COMPOSITE SOCIAL NETWORKS ANALYSIS

by

ERHENG ZHONG

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

January 2014, Hong Kong

Copyright © by Erheng Zhong 2014
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Erheng Zhong
ERHENG ZHONG
COMPOSITE SOCIAL NETWORKS ANALYSIS

by

ERHENG ZHONG

This is to certify that I have examined the above Ph.D. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

[Signature]
PROF. QIANG YANG, THESIS SUPERVISOR

[Signature]
PROF. SIU-WING CHENG, ACTING HEAD OF DEPARTMENT

Department of Computer Science and Engineering
23 January 2014
ACKNOWLEDGMENTS

First and foremost, I would like to express my thanks to my supervisor Prof. Qiang Yang sincerely for his valuable advice, in-depth guidance, support and freedom in my research during my PhD study. He taught me how to identify, solve and present research problems, and more importantly, how to establish the right attitude to research.

I also thank Dr. Wei Fan, who guided me into the world of data mining. As a mentor and a friend, his generous time and devotion on supervision and guidance turned me from an immature undergraduate student into a qualified researcher.

I want to thank many professors in HKUST, including Prof. Weichuan Yu, Prof. Dit-Yan Yeung, Prof. Nevin Zhang, Prof. Raymond Wong, Prof. Xiaoquan Zhang and Prof. Sunghun Kim for spending their valuable time on reading my thesis and attending my thesis defense, thesis proposal defense and Ph.D. qualification exam. Their constructive comments are greatly helpful for revising and improving this thesis work.

The members of the research group have been a great part of my experience. They are Bin Cao, Weizhu Chen, Liya Ji, Bo Liu, Nathan Nan Liu, Zhongqi Lu, Kaixiang Mo, Sinno Jialin Pan, Weike Pan, Si Shen, Ben Tan, Ying Wei, Bin Wu, Evan Wei Xiang, Ruiming Xie, Qian Xu, Lili Zhao, Vincent Wenchen Zheng, and Yin Zhu.

I would also like to thank many members of the data mining group in Tencent, including but not limited to Dr. Yong Li, Dr. Duoquan Li, Dr. Yading Yue, Mr. Lei Xiao, and Mr. Xiaoping Lai. They provided me many opportunities to solve real-world data mining problems.

Finally and above all, I want to give my deepest gratitude to my parents, Yangfeng Zhong and Xuefang Lin, and my wife, Xiangmin Zhong. Their love and support make all these come true! This thesis is dedicated to them.
TABLE OF CONTENTS

Title Page i
Authorization Page ii
Signature Page iii
Acknowledgments iv
Table of Contents v
List of Figures viii
List of Tables x
Abstract xi

Chapter 1 Introduction 1
1.1 Motivation and Overview 2
   1.1.1 What is Composite Social Network? 3
   1.1.2 Why is Composite Social Network Analysis a New Problem? 8
1.2 Research Objectives and Main Contributions 8
1.3 Executive Summary of Key Discoveries 11
1.4 Thesis Outline 11

Chapter 2 Survey on Composite Network Analysis 13
2.1 Social Network Analysis 13
   2.1.1 Link Prediction 13
   2.1.2 Community Discovery 15
   2.1.3 Social Influence Analysis 18
2.2 Multi-relational Network Analysis 20
   2.2.1 Link Prediction in Multi-relational Social Networks 21
   2.2.2 Community Discovery using Multiple Relations 24
   2.2.3 Other Applications 26
2.3 Cross-network Transfer Learning 29
## LIST OF FIGURES

1.1 Composite Social Network Illustration. Each user can join multiple social networks and different social networks are nested together by the shared users. 3

1.2 Social Media Illustration. One user can join multiple social media websites and knowledge in the social media are encoded into different representations. 4

1.3 Illustration of Degree Distributions 6

1.4 Thesis Outline 12

2.1 Illustration of Link Prediction 14

2.2 Illustration of Community Discovery 16

2.3 Illustration of Social Influence 18

2.4 Illustration of the 3-node Substructure 21

3.1 Illustration of Simple Combination 42

3.2 The Graphical Representation of ComSoc-MMSB 43

3.3 Illustration of Composite Network. Different colors represent different communities. The size of each node presents each user’s membership of the community, e.g., larger size indicates higher probability of the user belonging to this community. 45

