \in \mathcal{E}$) and two rectangles r_{p_i} and r_{p_j} overlap each other, all the intervals in the overlapping window, which is

Figure 4.4: The illustration of the sweeping algorithm.

$[l_j, r_i]$ in Fig. 4.4b, are added up by l_{ij} . Similarly, when l exits r_{p_i} , some intervals disappear, and weights of all intervals covered by r_p are decreased.

The sweeping process finishes when l has crossed all the rectangles. If I_k is the interval with highest weight in the whole process, then we can return any point in I_k as the optimal point o^* of the GI problem. For example, in Fig. 4.4a, the interval I_3 is the one with largest weight and is returned as the result.

Complexity. In order to efficiently maintain intervals (*i.e.*, creating, updating and deleting intervals) during the scanning process, a special tree-based data structure, Interval tree [124], is used in our implementation. It takes $\mathcal{O}(N \log N)$ to construct the Interval tree. For each point p_i , the tree is updated when l encounters and exits r_{p_i} , each of which takes $\mathcal{O}(\log N)$. Similar actions are performed for each edge e_{ij} . Note that not all edges in $\mathcal{E}$ are considered, only those that can be covered by a $d \times d$ rectangle are processed. Overall, the time complexity of RegRS algorithm is $\mathcal{O}((N + \overline{M}) \log N)$, where $\overline{M}$ is the number of processed edges.

4.3 Approximate Solution

Based on the time complexity analyzed above, runtime performance of RegRS degrades significantly when the size of query rectangle increases (which is also demonstrated in experiments in Section 4.4.5). This is because a larger query rectangle

can cover more edges, and hence, more edges will be processed, which will degrade the efficiency of the searching algorithm. In this section, we present an approximate algorithm to answer the RegRS problem efficiently. Firstly, we introduce the ϵ -approximate RegRS problem.

Definition 4.2 *Given an input $I = \langle \mathcal{P}, \mathcal{E}, w, l, d, \epsilon \rangle$, where $\mathcal{P}, \mathcal{E}, w, l, d$ are the input of the RegRS problem and ϵ is a real number such that $0 < \epsilon < 1$, the ϵ -**approximate RegRS problem** aims to return a $d \times d$ rectangle R_{appx} whose weight $s(R_{\text{appx}})$ is at least $\epsilon s(R^*)$, i.e., $s(R_{\text{appx}}) \geq \epsilon s(R^*)$, where R^* is the maximum region.*

4.3.1 Main Idea

The approximate algorithm is inspired by the following observation. Figure 4.5a shows the top 200 recommendation scores, $p(p_i|u)$, for one randomly picked user. It is clear that the distribution of recommendation scores follows a long tail distribution, i.e., most of them have very small values, which is a common phenomenon in recommendation systems. Since small-weighted POIs contribute little to the total weight of the region, intuitively if we ignore those points and keep only large-weighted ones, we can save searching time significantly without sacrificing much the accuracy. Moreover, we also need to consider edges as they also contribute to weights of regions. Particularly, since the edge e_{ij} 's weight is computed by $l_{ij} = q_{ji} \cdot w_i + q_{ji} \cdot w_j$, l_{ij} is likely to be large if w_i or w_j is large. Therefore, it is necessary to preserve all the edges of large-weighted points. As a result, neighbors of large-weighted points should be kept as well.

4.3.2 Solving Approximate RegRS Problem

Our approach is, instead of handling the whole space, we divide the space into $d \times d$ cells and use cells as spatial units to select points for approximate problem. Let us first define some notations used in our algorithms.

Definition 4.3 *Grid G_d is the set of vertical and horizontal lines that are defined as follows:*

$$G_d = \{(x, y) \in \mathbb{R}^2 \mid x = k_1 \cdot d \wedge y = k_2 \cdot d, k_1, k_2 \in \mathbb{Z}\}$$

Figure 4.5: Some observations

Apparently, G_d partitions the space into $d \times d$ cells, which are called *small cells*. Each small cell is uniquely identified by the pair of real numbers (k_1, k_2) , called the identity of the small cell, which are coefficients corresponding the left vertical line and top horizontal line of the cell, respectively. Subsequently, we define a *large cell* to be a $2d \times 2d$ cell constructed by 4 adjacent small cells as in Fig. 4.6. Large cells also have unique identity which is the identity of its top-left small cell. For any point $p = (x, y)$, it is easy to find the small cell with the identity $(\lfloor \frac{x}{d} \rfloor, \lfloor \frac{y}{d} \rfloor)$, and hence 4 large cells, that it falls in. Hereinafter, we will refer term cell to large cell, if there is no explanation.

It can be easily seen that any $d \times d$ rectangle, including the maximum rectangle R^* , is covered by a cell. Hence, our approach is to process each cell one by one. For a cell C , we *remove points from C , together with their edges, such that the maximum $d \times d$ rectangle returned by the RegRS algorithm on remaining points (and edges) is the answer of the ϵ -approximate RegRS problem on C* . Particularly, let c_{max} be the maximum $d \times d$ rectangle in C . Then, let S_C and S'_C be the weights of cell C before and after removing points respectively; and s_{max} and s'_{max} are defined similarly but for c_{max} . Then we want that

$$\frac{s'_{max}}{s_{max}} \geq \epsilon \quad (4.8)$$

 Figure 4.6: Dividing space by grid G_d

Set $\Delta_s = s_{max} - s'_{max}$ and $\Delta_S = S_C - S'_C$. Since removed points (and edges) can be outside c_{max} , it is obvious that $\Delta_s \leq \Delta_S$. We have:

$$\frac{s'_{max}}{s_{max}} = \frac{s_{max} - \Delta_s}{s_{max}} = 1 - \frac{\Delta_s}{s_{max}} \geq 1 - \frac{\Delta_S}{s_{max}} \quad (4.9)$$

Now, we assume that $\frac{s_{max}}{S_C} \geq \lambda$ where λ is a constant in $(0, 1]$, then

$$\frac{s'_{max}}{s_{max}} \geq 1 - \frac{\Delta_S}{\lambda S_C} = 1 - \frac{S_C - S'_C}{\lambda S_C} = \frac{1}{\lambda}(\lambda - 1 + \frac{S'_C}{S_C}) \quad (4.10)$$

From Eq. (4.10), the sufficient condition for (4.8) is:

$$\begin{aligned} \frac{1}{\lambda}(\lambda - 1 + \frac{S'_C}{S_C}) &\geq \epsilon \\ \Rightarrow \frac{S'_C}{S_C} &\geq 1 - (1 - \epsilon)\lambda \end{aligned} \quad (4.11)$$

The condition Eq. (4.11) means that, when removing points (and edges) from C , as long as S'_C is still not less than $(1 - (1 - \epsilon)\lambda)S_C$, the condition in Eq. (4.8) is always guaranteed.

4.3.2.1 Finding λ

To complete the condition (4.11), we need to find the value of λ , which is the lower bound of the ratio $\frac{s_{max}}{S_C}$. It is easy to prove that if cell C contains only points without edges, the lower bound is $\lambda = \frac{1}{4}$. Unfortunately, by the following theorem, a constant lower bound λ does not exist in the presence of edges.

Theorem 4.1 *When C contains both positively weighted points and edges, there does not exist a lower bound of the ratio λ . In other words, the ratio can be arbitrarily small.*

Proof: Below, we use the square-shaped 2×2 rectangle C for the ease of proof, but the statement is still true for the rectangles of any shape.

To prove Theorem 4.1, we consider two cases: 1) edge weights are *not bounded*, and 2) edge weights are *bounded by their two points*. In both cases, we show special examples when the lower bound of the ratio λ can be arbitrarily small.

Case 1: weights of edges are not bounded by their two ending points. In this case, we set up a special instance of cell C as follows: *every edge is a diagonal of one 1×1 rectangle c* (Fig. 4.7a). As a result, a rectangle c can cover two edges at most, which are its two diagonals. Note that, we can place *infinite* number of such edges in C . Assume that all edges have equal weights of 1, and c_{max} contains two edges, then:

$$\frac{s_{max}}{S_C} = \frac{s_{max}^{points} + 2}{S_C^{points} + E}$$

where s_{max}^{points} and S_C^{points} are total weight of points in c_{max} and C , respectively, and E is total number of edges in C . Now, if we set weights of points very close to zero, then edge weights will dominate point weights in rectangles, *i.e.*, $r_{max}^{points} \ll 2$ and $S_C^{points} \ll E$. This leads to $\frac{s_{max}}{S_C} \approx \frac{2}{E}$. This means that, if we add more edges into C , the ratio decreases accordingly. When $E \rightarrow \infty$, then $\frac{s_{max}}{S_C} \rightarrow 0$. In other words, λ can be arbitrarily small.

Case 2: weights of edges are bounded by their two ending points. The above proof is only valid when edge weights are arbitrarily large. However, in our region recommendation problem, we know that $l_{ij} = q_{ij} \cdot w_i + q_{ji} \cdot w_j \leq w_i + w_j$, because $q_{ij} = P(p_j|p_i) \leq 1, \forall i, j$. In other words, *the edge weight cannot be larger than sum of weights of its two points*. For the completeness of the proof, we need to consider this case as well.

The setup of the special instance is little more complicated. First, we place N points in the lower-left 1×1 rectangle of C such that those points form a $\frac{1}{4}$ arc of the

 Figure 4.7: Special point and edge arrangements for cell C .

circle with same center O of C . We call set of those points $\mathbf{D} = \{D_1, D_2, \dots, D_N\}$. For each point $D_i \in \mathbf{D}$, we create two more points E_i and F_i in the same positions but in the upper-right and lower-right 1×1 corner rectangles, respectively. Those points form two other point sets, namely $\mathbf{E}$ and $\mathbf{F}$. Obviously, $\mathbf{E}$ and $\mathbf{F}$ also create two $\frac{1}{4}$ arcs of the circles as in Fig. 4.7b. Then, for each triple $\langle D_i, E_i, F_i \rangle$, we create $N-1$ edges from D_i to some points in the line $\overline{E_i F_i}$ as in Fig. 4.7b. It is easy to see that a rectangle c covers an edge only when the point D_i of the edge is in the left edge of c . Finally, we set the weight of each D_i be one and other points be (very close to) zero. All the edge weights are one. Obviously the edge weights satisfy the above constraint.