3.4 Performances on Long-tail Users of Relational Networks (ComSoc-MMSB) 50

3.5 Performances on Long-tail Users of Social Networks (ComSoc-MMSB) 50

3.6 Performances on Long-tail Users of Social Medias (ComSoc-MMSB) 51

3.7 Performances on Long-tail Users of Programmer Networks (ComSoc-MMSB) 51

3.8 Performance Analysis on Douban collection (ComSoc-MMSB) 53

3.9 Parameter Analysis on Douban collection (ComSoc-MMSB) 53

3.10 Efficiency Analysis on Douban collection (ComSoc-MMSB) 54

4.1 Illustration of Dynamic Composite Network. (a)~(b) and (c)~(d) indicate the different numbers of communities and community memberships over time, where color indicates community and size presents community membership. (a)~(c) and (b)~(d) indicate the differences of community structures across networks. 58

4.2 Illustration of Community Dynamics. Different community relation strength can change over time. 59

4.3 Illustration of Dynamic and Composite Properties. (a) presents the ratio of common links across networks, where the ratio is much higher than that between two random graphs, implying individual networks influence each other. (b) illustrates the correlation between the number of degrees and the number of new links generated by the users. 60

4.4 The Graphical Representation of ComSoc-IT. Blue nodes indicate observed variables, greens present hyper priors, reds mean variables in last time stamp and grays indicate latent variables need to be inferred. 62
5.1 Synthetic example on how to exploit Composite Social Network adaptively. (a) the complete user-item interaction matrix - most items are empty and need to be modeled by recommendation system; (b) the user similarity based on user-item interactions; (c) “Facebook” relationship among them; (d) “Following” relationship on Twitter; (e) the naive combination of (c) and (d); (f) the adaptive combination reflecting the correct user similarity in (b).

5.2 Graphical Representation of ComSoc-Adap

5.3 MAP with Error Bars on Tencent Collections (ComSoc-Adap) [Data bars are the means of MAP; the error bars are the upper and lower limits with 95% confidence.]

5.4 MAP with Error Bars on Douban Collections (ComSoc-Adap)

5.5 Parameter and Efficiency Analysis (ComSoc-Adap)

6.1 Illustration

6.2 Performance Comparisons (ComSoc-ML). The error bars are the upper and lower limits with 95% confidence.

6.3 Performance Analysis (ComSoc-ML).

7.1 Map/Reduce Illustration (from Hadoop Tutorial of Yahoo!).

7.2 Speedup Comparisons between Map/AllReduce and Map/Reduce
LIST OF TABLES

1.1 Definition of Notations 5
1.2 Summary of Data Characteristics 5
2.1 Community Measures 16
2.2 Transfer Link Prediction Approaches Summary 32
2.3 Summary of Different Kinds of Social Network Analysis 36
2.4 Summary of Composite Community Discovery and Link Prediction 36
3.1 Performance Comparisons on MAP (ComSoc-MMSB) 49
4.1 Performance Comparisons on MAP (ComSoc-IT) 76
5.1 Summary of Data Characteristics 85
5.2 Performance Comparisons on Douban Collection (ComSoc-Ada p) 86
5.3 Performance Comparisons on Tencent Collection (ComSoc-Adap) 87
6.1 Summary of Data Characteristics 102
6.2 Performance Comparisons on MAP and AUC (ComSoc-ML) 103
COMPOSITE SOCIAL NETWORKS ANALYSIS

by

ERHENG ZHONG

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

ABSTRACT

People are interconnected through online social networks ubiquitous nowadays. The analysis of these networks also attracts many research interests with a broad range of applications. Various studies have been presented to study the network structure as well as users’ social characteristics. Despite of their success, most previous research works focus on analyzing individual networks. However, data in individual networks can be quite sparse and each individual social network may reflect only partial aspects of users’ social behaviors. Building models on such networks may overfit the rare observations and fail to capture the whole picture of users’ social interests. In reality, nowadays people join multiple networks for different purposes. For example, users may use Facebook to connect with their friends, talk with their families on Skype and follow celebrities on Twitter, etc. Thus, different networks are correlated with each other and nested together as composite social networks by the shared users. If we consider these users as the bridge, fragmented knowledge in individual networks can be utilized collectively to build more accurate models and obtain comprehensive understandings of users’ social behaviors.

In this research, our main idea is to extract common knowledge from different networks to solve the data sparsity problem but takes care of the network differences. We propose a general framework, known as ComSoc, based on hierarchical Bayesian models, by encoding common knowledge and network differences as latent factors. Based on this framework, we analyze composite social networks from four major aspects: 1). how to model the composite network structures; 2). how to model the dynamics and network co-evolution; 3). how to adaptively predict users’ social behaviors.
across social medias; and 4). how to measure users’ distances specifically in different networks. We will use large-scale social networking datasets, to carry out this research, in order to demonstrate how our ComSoc framework can be instantiated for solving these four problems. Finally, to handle big data, we propose a novel parallel framework that makes the model inference efficient.
CHAPTER 1

INTRODUCTION

Humans are social beings. Studying human’s social behaviors is always an important scientific research topic in computer science, social science, economics, and biology etc. With the recent development of Web 2.0, online social networks, such as Facebook, Google+, Twitter, LinkedIn, Flickr, YouTube, Last.fm, Github, etc. are becoming more and more popular and the produced massive data provide a possible way to analyze human’s preference, characteristics, and background comprehensively. For example, people can update their status, post information and interact with their friends on Facebook and Google+, browse news and discuss with their friends on Twitter, hunt jobs and follow professional information on LinkedIn, share their favorites images, videos and music on Flickr, YouTube and Last.fm respectively and program collaboratively with other software engineers on Github, and so on. Even in other online applications, such as portal, blog and electronic commerce web sites, social functions also play an important role. In a word, online social networks have infiltrated every fields of our daily and professional lives. Thus, analyzing these networks has not only academic significance but also importance social impacts.

Importantly, different social applications are not independent but interconnected. As people may join different applications for different purposes, users of different applications are shared. By considering these common users as the bridge, different networks are nested together. Without loss of generality, such nested networks are called composite social networks. In this thesis, we propose to model these composite networks from two aspects: composite network structure modeling and cross-network user characteristics inference. These research can help promote various social applications. For example, modeling composite network structures can be used for friend recommendation in multiple social medias and thus enhance current users’ experiences and lift the level of users’ activities, while inferring user characteristics cross-network can model users’ interests and hence improve the precision of personalization and advertising. In addition, departing from many existing network models that study single networks, our research on composite social networks leverage the rich semantics of typed relationships and heterogeneous user behaviors from multiple nested networks.
1.1 Motivation and Overview

Social network analysis has attracted many research interests, including link prediction [66], community detection [76], influence analysis [113], and node classification [6], etc. and benefits a broad range of applications, such as personalization [68], behavior targeting [63], social marketing [53], etc.

Despite their success, previous research works focus on individual-network analysis, which may suffer from the following problems. Firstly, data in each individual network can be very sparse, as most users may have only a few neighbors and limited behavior records in a social media application, and this is particularly true for a new network that is starting to form. Learning from such sparse network can cause the model to overfit rare observed links. Secondly, information in a single network can be incomplete. As in different time period, people may have different active levels in different applications. For example, one user builds more links on LinkedIn and interact less with her friends on Facebook when she just graduates. If we model each network independently, we cannot infer the graduation activity correctly, as people have different reasons to be inactive on Facebook and use LinkedIn when they just want to change jobs. Finally, each individual network may capture only partial aspects of users’ social interests. For example, friendship relationships on Facebook reflect users’ acquaintances in daily life, but contacts on YouTube capture users’ common preferences on videos instead.

Instead, users in composite social networks have various relationships, and can both exhibit different behaviors in each individual network and share some common latent interests across networks in the same time. For example, many people use Facebook to communicate with their friends, but post their real-time information on Twitter and browse video uploaded by others on YouTube. Unfortunately, due to the network differences, we cannot combine networks simply and apply existing algorithms. First, different networks have different properties, such as different density, degree distributions, clustering coefficient or diameters. If we merge two networks together, their specific network structures can be destroyed. For example, if we merge a dense network and a sparse network directly, the knowledge in the sparse network will be hidden and its network structure cannot be maintained. Second, users have different interests and generate different communities in different networks. Two users who are friends on Facebook are not necessary to follow each other on Twitter. That means the knowledge from different networks may be conflicted with each other and the simply combined network may misguide the model building. In addition, data in different social medias can be various and heterogeneous, including users’ social relations, profiles,
activities as well as social interactions. To exploit such heterogeneous knowledge, we have to unify their representations before mining. Through these common users, pieces of knowledge embedded in different individual networks may be exploited collectively to build comprehensive models and achieve superior performance. Thus, the motivation of this research is to model the heterogeneous knowledge from multiple networks together in a unified framework in order to benefit from the enriched knowledge, while resolving the network differences.