There are 3 possibilities of the position of a rectangle c : 1) c 's lower-left corner is a point $D_i \in \mathbf{D}$ (Fig. 4.8a); and hence the upper-right corner is the corresponding point E_i . In this position, c 's weight is N (1 from D_i and $N-1$ from $N-1$ edges); 2) the second position is obtained from the first case by shifting c vertically down (Fig. 4.8b). When being shifted down, c contains some more points D_k but loses some edges. Since we are free to choose positions of edges, edges are created so that whenever c receives a point D_k when shifting down, it loses one edge. Therefore, the weight of c is still N ; 3) other positions. For those positions, c covers no edges and at most N points D_i , hence it has a weight of N at most.

Overall, the weight of c_{max} is $s_{max} = N$. Meanwhile, the weight of C is $S_C = N + N(N-1) = N^2$, since there are N points D_i and $N-1$ edges for each D_i . Therefore, $\frac{s_{max}}{S_C} = \frac{N}{N^2} = \frac{1}{N}$. This means that, if we add more points following this arrange-

ment, the ratio decreases, and when $N \rightarrow \infty$, then $\frac{s_{max}}{S_C} \rightarrow 0$.

Overall, in any case, we do not have a constant lower bound λ for the ratio $\frac{s_{max}}{S_C}$.

Theorem 4.1 tells us that we do not have a fixed value for the lower bound λ . However, in practice, when points and edges are randomly located in C , we find that the ratio cannot be too small. Particularly, from the Foursquare dataset used in our experiment (see Section 4.4), we pick randomly some users and randomly some $2d \times 2d$ regions for each user ($d = 1km, 2km, 5km \dots$), then find λ for each test case and plot the distribution of λ in Fig. 4.5b. From the figure, we can see that the minimum value of λ is around 0.08. Note that, we also obtain the same observation for another dataset used in Section 4.4. This means that, in practice, we can set λ to any number below 0.08. In our implementation, λ is set to 0.05.

4.3.2.2 Computing upper bound of S_C

For each cell C , in order to keep track the ratio S'_C/S_C , we need to compute the weight S_C , which is very time-consuming. We know that the weight S_C is computed as $S_C = s(C) = \sum_{p_i \in C} w_i + \sum_{p_i \in C} \sum_{p_j \in C} q_{ij} \cdot w_i = \sum_{p_i \in C} (1 + \sum_{p_j \in C} q_{ij}) \cdot w_i$. Since q_{ij} is the transition between two POIs, it is independent of the query user and can be computed offline. Therefore, for each point p_i , we precompute the quantity $q_i^{ub} = \sum_{p_j \in \text{neigh}(p_i)} q_{ij}$, which is the sum of weights of edges from p_i to all its neighbors (*i.e.*, points that have edges to p_i). Thus, the upper bound of S_C can be computed as $S_C^U = \sum_{p_i \in C} (1 + q_i^{ub}) \cdot w_i$. Then, S_C^U can be used to replace S_C in the condition Eq. (4.11). However, S_C^U is not a good upper bound estimation of S_C because q_i^{ub} is computed from all neighbors of p_i while C only covers a small subset of neighbors. This may lead to the situation where we have to choose all points in C to satisfy the condition Eq. (4.11). To better estimate S_C^U , for each p_i , we maintain a list of tuples $\langle d_j, q_{ij}^{ub} \rangle$, where d_j is the distance from p_i to a neighbor p_j , and q_{ij}^{ub} is similar to q_i^{ub} but computed from neighbors within the distance d_j from p_i . The list is sorted by d_j . Then, given the size $d \times d$ of C , for each p_i , we use binary search to find q_{ij}^{ub} where $d \leq d_j$, and set $q_i^{ub} = q_{ij}^{ub}$. This step will take $\mathcal{O}(\log N_i^{nb})$ for each p_i , where $N_i^{nb} = |\text{neigh}(p_i)|$.

 Figure 4.8: Two possible positions of a rectangle c in case 2.

4.3.3 Approximate Algorithm

Now, we are ready to describe our approximate algorithm (Algorithm 3). Overall, we firstly divide space by G_d (line 1), and then check each cell C in the decreasing order of upper bounds S_C^U (line 5). In each cell C , we iterate points in the decreasing order of w_i , and add it, with its neighbors, into the selection set $\mathcal{P}'$ (line 9) until current weight S_C' of C satisfies condition Eq. (4.11). The selection step terminates when unchecked cells cannot give the maximum rectangle (line 6). Finally, the sweeping algorithm is performed on the new set $\mathcal{P}'$ and R_{appx} is returned (line 12).

Time complexity. The most costly parts of the algorithm 3 are from computing upper bounds of all cells and running sweeping algorithm on $\mathcal{P}'$. Overall, its complexity is $\mathcal{O}(N \log \bar{N}^{nb} + (N' + M') \log N')$, where $\bar{N}^{nb}$ is the average number of neighbors of all points, N' and M' are the number of points in $\mathcal{P}'$ and their edges, respectively. Considering $\bar{N}^{nb} \ll N$, $N' \ll N$ and $M' \ll \bar{M}$, we can see that the approximation algorithm has a better time complexity than the exact RegRS algorithm.

4.4 Experiments

4.4.1 Experimental Settings

In our experiment, we use two public datasets, **Foursquare** [125] and **Twitter** [127]. For both datasets, we first infer users' home locations by following the strategy in [127]. Then, for each dataset, we extract the users and check-ins located in the US, and keep users with at least 5 visited POIs and POIs with at least 5 users. After preprocessing,

Algorithm 3: Approximate algorithm

Input: $\langle \mathcal{P}, \mathcal{E}, w, l, d, \epsilon \rangle$
Output: Approximate optimal rectangle R_{approx}

- 1 Divide space by G_d ;
- 2 Compute the upper bound weight S_C^U of each cell; $S'_C \leftarrow 0, \forall C$ //initial new weights of cells
- 4 $\mathcal{P}' = \{\}$;
- 5 **foreach** cell C from largest to smallest S^U **do**
- 6 **if** $S_C^U < \max\{S'_C\} \times \lambda$ **then**
 └ break;
- 7 **foreach** point $p_i \in C$ from largest to smallest w_i **do**
- 8 **if** $p_i \in \mathcal{P}'$ **then**
 └ continue;
- 9 $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{p_i\} \cup \text{neigh}(p_i)$;
- 10 Update S'_C ;
- 11 **if** $S'_C/S_C^U \geq 1 - (1 - \epsilon)\lambda$ **then**
 └ break;
- 12 Run sweeping algorithm on $\langle \mathcal{P}', \mathcal{E}', w, l, d \rangle$ and return R_{approx} ;

the Foursquare dataset contains 8,368 users and 19,241 POIs with 332,952 check-ins; and the Twitter dataset contains 31,378 users, 21,951 POIs and 456,578 check-ins. The POI distribution of two datasets is shown in Fig. 4.9.

Following [45], we consider the region that is 100km around a user's home location is her hometown. Then, we create the training and testing data as follows. Since not all users have out-of-town check-ins, a user is in testing data if she has at least 5 visited POIs in her hometown, and at least two of her out-of-town POIs are covered by a $d \times d$ region ($d = 0.5km$). For test users, their hometown POIs are put into training data, while their out-of-town POIs are added into testing data. For remaining users, all their visited POIs (both hometown and out-of-town ones) are included in training data. The intuition behind this setup is that we want to test our model in the extreme case when target users do not have any out-of-town check-ins, *i.e.*, cold-start problem.

4.4.2 Evaluation Metrics

The goal of the region recommendation problem is to find the region that a given user is most likely to visit. Since from the data there is no indication of the region that the user likes the most, we consider the one that covers the most number of

POIs visited by the user is her groundtruth region. Next, we propose two metrics to evaluate methods for recommending region as follows.

Hit. For each test user, if a groundtruth region is recommended, the method gets a score of 1, otherwise 0. The final score is the average score of all test users¹.

$$Hit = \frac{\# \text{test cases returning groundtruth regions}}{\# \text{all test cases}}$$

Quality. In the previous metric, for a test case, the method gets score only when the recommended region covers all POIs in a groundtruth region, which is quite strict. Hence, we introduce another metric to better evaluate the performance of recommendation methods by considering partial coverage of set of visited POIs. In particular, for each test case t , the quality score of a method is computed as follows,

$$QScore(t) = \frac{\# \text{POIs in the returned region}}{\# \text{POIs in a groundtruth region}}$$

And the final *Quality* is computed by averaging the quality score of all test cases:

$$Quality = \frac{1}{|T|} \sum_t^T QScore(t)$$

4.4.3 Baseline Methods

Since there are no previous methods for region recommendation, we introduce two baselines to compare with our proposed method.

- **Most.dense.** This method always returns the region with most number of POIs. The intuition is that the more POIs a region contains, the higher chance users visit the region.
- **Most.Freq.** This method returns the region which is most frequently visited by users.
- **RegRS/E.** This method returns the region with the highest total visiting probability of its POIs. This method is similar to our proposed method without considering the influence between POIs.

¹Since only one region is returned by methods, Hit metric is equivalent to Precision metric. Recall metric cannot be used because there are *infinite* groundtruth regions.

Figure 4.9: POI distribution of two datasets.

- **RegRS.** This is our proposed method which considers the interaction between POIs.

Both RegRS/E and RegRS require the visiting probabilities of POIs, $P(p_i|u)$, as the input. Those probabilities can be obtained by using any POI recommendation model. We adopt the Non-negative Matrix Factorization method in [128] which performs well on POI recommendation [96]. Let rs_i^u be the recommendation score of a POI p_i w.r.t. a user u achieved by NMF, and then the visiting probability of u to p_i is $P(p_i|u) = \frac{rs_i^u}{\sum_j rs_j^u}$.

4.4.4 Recommendation Results

To test the performance of different region recommendation methods, we use query regions with different sizes $d \times d$, where $d = 0.5km, 1km, 2km$ and $5km$. Note that groundtruth regions of users have at least 2 POIs ($n \geq 2$). We also consider cases when users have groundtruth regions with at least 3 and 4 POIs ($n \geq 3$ and $n \geq 4$), respectively.

4.4.4.1 Performance on Foursquare dataset

Figure 4.10 shows the recommendation results of the four methods on Foursquare dataset. We make the following observations: Firstly, RegRS always outperforms other two baseline methods in all cases, especially when the query size is small. For example, when $d = 0.5km$, RegRS outperforms Most_dense by 400% and 300% in terms of Hit and Quality, respectively. Moreover, the Most_dense method is the worst because it recommends regions to users only based on the distribution of POIs but

(a) Hit

(b) Quality

Figure 4.10: Performance of region recommendation methods on Foursquare dataset.

ignores the user's personalized preference on POIs. On the other hand, **RegRS** always achieves better results than **RegRS/E** (for example, by 93% and 30.7% in terms of Hit and Quality when $d = 0.5km$). This demonstrates the usefulness of considering the influence between POIs in region recommendation. The **Most_Freq** method has promising performance in terms of Quality, and this indicates that POIs that are frequently visited by users tend to locate nearby. When the query size increases, the performance of all the methods improves, and the disparity between **RegRS** and other methods becomes less significant. When $d = 5km$, **RegRS** achieves better results than **Most_dense** by only 29% (Hit) and 18.5% (Quality). This is because when the query size is large, there is a higher chance that a region covers many visited POIs, and thus it is easier to find a groundtruth region even by the simple method.

Figure 4.10 also shows the recommendation accuracy of three methods on users whose the number n of visited POIs in groundtruth regions is at least 3 and 4, respectively. We observe that, compared to the case $n \geq 2$, the performance of all methods

Figure 4.11: Performance of region recommendation methods on Twitter dataset.

increases but **RegRS** improves better than the other two methods. For example, compare to $n \geq 2$, at $n \geq 4$, the Quality of **RegRS** rises up nearly twice, while **RegRS/E** only improves by 41% when $d = 0.5km$. For large regions, the **Most.dense** baseline achieves the best relative improvement but is still the worst method. Overall, **RegRS** gives better recommendation for different values of n .

4.4.4.2 Performance on Twitter dataset

Figures 4.11 presents the recommendation results on Twitter dataset. Similar to Foursquare dataset, **RegRS** still performs better than the other methods. For example, when $d = 0.5km$, **RegRS** beats the **Most.dense** baseline by 43% and 13% in terms of Hit and Quality, respectively. **Most.dense** has better results than **RegRS/E** when query region is small. It is possibly because the distribution of POIs in Twitter dataset is more spread than in Foursquare dataset (Fig. 4.9), and thus more small region candidates are created, most of which are noisy regions. This makes the performance

Table 4.1: Performance comparison between **RegRS** and **Appr-RegRS** in terms of Hit, Quality and running time (in seconds).

	Foursquare			Twitter		
	d = 0.5	d = 1	d = 5	d = 0.5	d = 1	d = 5
RegRS	0.0201	0.0468	0.0980	0.0275	0.0420	0.0912
	0.0666	0.1081	0.2433	0.0922	0.1255	0.2401
	2098 s	3282 s	9847 s	1160 s	1645 s	4216 s
Appr-RegRS ($\epsilon=0.5$)	0.0184	0.0421	0.0923	0.0206	0.0373	0.0904
	0.0649	0.1000	0.2391	0.0905	0.1233	0.2345
	1152 s	2014 s	8539 s	400 s	459 s	571 s
Appr-RegRS ($\epsilon=0.7$)	0.0189	0.0438	0.0947	0.0206	0.0373	0.0909
	0.0650	0.1028	0.2394	0.0901	0.1240	0.2348
	1164 s	2112 s	8891 s	434 s	488 s	607 s

of **RegRS**/E significantly worse. However, **RegRS** does not suffer this problem since it can filter out bad regions in which POIs have little influence on each other.

4.4.5 Approximate Algorithm

In this section, we evaluate the efficiency of **RegRS** and **RegRS** with approximation method (denoted by **Appr-RegRS**). We evaluate **Appr-RegRS** with two approximation ratio values, $\epsilon = 0.5$ and $\epsilon = 0.7$. Table 4.1 shows the results of both **RegRS** and **Appr-RegRS** in terms of Hit, Quality and running time on both datasets. From the table, it is observed that when the query region enlarges, the running time of **RegRS** increases. It is because when the query region size is larger, regions cover more edges and hence more edges are processed. Since the time complexity of **RegRS** is linear to the number of edges, its performance degrades when more edges are considered. On the other hand, **Appr-RegRS** is always faster than **RegRS** for all query sizes. For example, when $d = 0.5km$, the **Appr-RegRS** is nearly twice as fast as **RegRS** on Foursquare dataset. Moreover, the Hit scores achieved by **Appr-RegRS** are always at least 90% of those of **RegRS**. Overall, our proposed approximate method significantly improves the efficiency of **RegRS** while still achieves similar accuracy.

It is interesting that the running time improvement of **Appr-RegRS** over **RegRS** on Twitter dataset is much better than on Foursquare dataset. This might be due to the difference between POI distributions of two datasets (Fig. 4.9), where the POIs in Twitter dataset are distributed more evenly than those in Foursquare dataset. As a result, most of POIs concentrate in few cells in Foursquare dataset, while in Twitter dataset there are many cells with similar number of points. Therefore, more POIs are removed on Twitter dataset when we ignore small-weighted cells, which leads to the better efficiency improvement than on Foursquare dataset.

4.5 Conclusion

In this chapter, we introduce a novel problem of recommending regions for out-of-town users. We subsequently propose a general framework to solve the region recommendation problem. Instead of considering POIs separately, our framework utilizes interaction between POIs to achieve better recommendation accuracy. On the other hand, to overcome the issue of finding the optimal region, we propose an efficient searching algorithm based on the sweeping-based algorithm. We also develop a constant-bounded approximate algorithm to further improve efficiency. Experiments on two real datasets demonstrate the our model significantly outperforms baseline methods in region recommendation.

Chapter 5

Requirement-aware POI recommendation

With the rapid development of location-based social networks (LBSNs), such as Foursquare, POI recommendation systems have become an important service in helping users to find POIs they are interested in. Many POI recommendation models which are based on Collaborative Filtering (CF) techniques, such as user-/item-based CF [17,40] or Matrix Factorization [42,43], have been proposed. The main approach of those models is that the recommendation for a user is made based on other like-minded users' opinions. In other words, those models assume that users with similar interests, *i.e.*, visited many common POIs, are likely to visit the same POIs in the future, which is similar in other recommendation problems. To further improve the recommendation performance, additional information, such as social [40,91] and geographical [40,43,92] factors, has been effectively exploited. However, one important factor that is often ignored by POI recommendation models is user intention. Usually, when submitting recommendation requests, users have clear purposes or requirements, such as going to a cinema to watch a movie or having dinner in a restaurant. Obviously, those requirements are crucial information that help us to make more accurate recommendations as they explicitly reveal users' interests and hence need to be exploited. As a result, in this chapter, we propose a new problem, namely requirement-aware POI recommendation, as follows:

Definition 5.1 *Given a user's requirements, **requirement-aware POI recommendation** aims to find POIs that are both interested by the given user and relevant*

to the requirements.

Here, the user’s requirements are formulated by a query, which is represented by text, *e.g.*, “seafood restaurants” or “places for birthday party”.

From the definition, we can see that the recommended POIs should satisfy two conditions: (i) the recommended POIs should match the user’s preference; and (ii) the recommended POIs should be relevant to the given query (requirements). While the first condition can be achieved by any POI recommendation system, the presence of the query (the second condition) makes the problem more challenging as it requires recommendation systems to be able to model the semantic topics of different POIs and understand user’s intention precisely. For example, let consider a user’s query “looking for a restaurant with low price, cozy atmosphere and great service for a party”. Although there are many requests in the query, such as “looking for a restaurant”, “low price” and “cozy atmosphere”, the user may not consider all the requirements equally. In other words, she regards some requirements more important than others, *e.g.*, she prefers restaurants with “low price” to “cozy atmosphere”. As a result, a recommended POI may not need to meet all the user’s demands but should satisfy important ones. Therefore, a good POI recommendation model should understand what the user wants the most, *i.e.*, her intention, to recommend proper POIs to the user.

Handling and understanding user queries are important tasks in many information retrieval problems such as searching problem. Understanding user queries correctly helps to improve quality of search results and also benefits advertisement systems to better select ads that match user intent. Consequently, there have been many studies on query processing for online advertising and search [129–131]. Most of the existing studies consider that important terms in queries express user intent when performing the search, which is, however, not always true in the case of recommendation. For instance, in the above example, using some weighting schemes like TF-IDF, the term “cozy atmosphere” may be weighted higher (because it appears in fewer queries) than “low prize”, but it does not necessarily indicate that the user prefers the former than the latter. As a result, we need a better method to model user queries.

In recent years, with the fast development of deep learning methods, many advanced models have been proposed, especially for text processing tasks. One of the promising models that has received many focuses from research community is attention networks. Attention networks [52,132] have been proved to be effective in many NLP tasks, ranging from machine translation, language modeling [133] to web search applications [134]. In recommendation systems, using attention networks is still in early stages as we have not seen much work on recommendation applying attention networks. To the best of our knowledge, the attentive model from [135] is the only one so far that employs attention mechanism for recommendation. In this dissertation, we introduce our attention-based model for requirement-aware POI recommendation. Different from [135], where an attention model is proposed to capture a user's interests in different items (item-level attention) and different parts of an item (component-level attention), our model exploits the attention technique to understand the user intent when visiting a POI. As a result, our proposed model can achieve better recommendation accuracy than other methods using simple query processing techniques. Moreover, our model can be easily extended to utilize POI features, such as geographical features.

The organization of this chapter is as follows: In Section 5.1, we introduce our proposed model, which utilizes attention mechanism to analyze user queries and hence better understand user requirements when seeking POIs. Furthermore, we present how to extend our model to incorporate more information to enhance the model performance. Subsequently, we show the experimental results in Section 5.2 before concluding our work in Section 5.3.

5.1 Requirement-aware POI Recommendation Model

In this section, we will introduce our proposed model for the requirement-aware POI recommendation. We firstly describe our first component which is based on the Attention model, then present how to incorporate the geographical features into the model. Finally, we discuss the optimization process to learn the parameters for the model.

5.1.1 Preliminaries

We start with some notations used in this section. Our data contains a set users $\mathcal{U}$, a set of POIs $\mathcal{P}$, a vocabulary $\mathcal{W}$, and a set of check-ins $\mathcal{C}$, each of which is represented by a tuple $\langle u, q, p \rangle$ consisting of user $u \in \mathcal{U}$, POI $p \in \mathcal{P}$, and query q . Here, a query is represented by a set of terms¹, *i.e.*, $q = \{w_1, w_2, \dots, w_n\}$, $w_i \in \mathcal{W}$, expressing user requirements when looking for a POI. Our goal is, for future check-ins, given a user u and her query q , to recommend a POI p that both interests her and satisfies her requirements. Note that, in this dissertation, we focus only on *new* POI recommendation, *i.e.*, recommending POIs that have not been visited by the user before, which is more challenging problem than recommending both visited and un-visited POIs.