1.1.1 What is Composite Social Network?

Generally, a composite social network can be considered as a set networks of which users overlap with each other, as shown in Figure 1.1, where one user joins multiple social networking applications at the same time. We define this formally as follows. The notations are summarized in Table 1.1. Let $\mathcal{G} = \{G_i = (U_i, E_i)\}_{i=1}^\ell$ be a set of social networks, where $G_i$ is the $i$-th individual network, $U_i$ is the user set of $G_i$, $E_i$ is the user relationship of $U_i$, and $\ell$ is the total number of individual networks. The whole user set is $U = \bigcup\{U_i\}_{i=1}^\ell = \{u_j\}_{j=1}^N$ and the link set is $E = \bigcup\{E_i\}_{i=1}^\ell$, where $N$ is the number of users. Besides social relations, people have profiles (age, gender, education, etc.) and daily behaviors (have watched Harry Potter, a fan of Lady Gaga, etc.), and thus each user can be represented as a feature vector, $x \in \mathbb{R}^{1 \times Q}$ where $Q$ is the number of features and each dimension denotes one characteristic or one behavior. Let $X = \{x_i\}_{i=1}^N$ denote the feature matrices for $U$. In addition, we let $V = \{v_j\}_{j=1}^Q$ be an item set, where $v_j$ is the $j$-th item, which corresponds to each dimension in $X$. Finally, user pairs have interactive behaviors, e.g., chat with friends, reviewing friends’ posts, etc. We represent these as $R = \{r_{ij}\}_{u_i, u_j \in U}$, where $r_{ij} \in \mathbb{R}^{1 \times z}$. 

Figure 1.1: Composite Social Network Illustration. Each user can join multiple social networks and different social networks are nested together by the shared users.
An important property is that users in different individual networks overlap, and formally,

\[ U_i \cap U_j \neq \emptyset, \forall i \neq j, U_i \subseteq U, U_j \subseteq U \]  

(1.1)

A composite network may evolve with time. For this situation, we consider a sequence of composite networks, denoted by \( \{G^t\}_{t=1}^T \), where \( G^t = \{G^t_i = (U_i, E^t_i)\}_{i=1}^\ell \) is the composite network observed at time \( t \). In each \( G^t_i \), \( G^t_i \) is the \( i \)-th individual network, \( U_i \) is the user set of \( G^t_i \), \( E^t_i \) is the user relationship of \( U_i \) at time \( t \), where \( E^t = \bigcup \{E^t_i \mid i=1 \ldots \ell \} \). We assume the set of users \( U \) is constant. Although new users can join networks and existing users can leave networks, we can still include these users in \( U \) but consider them to be inactive. In summary, as shown in Figure 1.2, a composite social network contains three different kinds of data:
Table 1.1: Definition of Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Notation Description</th>
<th>Notation</th>
<th>Notation Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = { G_i }_{i=1}^\ell$</td>
<td>Composite Social Network</td>
<td>$\ell$</td>
<td>Number of networks in $G$</td>
</tr>
<tr>
<td>$\mathcal{G} = { \mathcal{G}<em>t }</em>{t=1}^T$</td>
<td>Sequence of Composite Networks</td>
<td>$T$</td>
<td>Number of time stamps</td>
</tr>
<tr>
<td>$U = { u_i }_{i=1}^N$</td>
<td>User set of $\mathcal{G}$</td>
<td>$N$</td>
<td>Number of users in $U$</td>
</tr>
<tr>
<td>$E = \bigcup { E_i }_{i=1}^\ell$</td>
<td>Link set of $\mathcal{G}$</td>
<td>$M_i$</td>
<td>Number of links in $E_i$</td>
</tr>
<tr>
<td>$V = { v_i }_{i=1}^Q$</td>
<td>Attribute set</td>
<td>$Q$</td>
<td>Number of Attributes in $V$</td>
</tr>
<tr>
<td>$X = { x_i }_{i=1}^Q$</td>
<td>Users’ features</td>
<td>$F$</td>
<td>Number of non-empty attributes</td>
</tr>
<tr>
<td>$R = { r_{ij} }_{u_i,u_j \in U}$</td>
<td>Social Behaviors</td>
<td>$Z$</td>
<td>Number of social behaviors</td>
</tr>
</tbody>
</table>