5.1.1.1 Latent factor models

Our proposed model is based on latent factor models, which embed users, POIs and query words to a low-dimensional latent space. Formally, we denote user latent vectors as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{|\mathcal{U}|}] \in \mathbb{R}^{\mathcal{K} \times |\mathcal{U}|}$, POI latent vector $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{|\mathcal{P}|}] \in \mathbb{R}^{\mathcal{K} \times |\mathcal{P}|}$ and word latent vector $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{|\mathcal{W}|}] \in \mathbb{R}^{\mathcal{K} \times |\mathcal{W}|}$, where $\mathcal{K} \ll \min(|\mathcal{U}|, |\mathcal{P}|, |\mathcal{W}|)$ is the latent feature dimension. In latent factor models, matrix factorization is usually used as the modeling method, where the check-in preference score R_{ij} between user i and POI j is estimated as:

$$\hat{R}_{ij} = \langle \mathbf{u}_i, \mathbf{p}_j \rangle = \mathbf{u}_i^\top \mathbf{p}_j. \quad (5.1)$$

The objective is to minimize the difference between predicted and observed check-ins preferences using following regularized squared loss:

$$\arg \min_{\mathbf{U}, \mathbf{P}} \sum_{(i,j) \in \mathcal{C}} (R_{ij} - \hat{R}_{ij})^2 + \lambda(\|\mathbf{U}\|^2 + \|\mathbf{P}\|^2), \quad (5.2)$$

where λ is a factor used to control the regularization strength to avoid overfitting. After the model is trained and the optimized user and POI latent vectors are obtained, recommendation for a user i can be done by selecting top POIs with respect

¹In this chapter, “query word” and “query term” have the same meaning, and we use them interchangeably.

to estimated scores $\hat{R}_{ij}$. Note that, the above baseline model does not consider other context information, such as user requirements.

5.1.1.2 Bayesian Personalized Ranking (BPR)

Unlike other recommendation systems like movie or book recommendation, in POI recommendation, user ratings (*i.e.*, check-ins) on POIs are implicit feedback, *i.e.* the check-ins only offer *positive examples* that user likes. Therefore, learning model parameters by fitting the whole check-ins matrix does not bring good performance [43]. To address this issue, ranking-based models [43,118] are usually used as they are better in handling the implicitness of user feedback.

Among them, Bayesian Personalized Ranking (BPR) [118] is a widely-used framework for implicit feedback data, and hence it can be used in POI recommendation. Specifically, the framework assumes that if user i has visited a POI j , the user prefers this POI over un-visited POIs. As a result, the objective of BPR is to rank the POI j higher than un-visited POIs with respect to user i by maximizing the difference between user i 's preference score for POI j and un-visited POIs. Formally, the optimization objective for BPR is given as:

$$\arg \min_{\mathbf{U}, \mathbf{P}} \sum_{(i,j,k) \in \mathcal{D}} -\ln \sigma(\hat{R}_{ij} - \hat{R}_{ik}) + \lambda(\|\mathbf{U}\|^2 + \|\mathbf{P}\|^2), \quad (5.3)$$

where σ is the sigmoid function, λ is the parameter to control strength of regularization (usually an L_2 -norm), and $\mathcal{D}$ is the set of tuples:

$$\mathcal{D} = \{(i, j, k) | j \in \mathcal{R}(i) \wedge k \in \mathcal{P} \setminus \mathcal{R}(i)\}, \quad (5.4)$$

where $\mathcal{R}(i)$ represents the set of POIs that is visited by i -th user. Each element $(i, j, k) \in \mathcal{D}$ expresses the assumption that user i prefers POI j to POI k .

In this work, BPR is used as our basic learning model because of its capability of handling users' implicit check-in feedback.

5.1.2 Proposed Model

In this section, we will introduce our Attention-based Requirement-aware POI recommendation (AttReqPR) model. First, we present the general framework, elaborating

the motivation of the model. Then, we show detailed formulation of our attention model for user queries, and the optimization details of AttReqPR. Lastly, we present the extension of our proposed model to incorporate the geographical influence.

5.1.2.1 General Framework

AttReqPR is a neural network that models user's preference score with respect to a POI given her query (requirements). In particular, the AttReqPR simultaneously models user's preference to her check-in POI and the matching between query and the POI. While the first part can be easily achieved by using any latent factor model, the latter part requires the model to precisely understand user query to find POIs that are relevant to user intent.

As a query is represented as the set of words, *i.e.* $q = \{w_1, w_2, \dots, w_n\}$, we consider each word is a requirement from the user. Hence, we use $\alpha(i, w)$ to denote user i 's preference degree for word w . In other words, $\alpha(i, w)$ expresses how much the user is interested in the requirement (word) w . Moreover, we argue that the user's degree of interests in the word w depends on not only users but also other words in the query. For example, the requirement "low price" from the query "find a restaurant with low price to hang out with friends" may be important to the user because she is going with many friends, but is not much necessary when she wants to "find a restaurant with low price for family". As a result, we now use $\alpha(i, w, q)$ to denote user i 's preference degree for requirement w with respect to the query q .

Objective Function. Based on BPR optimization approach [118], the objective function of AttReqPR is defined as follows:

$$\begin{aligned} \arg \min_{\mathbf{U}, \mathbf{P}, \mathbf{W}, \Theta} \sum_{(i, q, j, k) \in \mathcal{D}} & -\ln \sigma \left\{ \left(\mathbf{u}_i + \sum_{w_t \in q} \alpha(i, w_t, q) \mathbf{w}_{w_t} \right)^\top \mathbf{p}_j - \left(\mathbf{u}_i + \sum_{w_t \in q} \alpha(i, w_t, q) \mathbf{w}_{w_t} \right)^\top \mathbf{p}_k \right\} \\ & + \lambda (\|\mathbf{U}\|^2 + \|\mathbf{P}\|^2 + \|\mathbf{W}\|^2), \end{aligned} \quad (5.5)$$

where $\mathcal{D}$ is the training set consisting of set of tuples (i, q, j, k) , each of which indicates user i has visited POI j instead of POI k given requirements (query) q ; and Θ is the parameters in attention network. $\alpha(i, w, q)$ is the attention module, which estimates the preference degree of user i to word w in query q .

Inference. After the model is trained, we obtain the optimized user, POI and word vectors, *i.e.*, $\mathbf{U}, \mathbf{P}, \mathbf{W}$, as well as parameters of the attention network. Then, given a user i and requirement q , the recommendation is reduced to the problem of ranking all the POIs j un-visited by i based on estimated score $\hat{R}_{i,q,j}$:

$$\hat{R}_{i,q,j} = \left(\mathbf{u}_i + \sum_{w_t \in q} \alpha(i, w_t, q) \mathbf{w}_{w_t} \right)^\top \mathbf{p}_j. \quad (5.6)$$

Relation to other Latent Factor models. Note that if we consider a query is the average of its words, *i.e.*, $\alpha(i, w_t, q) = \frac{1}{N_q}, \forall w_t \in q$, where N_q is the number of words in q , the Eq.(5.6) is rewritten as:

$$\begin{aligned} \hat{R}_{i,q,j} &= \left(\mathbf{u}_i + \sum_{w_t \in q} \frac{1}{N_q} \mathbf{w}_{w_t} \right)^\top \mathbf{p}_j \\ &= \frac{1}{N_q} \sum_{w_t \in q} \left(\mathbf{u}_i + \mathbf{w}_{w_t} \right)^\top \mathbf{p}_j. \end{aligned} \quad (5.7)$$

In other words, $\hat{R}_{i,q,j}$ is the average of $\hat{R}_{i,w_t,j} = \left(\mathbf{u}_i + \mathbf{w}_{w_t} \right)^\top \mathbf{p}_j$. Here, the sub-model $\hat{R}_{i,w_t,j}$ is equivalent to other latent factor models, such as Pairwise Interaction Tensor Factorization (PITF) [136] or Latent Collaborative Retrieval (LCR) [87]. Therefore, our AttReqPR will degenerate into PITF or LCR when we set equal weights for words. However, this contradicts our assumption that users have different focuses on different requirements (represented by words).

Figure 5.1 illustrates two architectures of non-attention (PITF or LCR) model and attention-based model (our proposed AttReqPR model). Both types of models receive set of inputs: user i , positive POI j , negative POI k and set of words $\{w_1, \dots, w_k\}$. Each of input is represented by an 1-hot encoded vector (for example, user i is represented by vector $\mathbf{x}_u$ where $\mathbf{x}_u(i) = 1$ and $\mathbf{x}_u(j) = 0, \forall j \neq i$). Above the input layer is the embedding layer, which is a fully connected layer that projects 1-hot representation to a dense vector. The obtained user/POI/word embeddings are equivalent to the latent vectors for user/POI/word in the context of latent factor model, respectively. In other words, the weight matrices of the embedding layers are identical to latent factor matrices:

$$\mathbf{M}_u = \mathbf{U}, \quad \mathbf{M}_p = \mathbf{P}, \quad \mathbf{M}_w = \mathbf{W}. \quad (5.8)$$

Figure 5.1: The architectures of non-attention and attention-based models.

The difference between non-attention models and AttReqPR is that, after embedding layer, our proposed model has an attention layer. The attention layer, which is a sub-network, takes the user latent vector $\mathbf{u}_i$ and word latent factors $\{w_{w_t}\}$ as an input and outputs an attentive weight $\alpha(i, w_t, q)$ for the word w_t . Thus, the final representation of the query is calculated by the weighted sum $\bar{\mathbf{q}} = \sum_{w_t \in q} \alpha(i, w_t, q) \mathbf{w}_{w_t}$ (green square in Fig. 5.1b). Lastly, in both models, the query latent vector is combined with the basic user latent vector, and the stochastic gradient descent technique can be used to optimize to BPR pairwise learning objective (Eq. (5.5)).

Figure 5.2: Attention layer

5.1.2.2 Attention layer

The goal of the attention layer is to assign words (or requirements) attentive weights that are consistent with user intent, and then apply the weighted sum to construct query representation. Given the basic user latent vector $\mathbf{u}_i$, set of latent vectors $\{\mathbf{w}_w\}$ of words $\{w\}$ in query q and a word $w_k \in q$, we use a three-layer network to compute the attention score $\alpha(i, w_t, q)$ as shown in Fig. 5.2.

Specifically, the first layer aggregates query word latent vectors to form an aggregated latent vector of query words. The reason we need to generate the aggregated word latent vector is, as discussed above, the user's interest on a word may be influenced by other words in the query, hence we need to consider other words in the query when computing weight for this word. Different aggregation operations, such as average pooling or top-k average pooling, can be used here, however, following [137], max-pooling aggregation tends to bring better empirical performance. Max-pooling constructs one vector by taking the maximum value of each dimension of all word vectors, which can be formalized as

$$f_{max}(q) = \begin{bmatrix} \max(\mathbf{w}_1(1), \dots, \mathbf{w}_{|q|}(1)) \\ \max(\mathbf{w}_1(2), \dots, \mathbf{w}_{|q|}(2)) \\ \vdots \\ \max(\mathbf{w}_1(n), \dots, \mathbf{w}_{|q|}(n)) \end{bmatrix}$$

where $\mathbf{w}_l(k)$ denotes the k -th dimension in $\mathbf{w}_l$. In contrast to average pooling, max pooling, which is a nonlinear operation, models interactions among input vectors,

i.e., features/topics from each input vector are compared and only most significant topics will be selected. Then, the second layer builds the hybrid representation by aggregating targeting word w_k 's latent vector and aggregated query latent vector. In this layer, Hadamard product is used so that the resulting hybrid latent vector highlights topics that w_k represents with respect to the query.

$$\mathbf{v}_{hybrid} = \mathbf{w}_{w_k} \odot f_{max}(q). \quad (5.9)$$

Finally, the hybrid latent vector and user latent vector are passed to the last layer to compute the attention score $a(i, w_t, q)$ as follows:

$$a(i, w_t, q) = (\tilde{\mathbf{W}}_U \mathbf{u}_i + \mathbf{b}_U)^\top \mathbf{v}_{hybrid} + \mathbf{c}, \quad (5.10)$$

where the matrix $\tilde{\mathbf{W}}_U$ and bias $\mathbf{b}_U$ are parameters to transform user latent vector so that the layer can model the nonlinearity of interactions between user vector $\mathbf{u}_i$ and hybrid vector $\mathbf{v}_{hybrid}$ (obtained from previous layer), and $\mathbf{c}$ is the bias of the last layer.

The final attention weights are obtained by normalizing the above attention scores using Softmax, which can be interpreted as the attention degree of user i on word w_t :

$$\alpha(i, w_t, q) = \frac{\exp(a(i, w_t, q))}{\sum_{w_m \in q} \exp(a(i, w_m, q))}. \quad (5.11)$$

5.1.2.3 Learning algorithm

We apply stochastic gradient descent algorithm with bootstrap sampling to learn parameters of the model. The steps of the learning process are summarized in Algorithm 4.

Specifically, for each training instance $\langle i, q, j, k \rangle$ drawn from training dataset $\mathcal{D}$, the algorithm first computes the attentive weight for each word $w_t \in q$ w.r.t. user i (line 5) using the attention layer. Subsequently, the attentive query representation is computed as weighted sum of word latent vectors (line 6), which then is combined with user latent vector $\mathbf{u}_i$ to compute the score $\hat{R}_{i,q,j,k}$ (line 7-8). Finally, parameters are updated using back propagation algorithm (line 10 are the gradients of model parameters updated using chain rules).

Algorithm 4: Learning process

Input: Set of observed tuples $\mathcal{D} = \{\langle i, q, j, k \rangle\}$, representing user i , query $q = \{w_1, \dots, w_n\}$, visited POI j and un-visited POI k
Output: Latent matrix $\mathbf{U}, \mathbf{P}, \mathbf{W}$ and parameters in attention model Θ
begin
 1 Initial randomly $\mathbf{U}, \mathbf{P}, \mathbf{V}$ and Θ ;
 2 **repeat**
 3 Draw $\langle i, q, j, k \rangle$ from $\mathcal{D}$;
 4 **foreach** word $w_t \in q$ **do**
 5 Compute $\alpha(i, w_t, q)$ according to Eqs.(5.10) and (5.11);
 6 Compute attentive query representation $\bar{\mathbf{q}} = \sum_{w_t \in q} \alpha(i, w_t, q) \mathbf{w}_{w_t}$;
 7 $\mathbf{v}_{i,q} = \mathbf{u}_i + \bar{\mathbf{q}}$;
 8 $\hat{R}_{i,q,j,k} = \mathbf{v}_{i,q} \mathbf{p}_j - \mathbf{v}_{i,q} \mathbf{p}_k$;
 9 **foreach** parameter θ in $\{\mathbf{U}, \mathbf{P}, \mathbf{W}, \Theta\}$ **do**
 10 Update $\theta \leftarrow \theta + \eta \cdot \left(\frac{\exp(-\hat{R}_{i,q,j,k})}{1 + \exp(-\hat{R}_{i,q,j,k})} \cdot \frac{\partial \hat{R}_{i,q,j,k}}{\partial \theta} + \lambda \cdot \theta \right)$
until convergence;
 11 return $\mathbf{U}, \mathbf{P}, \mathbf{W}$ and Θ .

5.1.3 Incorporating geographical influence

In POI recommendation, geographical influence is usually exploited to enhance recommendation accuracy [40, 43, 47]. In our proposed model, we are capable of utilizing this piece of information by extending AttReqRS to incorporate POIs' geographical features. This section will describe the technique to integrate geographical features into our proposed model.

Similar with previous work, we assume that users usually visit nearby POIs, and hence, POIs located near each other tend to have similar preferences. Following [43, 47], we model each POI's geographical preference by their neighbor POIs. Specifically, each POI j is associated with a $|\mathcal{P}|$ -dimension feature vector $\mathbf{f}_j$, where $|\mathcal{P}|$ is the total number of POIs, and $\mathbf{f}_j(l) = 1$ if l is a neighbor of j , and 0 otherwise. In this work, a POI is considered a neighbor of POI j if it is among k -nearest neighbors $\mathcal{N}_k(j)$ of j . Then, the rating score equation (5.6) is modified to incorporate the feature vector $\mathbf{f}$ into the model as follows:

$$\hat{R}_{i,q,j} = \left(\mathbf{u}_i + \sum_{w_t \in q} \alpha(i, w_t, q) \mathbf{w}_{w_t} \right)^\top \mathbf{p}_j + \beta \mathbf{u}_i^\top \mathbf{G}(\mathbf{f}_j), \quad (5.12)$$

Table 5.1: Yelp dataset

Datasets	NV	AZ
#Users	22,036	18,014
#POIs	9,254	10,957
#Distinct words	31,288	10,285
#Check-ins	556,107	488,442
Avg. Words/Review	54.8	48.9

5.2.1 Dataset

One problem when evaluating the requirement-aware POI recommendation methods is that it is not trivial to know user’s requirements when visiting POIs. Indeed, there is not a check-in dataset that contains user’s explicit queries. To solve this issue, we use the textual content associated with the check-in (such as posts, tweets or reviews) as user requirements.

Specifically, in these experiments, we use the **Yelp** data obtained from Yelp’s Challenge Dataset², which consists of a number of geo-tagged businesses (POIs) and reviews within several cities. In this dataset, we consider each user review on a POI as a check-in of the user in that POI, and the review’s text content as user’s requirements. We first create two datasets by extracting check-in data from two US states, namely Nevada (**NV**) and Arizona (**AZ**), which are two states with most number of check-ins. Then, for each dataset, we remove users with less than 10 visited POIs, and POIs visited by less than 10 users. For reviews, we filter out stop-words and words appearing in less than 5 reviews. The dataset statistics for both regions are summarized in Table 5.1.

Evaluation Protocols. To evaluate the performance of recommendation models, we applied the leave-on-out evaluation, which has been used in previous work [118, 138]. In particular, for each user, we took her latest check-in into testing set, her second most recent check-in into tuning set, and remaining check-ins into training set. As a result, the number of test cases in each dataset equals to the number of users (as shown in the Table 5.1).

²<http://www.yelp.com/dataset>

Evaluation Metrics. We adopt two evaluation metrics, namely Hit Ratio (HIT@K) and Normalized Discounted Cumulative Gain (NDCG@K), where HIT@K measures whether groundtruth POI is among top-K results, and NDCG@K considers the position of groundtruth POI in the returned list. In following experiments, the number K of returned POIs is varied from 5, 10 to 20, where 5 is the default value.

5.2.2 Baseline Methods

For the comparison, besides traditional recommendation methods, UCF and BPR-MF, we use some context-aware recommendation methods, such as PITF or Factorization Machine, to compete with our proposed model.

- **UCF.** This is user-based Collaborative Filtering method, where user-user similarity is computed based on the check-in data.
- **BPR-MF [118].** BPR-MF is a pair-wise ranking latent factor model, which is proposed for handling implicit feedback data. BPR-MF has been proved to be effective for POI recommendation [43] and is also the basic learning method of our proposed model.
- **PITF [136].** PITF is a ranking-based tensor factorization model, where BPR learning method is used. Here, three dimensions of the tensor correspond to users, POIs and words, respectively; and the rating score is computed as Eq.(5.7).
- **FM [31].** Factorization Machine is proposed for recommendation with extra information, such as user/item features or context information. In this model, we use query words as context information (global features).
- **TF-IDF.** This model is similar to our proposed model, except that weight of each word is fixed and equals to its tf-idf score. We use this baseline model to demonstrate the improvement that the attention model can make.
- **AttReqRS.** This is our proposed model for requirement-aware POI recommendation, using attention mechanism to process user queries.

- **AttReqRS+G.** As presented in Section 5.1.3, this is the enhanced model of AttReqRS by considering geographical influence.

In our experiments, we set the number of dimensions of all latent models to 50. For other parameters of models, we use the tuning data to search the optimal values. Especially, for our proposed model AttReqRS, we set the regularization term λ to 0.01, and the geographical contributing weight β in Eq.(5.12) to 0.5.