Distributions

| Dir(·) | Dirichlet distribution | Beta(·) | Beta distribution |
| Multi(·) | Multi-nominal distribution | Gam(·) | Gamma distribution |
| Bern(·) | Bernoulli distribution | N(·) | Gaussian distribution |
| $\theta, \phi$ and $\psi$ | Topic proportions | $\alpha, \beta, \gamma, \nu$ | Priors/Hyperparameters |

Model

| $K$ | Number of topics/communities |
| $w_i$ | Latent factors of user $u_i$ |
| $B$ | Community compatibility matrix |
| $\lambda_k$ | Network-specific factors of $G_k$ |
| $\pi_i$ | Membership vector of user $u_i$ |

Table 1.2: Summary of Data Characteristics

<table>
<thead>
<tr>
<th>Collections</th>
<th>#User</th>
<th>#Links</th>
<th>#T</th>
<th>Starting time</th>
<th>#T</th>
<th>Types of Relations/Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tencent</td>
<td>$\sim 1.3M$</td>
<td>$\sim 110M$</td>
<td>12 weeks</td>
<td>Instant Messaging (QQ), Microblog Following (MB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Douban</td>
<td>$\sim 0.05M$</td>
<td>$\sim 1M$</td>
<td>32 months</td>
<td>Online, Offline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epinions</td>
<td>$\sim 0.1M$</td>
<td>$\sim 0.8M$</td>
<td>2001/01</td>
<td>Trust, Distrust</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facebook</td>
<td>$\sim 0.06M$</td>
<td>$\sim 1.8M$</td>
<td>2011/06</td>
<td>Link, Wall Posting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Renren</td>
<td>$\sim 0.5M$</td>
<td>$\sim 32M$</td>
<td>2012/04</td>
<td>Footprint, Visiting, Talking, Buddy Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Twitter</td>
<td>$\sim 0.3M$</td>
<td>$\sim 0.9M$</td>
<td>2009/09</td>
<td>Forwarding, Mention</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sina Weibo</td>
<td>$\sim 6M$</td>
<td>$\sim 320M$</td>
<td>2009/12</td>
<td>Forwarding, Mention</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Github</td>
<td>$\sim 0.05M$</td>
<td>$\sim 1M$</td>
<td>2012/03</td>
<td>Following, Collaborating</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StackOverflow</td>
<td>$\sim 0.8M$</td>
<td>$\sim 33M$</td>
<td>2008/11</td>
<td>Answering, Commenting, Voting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Social relations $E$ in different individual networks and their dynamics $E^\ell$
- User features $X$, including users’ profiles and daily behaviors
- Social interactions $R$, e.g., chat, reviewing posts, visiting friends’ home pages, etc.

As follows, we list nine real-world composite social networks that we use in our experiments.

The datasets come from different companies:

- **Tencent**\(^1\) contains an instant messaging network (QQ) and a Microblog network (MB)
- **Epinion**\(^2\) has peoples’ trust and distrust relationships

\(^1\)http://www.tencent.com/en-us/index.shtml  
\(^2\)http://konect.uni-koblenz.de/networks/epinions
Figure 1.3: Illustration of Degree Distributions
Facebook\(^3\) captures users’ friendships and wall posting actions

Renren\(^4\) contains users’ four kinds of interactions: leaving footprint, visiting friends’ homepages, talking with friends, and sending buddy applications

Twitter\[^50\] contains two kinds of interactions: forward (RT) and mention (@) others’ tweets

Weibo\(^5\) contains two kinds of interactions: forward (RT) and mention (@) others’ tweets

Github\(^6\) has two kinds of relationships: collaborators and followers

StackOverflow\(^7\) contains three kinds of interactions: answer others’ questions, comment on others’ answers and vote others’ posts

According to different link types, these datasets can be classified as relational networks, e.g., Tencent and Epinion, and interaction networks (the remaining six datasets). User pairs in relational networks are distinct but users can interact with each other multiple times in interaction networks. Data statistics can be found in Table 1.2. Their degree distributions are plotted in Figure 1.3. Although the degree distributions all follow the power-low distribution, different individual networks in each collection have similar but different properties. It also implies that simply merging networks does not work, where networks’ specific patterns can be hidden. In summary, these datasets come from different applications with different scales. The correlations between individual networks are very different. For example, forwarding and mention may have highly positive correlation while trust and distrust relations are strongly negative correlated. This variety property makes the experiments convincing. In addition, user groups in different networks are different. For example, most users in Tencent, Weibo and Renren collections come from China, while most users in other networks come from other countries. To crawl data, we employ random walk based sampling method to select sub networks in Tencent, Douban and Renren networks and extract relational knowledge from whole public data dumps of other datasets. In each dataset, users in different individual networks can be identified by unified user identity, such as the QQ number in the Tencent collection. In addition, some recent techniques can also solve this user alignment problem [62, 123, 124].