5.2.3 Experimental Results

5.2.3.1 Results on Requirement-aware POI recommendation

Figure 5.4 shows the performance of different POI recommendation methods on two datasets. From the figure, it is observable that two baseline models UCF and BPR-MF are significantly worse than other methods. This is because those two models do not consider user query when recommending POIs to users, which demonstrates the importance of exploiting this type of context information for POI recommendation. On the other hand, among requirement-aware POI methods, PITF performs worse than other models, especially TF-IDF and our propose model AttReqRS. This is because PITF takes average of word representation as query representation, *i.e.*, considers words in the query equally. In contrast, TF-IDF weights each query word by its tf-idf score, and hence focuses more on meaningful words and alleviates the effect of insignificant words (such as food-unrelated or place-unrelated words). As a result, TF-IDF consistently achieve better performance than PITF by at least 36% and 45% in terms of HIT@5 and NDCG@5, respectively, on NV dataset. Nonetheless, our proposed model is able to improve the recommendation accuracy even further. Specifically, AttReqRS outperforms TF-IDF by 21% and 19% in terms of HIT@K and NDCG@K, respectively, on NV dataset. The corresponding improvements on AZ dataset are 29% and 26%, respectively. This demonstrates attention deep model helps us to better handling user queries, and therefore to understand user intent more precisely than fixed-weighting methods like TF-IDF.

Figure 5.4: Performance of methods in requirement-aware POI recommendation

5.2.3.2 Incorporating geographical influence

As presented in Section 5.1.3, we introduce an extension of our proposed model, namely AttReqRS+G, by considering geographical influence. From Fig. 5.4, we observe that our extended model AttReqRS+G can improve the recommendation performance of AttReqRS noticeably. Especially, AttReqRS+G achieves better accuracy by 11% and 13% (resp. 10% and 14%) in terms of HIT@5 (resp. NDCG@5) in NV and AZ datasets, respectively. This has proved that users tend to visit POIs close to their visited POIs, which is widely accepted by many existing work in POI recommendation.

To further analysis the neighborhood effect of nearby POIs on a POI, we vary the number of neighbor POIs, and record the performance of AttReqRS+G in Fig. 5.5. From the figure, we can see that the performance of AttReqRS+G is quite stable with different numbers of neighbor POIs. AttReqRS+G achieves the peak of performance

Figure 5.5: Performance of AttRegRS+G with different numbers of neighbor POIs.

at 6 neighbor POIs, which is thus set as the number of neighbor POIs for AttRegRS+G in other experiments.

5.2.3.3 Performance over users of different sparsity levels

Our proposed model aims to recommend POIs to users based on both user preferences and user requirements. In many cases, when users do not have many visited POIs, user requirements become essential information to accurately recommend POIs that users are interested in. To investigate the performance of our model for users with different sparsity levels, we compare models in the experiment for users with different numbers of visited POIs, and show the results on NV dataset in Fig. 5.6 (we observe similar results on AZ dataset, and hence do not show for saving space). For this experiment, instead of re-training the model with different set of users, we train the models for all users and show the results for different groups of users with respect to the number of POIs that users have visited. From Fig. 5.6, we observe that:

- Our models, AttReqRS and AttReqRS+G, consistently outperform other baseline methods for all user groups with different numbers of visited POIs. It demonstrates that our models are robust and flexible on different datasets.
- We also observe that our models perform much better than other methods for users with few visited POIs. This demonstrates that our proposed models handle better user queries, and therefore are able to recommend precisely to users even they do not have many history check-in data.

Figure 5.6: Performance of recommendation methods on users with different numbers of visited POIs.

5.2.3.4 Performance over queries of different numbers of words

Besides users' number of visited POIs, we also evaluate all the methods for queries with different numbers of words, and plot the results, (HIT@5 in both datasets) in the Fig. 5.7. From the figure, we can see that the performance of all methods are very poor when queries have only few words (*e.g.*, fewer than 10 words per query). This maybe because short queries do not express all the user requirements when looking for a POI. Hence, for short queries, it is difficult for methods to understand user intent properly. When queries are longer, the performance of all methods increase, but our proposed methods, AttReqRS and AttReqRS+G, improves much better. For example, the HIT@5 score of AttReqRS in NV dataset is improved by 2.8 times when the query length is increased from 0-10 to 80-89 words, while this improvement rate from TF-IDF is just 1.8. In the AZ dataset, the similar trend can be seen where our proposed methods outperform other baselines for all query group with different length. One noteworthy observation is that while FM method has significant performance advancement when increasing query length, TF-IDF baseline does not gain much improvement and even loses its accuracy when the number of words per query is larger than 50 in both datasets. This means that the more complicated methods like FM can handle user queries better than fixed-weighting methods as TF-IDF. Overall, our proposed models can achieve the best accuracy for all queries of different length among all the methods considered in this experiment.

(a) HIT@5 - NV

(b) HIT@5 - AZ

Figure 5.7: Performance of recommendation methods on queries with different numbers of words.

5.3 Conclusion

In this chapter, we presented our proposed model for the requirement-aware POI recommendation problem. Our model, AttReqRS, is a latent factor-based deep learning model. In order to precisely understand user intent when she is looking for POIs, AttReqRS introduces an attention layer, which aims to emphasize important requirements in the user query with respect to user preference. Using attention mechanism helps us to handle user queries more flexibly and properly. Moreover, AttReqRS can be easily extended to incorporate additional information such as geographical influence. From comprehensive experiments on two real-world datasets, we have demonstrated that our proposed model consistently outperforms other state-of-the-art methods in POI recommendation. In the future, we plan to extend our model by considering other types of information such as POI content information (category or description), which might be important when the user requirements are known.

Chapter 6

Conclusions and Future Work

In this chapter, we first summary the chapters to highlight main contributions of this thesis. Then, we point out some promising directions of future work.

6.1 Conclusions

Nowadays, due to the rapid development of online Web services, such as social networks and blogs, recommendation systems play a more and more important role on helping users to find what they are interested in. Different from traditional recommendation systems, which only consider the user-item interaction, context-aware recommendation systems take advantage of context information to gain better recommendation accuracy. Many context-aware recommendation systems have been proposed for different problems and using various types of context information. In this thesis, we introduce three context-aware recommendation models, which aim to solve different recommendation problems, namely recommendation in heterogeneous networks, out-of-town region recommendation and requirement-aware recommendation, respectively.

In Chapter 3, we introduce our general model for recommendation in heterogeneous networks. We first construct a heterogeneous graph to represent different types of entities and their relations in the heterogeneous network. Then, a proximity estimation algorithm, namely multivariate Markov chain (MMC), is employed in the heterogeneous graph to make recommendation w.r.t query. To avoid the issue of manual parameter assignment on MMC, an optimization scheme to learn transition

parameters is proposed for our model. To the best of our knowledge, this is the first work proposing a ranking-based method for learning parameters in an MMC-based model. The learned parameters not only help our model to outperform other baseline methods, but also give us better understandings of roles of different entity types in each recommendation problem.

In Chapter 4, we introduce a novel problem, which aims to recommend regions to users when they travel out of their hometowns. Subsequently, a general framework is proposed to solve the out-of-town region recommendation. To gain better recommendation performance, our framework exploits the interaction between POIs in the region, rather than treating them separately. Experiments demonstrate that considering the cross-dependence between POIs is beneficial when making the region recommendation. On the other hand, since our model encounters the efficiency issue of finding the optimal region when the number of POIs is large, we propose an efficient searching algorithm based on the sweeping-based algorithm. Moreover, we also develop an approximate algorithm to further improve the efficiency. It is proved that both our exact and approximate algorithms work effectively on large datasets and query regions, and approximate algorithm scales much better when the size of dataset and query regions increases.

In Chapter 5, we present our proposed model for the requirement-aware POI recommendation problem. Our model is a deep learning model which utilizes an attention layer for analyzing user queries and hence understanding user intent properly when seeking a POI. Through experiments, it has been showed that our proposed model outperforms other state-of-the-art methods significantly, which proves the efficiency of the attention mechanism in our model in understanding the user requirements when recommending POIs to users. Moreover, the model can be easily extended by incorporating user or POI information, such as the geographical influence for capturing the spatial context. This extension further enhances the performance of our model substantially.

6.2 Future Work

There are many potential directions for future work. Here, we present few of them:

- In the real-life scenario, two entities may have different type of interactions. For example, in Yelp, the popular restaurant review website, a user can make a check-in, write a review or make a rating to a restaurant. Different interactions may bring different meanings. Our proposed model in Chapter 3 is not able to handle this situation properly. Some existing work have proposed ranking methods on this special type of networks [139, 140], however, since they are based on random walk algorithms, they cannot explicitly control the roles the entity types as our model.
- We plan to explore more types of context information that is useful for recommendation. It could be the user’s immediate feedback on questions [141] or recommended items [83] from recommendation systems, which allows recommendation systems to learn more about the user’s preference and bring better recommendation results.
- One of the most challenging issue that our models encounter is the scalability and efficiency problem, especially in training process. Although some effective methods have been proposed to overcome this problem, that is not enough, considering the rapid rate of new coming data in many systems like Twitter and Meetup. One possible solution is to derive some incremental algorithms so that models can be updated without being retrained when new data comes.

References

- [1] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, “Recommending and evaluating choices in a virtual community of use,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1995, pp. 194–201.
- [2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: An open architecture for collaborative filtering of netnews,” in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. ACM, 1994, pp. 175–186.
- [3] U. Shardanand and P. Maes, “Social information filtering: Algorithms for automating “word of mouth”,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1995, pp. 210–217.
- [4] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han, “Event-based social networks: Linking the online and offline social worlds,” in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 1032–1040.
- [5] A. Zimmermann, A. Lorenz, and R. Oppermann, “An operational definition of context,” in *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context*. Springer, 2007, pp. 558–571.
- [6] A. K. Dey, G. D. Abowd, and D. Salber, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” *Human-computer interaction*, vol. 16, no. 2, pp. 97–166, 2001.
- [7] S. El Helou, C. Salzmann, and D. Gillet, “The 3a personalized, contextual and relation-based recommender system,” *Journal of Universal Computer Science*, vol. 16, no. 16, pp. 2179–2195, 2010.