As we observe above, some single networks contain directional links while other networks contain bi-directional links. Considering that our proposed models are all designed for datasets with

\(^3\)http://socialnetworks.mpi-sws.org  
\(^4\)http://www.renren.com  
\(^5\)http://www.wise2012.cs.ucy.ac.cy/challenge.html  
\(^6\)http://www.githubarchive.org/  
\(^7\)http://meta.stackoverflow.com/
directional links, each bi-directional link is separated into two links. In addition, both directional and bi-directional links are used to infer users’ community memberships across networks. From this aspect, bi-directional links can be used to infer users’ community memberships which are also the basic to infer directional links in our proposed models.

1.1.2 Why is Composite Social Network Analysis a New Problem?

Numerous methods have been developed for the analysis of individual social networks. However, most of these methods cannot be directly applied to mining composite social networks. This is not only because heterogeneous links across entities of different types may carry rather different semantic meanings but also because composite social network in general captures much richer information than its simply combined network counterpart. As stated above, an individual network can be obtained by simply merging multiple links from nested networks, but with significant information loss. Moreover, with rich heterogeneous information preserved in an original composite social network, many powerful and novel data mining functions need to be developed to explore the rich information hidden in the heterogeneous relationships across networks. To achieve these, we have to overcome the following challenges when modeling composite networks

1. As we stated above, different networks may have different properties and different networks may require different kinds of knowledge from other networks. Thus, we need to decide which knowledge are useful for other networks and can be exploited before designing models.

2. In social networks, users may have different levels of activities and different applications have different requirements. Not all the users need auxiliary knowledge and not all applications can utilize networks from multiple networks. This process should be encoded into the model building process automatically.

3. As knowledge are encoded in different kinds of data with heterogeneous representations, such as social relations and user behaviors. To utilize them collectively, we need to extract knowledge from these heterogeneous data and transform them into a unified space during the model building.

1.2 Research Objectives and Main Contributions

Given the heterogeneous data from multiple nested social networks, we aim to research the following objectives:
1. **Objective 1: Modeling the Composite Social Network Structure** First of all, given a composite social network, how can we describe the network structure using a compact model and predict its structure changes? This process can help us understand the differences and the commonalities between different individual networks. It also benefit other research objectives. Specifically, we plan to model the generation process of each edge exists between an arbitrary user pair in all individual networks by transferring relational knowledge from other networks.

2. **Objective 2: Modeling the Dynamics of Composite Social Networks** As networks evolve with time and different networks can influence each other, we also aim to model the dynamics in composite social network. On one hand, we plan to extract the common structure changes and specific growth patterns of each individual network. On the other hand, we plan to detect the interactions between different networks and understand the network co-evolution. Formally, the goal is to construct the composite network in next time stamp given the previous network sequence.

3. **Objective 3: Adaptive User Behavior Prediction across Social Medias** Users perform different kinds of behaviors inside different social medias, and accurate prediction of user behaviors is important for many social media applications. In addition, users’ behaviors are closely related to their social relations and different social relations reflect users’ behaviors from different aspects. Based on these motivations, we plan to predict users’ unobserved behaviors/profiles in different individual networks using the historical behavior records with the help of relational knowledge in the composite network.

4. **Objective 4: Adaptive User Distance Modeling** One important challenge in social network analysis is how to model users’ distance as a single measure. Accurate modeling of users’ distance is the basic for other analysis tasks. We plan to measure the distance between any two users in each individual network by exploiting the knowledge in users’ features, social behaviors and social relations collectively. Due to the enriched knowledge in composite social networks, the constructed distance metric would be more accurately.