- [8] A. Schmidt, “Impact of context-awareness on the architecture of learning support systems,” *Architecture Solutions for E-learning Systems*, pp. 306–319, 2008.
- [9] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. IEEE Computer Society, 1994, pp. 85–90.
- [10] G. Chen, D. Kotz *et al.*, “A survey of context-aware mobile computing research,” Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, Tech. Rep., 2000.
- [11] A. Schmidt, M. Beigl, and H.-W. Gellersen, “There is more to context than location,” *Computers and Graphics*, vol. 23, no. 6, pp. 893–901, 1999.
- [12] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, “Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering,” in *Proceedings of the Fourth ACM Conference on Recommender Systems*. ACM, 2010, pp. 79–86.
- [13] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, “Fast context-aware recommendations with factorization machines,” in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2011, pp. 635–644.
- [14] E. Zhong, W. Fan, and Q. Yang, “Contextual collaborative filtering via hierarchical matrix factorization,” in *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012, pp. 744–755.
- [15] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, “Tfmap: Optimizing map for top-n context-aware recommendation,” in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2012, pp. 155–164.
- [16] X. Ning and G. Karypis, “Sparse linear methods with side information for top-n recommendations,” in *Proceedings of the Sixth ACM Conference on Recommender Systems*. ACM, 2012, pp. 155–162.
- [17] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel, “Lars: A location-aware recommender system,” in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*. IEEE Computer Society, 2012.

- [18] X. Liu and K. Aberer, “Soco: A social network aided context-aware recommender system,” in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 781–802.
- [19] M. Jamali and L. Lakshmanan, “Heteromf: Recommendation in heterogeneous information networks using context dependent factor models,” in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 643–654.
- [20] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,” *ACM Computing Surveys*, pp. 3:1–3:45, 2014.
- [21] T. Zhu, P. Harrington, J. Li, and L. Tang, “Bundle recommendation in e-commerce,” in *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2014, pp. 657–666.
- [22] J. McAuley, R. Pandey, and J. Leskovec, “Inferring networks of substitutable and complementary products,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 785–794.
- [23] D.-C. Nie, Y. Fu, J.-L. Zhou, Z. Liu, Z.-K. Zhang, and C. Liu, “A personalized recommendation algorithm via biased random walk,” in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2014, pp. 292–296.
- [24] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, “Personalized entity recommendation: A heterogeneous information network approach,” in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. ACM, 2014, pp. 283–292.
- [25] I. Konstas, V. Stathopoulos, and J. M. Jose, “On social networks and collaborative recommendation,” in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2009, pp. 195–202.
- [26] W. Zhang, J. Wang, and W. Feng, “Combining latent factor model with location features for event-based group recommendation,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2013, pp. 910–918.

- [27] S. Lee, S.-i. Song, M. Kahng, D. Lee, and S.-g. Lee, “Random walk based entity ranking on graph for multidimensional recommendation,” in *Proceedings of the Fifth ACM Conference on Recommender Systems*. ACM, 2011, pp. 93–100.
- [28] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, “Temporal recommendation on graphs via long- and short-term preference fusion,” in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2010, pp. 723–732.
- [29] M. Jiang, P. Cui, X. Chen, F. Wang, W. Zhu, and S. Yang, “Social recommendation with cross-domain transferable knowledge,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3084–3097, 2015.
- [30] A. P. Singh and G. J. Gordon, “Relational learning via collective matrix factorization,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 650–658.
- [31] S. Rendle, “Factorization machines,” in *Proceedings of the 2010 IEEE International Conference on Data Mining*. IEEE Computer Society, 2010, pp. 995–1000.
- [32] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, “Who, where, when and what: Discover spatio-temporal topics for twitter users,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2013, pp. 605–613.
- [33] Y. Kim, Y. Park, and K. Shim, “Digtobi: A recommendation system for digg articles using probabilistic modeling,” in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 691–702.
- [34] K. Zhao, G. Cong, Q. Yuan, and K. Q. Zhu, “Sar: a sentiment-aspect-region model for user preference analysis in geo-tagged reviews,” in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 675–686.
- [35] X. Jiang, X. Sun, and H. Zhuge, “Towards an effective and unbiased ranking of scientific literature through mutual reinforcement,” in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, 2012, pp. 714–723.

- [36] H. Sayyadi and L. Getoor, “Futurerank: Ranking scientific articles by predicting their future pagerank,” in *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, 2009, pp. 533–544.
- [37] J.-L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains,” *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.
- [38] W. Ching, S. Zhang, and M. Ng, “On multi-dimensional markov chain models,” *Pacific Journal of Optimization*, vol. 3, no. 2, 2007.
- [39] J. Nicolau, “A new model for multivariate markov chains,” *Scandinavian Journal of Statistics*, vol. 41, no. 4, pp. 1124–1135, 2014.
- [40] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, “Exploiting geographical influence for collaborative point-of-interest recommendation,” in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2011, pp. 325–334.
- [41] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen, “Lcars: A location-content-aware recommender system,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2013, pp. 221–229.
- [42] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, “Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 831–840.
- [43] X. Li, G. Cong, X.-L. Li, T.-A. N. Pham, and S. Krishnaswamy, “Rank-geofm: A ranking based geographical factorization method for point of interest recommendation,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 433–442.
- [44] J. Bao, Y. Zheng, and M. F. Mokbel, “Location-based and preference-aware recommendation using sparse geo-social networking data,” in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 199–208.

- [45] W. Wang, H. Yin, L. Chen, Y. Sun, S. Sadiq, and X. Zhou, “Geo-sage: A geographical sparse additive generative model for spatial item recommendation,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1255–1264.
- [46] X. Cao, G. Cong, and C. S. Jensen, “Retrieving top-k prestige-based relevant spatial web objects,” *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 373–384, 2010.
- [47] Y. Liu, W. Wei, A. Sun, and C. Miao, “Exploiting geographical neighborhood characteristics for location recommendation,” in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, 2014, pp. 739–748.
- [48] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th International Conference on World Wide Web*. ACM, 2017, pp. 173–182.
- [49] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations,” in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.
- [50] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, “A hybrid collaborative filtering model with deep structure for recommender systems,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 1309–1315.
- [51] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “Deepfm: A factorization-machine based neural network for ctr prediction,” *arXiv preprint arXiv:1703.04247*, 2017.
- [52] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [53] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, and T.-S. Chua, “Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning,” *arXiv preprint arXiv:1611.05594*, 2016.
- [54] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

- [55] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, “Image captioning with semantic attention,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4651–4659.
- [56] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, “Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention,” in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 335–344.
- [57] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th International Conference on World Wide Web*. ACM, 2001, pp. 285–295.
- [58] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” in *Advances in Neural Information Processing Systems*, vol. 20, 2011, pp. 1–8.
- [59] Y. Koren, R. Bell, C. Volinsky *et al.*, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
- [60] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [61] T. Hofmann, “Latent semantic models for collaborative filtering,” *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.
- [62] R. Krestel, P. Fankhauser, and W. Nejdl, “Latent dirichlet allocation for tag recommendation,” in *Proceedings of the Third ACM Conference on Recommender Systems*. ACM, 2009, pp. 61–68.
- [63] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in *Recommender Systems Handbook*. Springer, 2011, pp. 73–105.
- [64] A. Ansari, S. Essegaiier, and R. Kohli, “Internet recommendation systems,” *Journal of Marketing research*, vol. 37, no. 3, pp. 363–375, 2000.
- [65] P. Melville, R. J. Mooney, and R. Nagarajan, “Content-boosted collaborative filtering for improved recommendations,” in *Eighteenth National Conference on Artificial Intelligence*, 2002, pp. 187–192.

- [66] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, “Combining content-based and collaborative filters in an online newspaper,” in *Proceedings of ACM SIGIR workshop on recommender systems*, vol. 60, 1999.
- [67] M. Pazzani and D. Billsus, “Learning and revising user profiles: The identification of interesting web sites,” *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997.
- [68] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.
- [69] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, “Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach,” in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 473–480.
- [70] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, “Probabilistic memory-based collaborative filtering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56–69, 2004.
- [71] H. Wang, M. Terrovitis, and N. Mamoulis, “Location recommendation in location-based social networks using user check-in data,” in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 374–383.
- [72] J. Zhang, J. Tang, B. Liang, Z. Yang, S. Wang, J. Zuo, and J. Li, “Recommendation over a heterogeneous social network,” in *2008 The Ninth International Conference on Web-Age Information Management*. IEEE, 2008, pp. 309–316.
- [73] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han, “Recommendation in heterogeneous information networks with implicit user feedback,” in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 347–350.
- [74] D. Agarwal and B.-C. Chen, “Regression-based latent factor models,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 19–28.
- [75] C. Luo, W. Pang, Z. Wang, and C. Lin, “Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations,” in *2014 IEEE International Conference on Data Mining*. IEEE, 2014, pp. 917–922.

- [76] A. Zimdars, D. M. Chickering, and C. Meek, “Using temporal data for making recommendations,” in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 580–588.
- [77] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, “Using sequential and non-sequential patterns in predictive web usage mining tasks,” in *Proceedings of the 2002 IEEE International Conference on Data Mining*. IEEE, 2002, pp. 669–672.
- [78] G. Shani, D. Heckerman, and R. I. Brafman, “An mdp-based recommender system,” *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1265–1295, 2005.
- [79] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, “Factorizing personalized markov chains for next-basket recommendation,” in *Proceedings of the 19th International Conference on World Wide Web*. ACM, 2010, pp. 811–820.
- [80] C. Cheng, H. Yang, M. R. Lyu, and I. King, “Where you like to go next: Successive point-of-interest recommendation,” in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, vol. 13, 2013, pp. 2605–2611.
- [81] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, “Personalized ranking metric embedding for next new poi recommendation,” in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 2069–2075.
- [82] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, “Playlist prediction via metric embedding,” in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 714–722.
- [83] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng, “Collaborative competitive filtering: Learning recommender using context of user choice,” in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2011, pp. 295–304.
- [84] Z. Sun, J. Yang, J. Zhang, and A. Bozzon, “Exploiting both vertical and horizontal dimensions of feature hierarchy for effective recommendation,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 189–195.

- [85] J. Yang, Z. Sun, A. Bozzon, and J. Zhang, “Learning hierarchical feature influence for recommendation by recursive regularization.” in *Proceedings of the 10th ACM conference on Recommender systems*, 2016, pp. 51–58.
- [86] Q. Yuan, G. Cong, K. Zhao, Z. Ma, and A. Sun, “Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users,” *ACM Transactions on Information Systems (TOIS)*, vol. 33, no. 1, p. 2, 2015.
- [87] J. Weston, C. Wang, R. Weiss, and A. Berenzweig, “Latent collaborative retrieval,” *arXiv preprint arXiv:1206.4603*, 2012.
- [88] K.-J. Hsiao, A. Kulesza, and A. O. Hero, “Social collaborative retrieval,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 680–689, 2014.
- [89] W. Zhang and J. Wang, “Location and time aware social collaborative retrieval for new successive point-of-interest recommendation,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1221–1230.
- [90] J.-D. Zhang and C.-Y. Chow, “Spatiotemporal sequential influence modeling for location recommendations: A gravity-based approach,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 7, no. 1, p. 11, 2015.
- [91] J. D. Zhang and C. Y. Chow, “igslr: Personalized geo-social location recommendation: A kernel density estimation approach,” in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 334–343.
- [92] C. Cheng, H. Yang, I. King, and M. R. Lyu, “Fused matrix factorization with geographical and social influence in location-based social networks,” in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012, pp. 17–23.
- [93] J.-D. Zhang, C.-Y. Chow, and Y. Li, “Lore: exploiting sequential influence for location recommendations,” in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2014, pp. 103–112.

- [94] J.-D. Zhang and C.-Y. Chow, “Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 443–452.
- [95] B. Hu and M. Ester, “Spatial topic modeling in online social media for location recommendation,” in *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 2013, pp. 25–32.
- [96] B. Liu, Y. Fu, Z. Yao, and H. Xiong, “Learning geographical preferences for point-of-interest recommendation,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2013, pp. 1043–1051.
- [97] B. Liu, H. Xiong, S. Papadimitriou, Y. Fu, and Z. Yao, “A general geographical probabilistic factor model for point of interest recommendation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1167–1179, 2015.
- [98] M. Ye, P. Yin, and W.-C. Lee, “Location recommendation for location-based social networks,” in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2010, pp. 458–461.
- [99] J.-D. Zhang, C.-Y. Chow, and Y. Zheng, “Orec: An opinion-based point-of-interest recommendation framework,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1641–1650.
- [100] H. Li, R. Hong, S. Zhu, and Y. Ge, “Point-of-interest recommender systems: A separate-space perspective,” in *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2015, pp. 231–240.
- [101] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, “A random walk around the city: New venue recommendation in location-based social networks,” in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. Ieee, 2012, pp. 144–153.

- [102] J. J.-C. Ying, W.-N. Kuo, V. S. Tseng, and E. H.-C. Lu, “Mining user check-in behavior with a random walk for urban point-of-interest recommendations,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 40, 2014.
- [103] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 263–272.
- [104] H. Gao, J. Tang, X. Hu, and H. Liu, “Content-aware point of interest recommendation on location-based social networks,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [105] D. Lian, Y. Ge, F. Zhang, N. J. Yuan, X. Xie, T. Zhou, and Y. Rui, “Content-aware collaborative filtering for location recommendation based on human mobility data,” in *2015 IEEE International Conference on Data Mining*. IEEE, 2015, pp. 261–270.
- [106] P. Bhargava, T. Phan, J. Zhou, and J. Lee, “Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data,” in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 130–140.
- [107] X. Liu, Y. Liu, K. Aberer, and C. Miao, “Personalized point-of-interest recommendation by mining users’ preference transition,” in *Proceedings of the 22nd ACM international conference on Information and Knowledge Management*. ACM, 2013, pp. 733–738.
- [108] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King, “Stellar: Spatial-temporal latent ranking for successive point-of-interest recommendation,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [109] J. He, X. Li, L. Liao, D. Song, and W. K. Cheung, “Inferring a personalized next point-of-interest recommendation model with latent behavior patterns,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [110] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, “Time-aware point-of-interest recommendation,” in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2013, pp. 363–372.

- [111] Q. Yuan, G. Cong, and A. Sun, “Graph-based point-of-interest recommendation with geographical and temporal influences,” in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, 2014, pp. 659–668.
- [112] R. Deveaud, M.-D. Albakour, C. Macdonald, I. Ounis *et al.*, “Experiments with a venue-centric model for personalised and time-aware venue suggestion,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 53–62.
- [113] G. Ference, M. Ye, and W.-C. Lee, “Location recommendation for out-of-town users in location-based social networks,” in *Proceedings of the 22nd ACM international conference on Information and Knowledge Management*. ACM, 2013, pp. 721–726.
- [114] T. Kurashima, T. Iwata, T. Hoshide, N. Takaya, and K. Fujimura, “Geo topic model: Joint modeling of user’s activity area and interests for location recommendation,” in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*. ACM, 2013, pp. 375–384.
- [115] H. Yin, X. Zhou, Y. Shao, H. Wang, and S. Sadiq, “Joint modeling of user check-in behaviors for point-of-interest recommendation,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1631–1640.
- [116] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang, “A general graph-based model for recommendation in event-based social networks,” in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 567–578.
- [117] W.-K. Ching and M. K. Ng, “Multivariate markov chains,” *Markov Chains: Models, Algorithms and Applications*, pp. 141–169, 2006.
- [118] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 452–461.
- [119] I. Cantador, P. Brusilovsky, and T. Kuflik, “2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011),” in *Proceedings of the Fifth ACM Conference on Recommender Systems*. ACM, 2011.

- [120] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, “The yahoo! music dataset and kdd-cup ’11,” in *Proceedings of KDD Cup 2011 competition, San Diego, CA, USA, 2011*, 2012, pp. 8–18.
- [121] X. Liu, Y. Tian, M. Ye, and W.-C. Lee, “Exploring personal impact for group recommendation,” in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, 2012, pp. 674–683.
- [122] W. Feng and J. Wang, “Incorporating heterogeneous information for personalized tag recommendation in social tagging systems,” in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 1276–1284.
- [123] H. Imai and T. Asano, “Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane,” *Journal of algorithms*, vol. 4, no. 4, pp. 310–323, 1983.
- [124] S. C. Nandy and B. B. Bhattacharya, “A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids,” *Computers and Mathematics with Applications*, vol. 29, no. 8, pp. 45–61, 1995.
- [125] H. Gao, J. Tang, and H. Liu, “gscorr: Modeling geo-social correlations for new check-ins on location-based social networks,” in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, 2012, pp. 1582–1586.
- [126] J. Cranshaw, R. Schwartz, J. I. Hong, and N. Sadeh, “The livelihoods project: Utilizing social media to understand the dynamics of a city,” in *Sixth International AAAI Conference on Weblogs and Social Media*, 2012, p. 58.
- [127] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui, “Exploring millions of footprints in location sharing services,” *Fifth International AAAI Conference on Weblogs and Social Media*, vol. 2011, pp. 81–88, 2011.
- [128] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems*, 2001, pp. 556–562.
- [129] R. Jones and D. C. Fain, “Query word deletion prediction,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 435–436.

- [130] P. Liu, J. Azimi, and R. Zhang, “Contextual query intent extraction for paid search selection,” in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 71–72.
- [131] K. T. Maxwell and W. B. Croft, “Compact query term selection using topically related text,” in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 583–592.
- [132] W. L. L. C.-C. Yulia, T. S. Amir, R. F. A. C. D. Alan, and W. B. I. Trancoso, “Not all contexts are created equal: Better word representations with variable attention,” *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1367–1372, 2015.
- [133] Z. Yin, K.-h. Chang, and R. Zhang, “Deepprobe: Information directed sequence understanding and chatbot design via recurrent neural networks,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 2131–2139.
- [134] S. Zhai, K.-h. Chang, R. Zhang, and Z. M. Zhang, “Deepintent: Learning attentions for online advertising with recurrent neural networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1295–1304.
- [135] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, “A entire collaborative filtering: Multimedia recommendation with item-and component-level a ention,” in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017.
- [136] S. Rendle and L. Schmidt-Thieme, “Pairwise interaction tensor factorization for personalized tag recommendation,” in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 81–90.
- [137] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, “Learning hierarchical representation model for nextbasket recommendation,” in *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 403–412.

- [138] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, “Fast matrix factorization for online recommendation with implicit feedback,” in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 549–558.
- [139] D. Zhou, J. Huang, and B. Schölkopf, “Learning with hypergraphs: Clustering, classification, and embedding,” in *Advances in Neural Information Processing Systems*, 2006, pp. 1601–1608.
- [140] A. Bellaachia and M. Al-Dhelaan, “Random walks in hypergraph,” in *Proceedings of the 2013 International Conference on Applied Mathematics and Computational Method*, 2013, pp. 187–194.
- [141] K. H. Konstantina Christakopoulou, Filip Radlinski, “Towards conversational recommender systems,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 815–824.

Author's Publications

- [1] **Tuan-Anh Nguyen Pham**, Xutao Li, Gao Cong, Zhenjie Zhang: A general graph-based model for recommendation in event-based social networks. Proceedings of 31st International Conference on Data Engineering (**ICDE 2015**), 567-578, Seoul, Korea, April 13-17, 2015.
- [2] Xutao Li, Gao Cong, Xiao-Li Li, **Tuan-Anh Nguyen Pham**, Shonali Krishnaswamy: Rank-GeoFM: a ranking based geographical factorization method for point of interest recommendation. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (**SIGIR 2015**), 433-442, Santiago, Chile, August 9-13, 2015.
- [3] Xutao Li, **Tuan-Anh Nguyen Pham**, Gao Cong, Quan Yuan, Xiao-Li Li, Shonali Krishnaswamy: Where you Instagram? Associating Your Instagram Photos with Points of Interest. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (**CIKM 2015**), 1231-1240, Melbourne, Australia, October 19-23, 2015.
- [4] **Tuan-Anh Nguyen Pham**, Xutao Li, Gao Cong, Zhenjie Zhang: A General Recommendation Model for Heterogeneous Networks. IEEE Transactions on Knowledge & Data Engineering (**TKDE**), vol.PP, no.99, pp.1-1, doi: 10.1109/TKDE.2016.2601091.
- [5] **Tuan-Anh Nguyen Pham**, Xutao Li, Gao Cong: A General Model for Out-of-town Region Recommendation. Proceedings of the 26th International Conference on World Wide Web (**WWW 2017**), 401-410, Perth, Australia, April 3-7, 2017.

- [6] Yiding Liu, **Tuan-Anh Nguyen Pham**, Gao Cong, Quan Yuan: An Experimental Evaluation of Point-of-interest Recommendation in Location-based Social Networks. Proceedings of the VLDB Endowment (**VLDB 2017**), 1010-1021, Munich, Germany, August 28 - September 1, 2017.