The main contribution of this thesis are two folds. Firstly, we present a new and important research problems on social network analysis, where the target is to exploit the knowledge from multiple networks to solve the data sparsity problem. Secondly, we propose a hierarchical Bayesian framework that can exploit knowledge from different networks and under different representation. The motivation of the framework is to benefit from the enriched information of multiple networks.
while avoiding negative impacts resulting from network differences. Specifically, we introduce a latent feature vector $w_i$ to describe each user $u_i$. This vector can be understood as one user’s latent interests. These interests determine users’ profiles/behaviors and their social relations. Since this vector is related to all individual networks and thus it can be considered as a bridge to capture the cross-network knowledge. In addition, for each individual network $G_k$, we assign a latent feature matrix $\lambda_k$. This matrix encodes the network’s specific properties, such as main styles, application fields, etc. These matrices influence users’ concrete behaviors and social relations in different individual networks and hence capture the network differences. In addition, we represent each attribute as a vector $v_j$. This vector represents each profile/behavior’s distribution over users’ latent interests. Then, for all users and networks, we have $W = \{w_i\}_{i=1}^N$, $\lambda = \{\lambda_k\}_{k=1}^\ell$ and $V = \{v_j\}_{j=1}^Q$.

Based on these latent variables, we aim to build a model $\Theta$, which can maximize the probabilities of the observed data $G$ and $X$.

$$\max_{\Theta} P(G, X|\Theta, W, \lambda, V) \cdot P(W, \lambda|\Lambda) - R(\Theta)$$  \hspace{1cm} (1.2)

where $\Lambda$ denotes the priors of the latent variables and $R(\Theta)$ is the regularization term to control the complexity of the model. Then, for each research objective, we can derive the specific function:

1. **Objective 1: Modeling the Composite Social Network Structure** (Chapter 3)

$$\max_{\Theta_1} P(G|\Theta_1, W, \lambda) \cdot P(W, \lambda|\Lambda) - R(\Theta_1)$$  \hspace{1cm} (1.3)

We will construct link and community structures in composite networks $G$ by building a compact model to describe users’ and networks’ characteristics.

2. **Objective 2: Modeling the Dynamics of Composite Social Networks** (Chapter 4)

$$\max_{\Theta_2} P(G|\Theta_2, W, \lambda) \cdot P(W, \lambda|\Lambda) - R(\Theta_2)$$  \hspace{1cm} (1.4)

We propose to model the networks as well as their changes with time $G$ by extending the model built from the first objective.

3. **Objective 3: Adaptive User Behavior Prediction across Social Medias** (Chapter 5)

$$\max_{\Theta_3} P(X|\Theta_3, W, \lambda, V, G) \cdot P(W, \lambda, V|\Lambda) - R(\Theta_3)$$  \hspace{1cm} (1.5)

Our target is to infer users’ features $X$, including their behaviors and profiles, with the help of relational data in composite social networks.
4. **Objective 4: Adaptive User Distance Modeling** (Chapter 6)

\[
\max_{\Theta_4} P(G|\Theta_4, W, \lambda, V, X) \cdot P(W, \lambda, V|\Lambda) - R(\Theta_4)
\]

We aim to measure the distance between any two given users, by utilizing the existing link information, community structures, behaviors and profiles in a unified framework.

In the following chapters, we instantiate these objective functions to solve specific real-world applications by building different hierarchical Bayesian models.

### 1.3 Executive Summary of Key Discoveries

We conclude the key findings of this thesis by analyzing the real-world composite social networks.

- Social networks are different but correlated as analyzed in Chapter 3 and 4. Although users may have different behaviors in different social networks, each user may follow a consistent pattern, which can be extracted to enhance the network structure modeling in each individual network. In addition, users’ behaviors are influenced across networks, which is useful to model the network dynamics and network evolution.

- Users’ behaviors are closely related to their social relations but network-dependent as shown in Chapter 5. The closeness between social relations and behaviors provides us an effective solution to improve the user behavior prediction. However, different users may be affected by neighbors from different networks, thus we have to pick different networks for different users adaptively.

- Users in different networks have different but related distances, which can be modeled by users’ relational knowledge, community information, social interactions and behaviors all contribute to modeling. By modeling user distance, the accuracy of link prediction and community detection can be improved as shown in Chapter 6.

### 1.4 Thesis Outline

The thesis is organized into seven chapters as shown in Figure 1.4. Firstly, we introduce our motivations, problem settings on composite social networks, challenges and contributions in Introduction. In chapter 2, we first survey social network analysis, multi-relational network analysis works, and
cross-network knowledge transfer works; and then we summarize some closely related techniques. In chapters 3, 4, 5 and 6, we study each of the four different research objects and instantiate the learning framework correspondingly. In addition, we suggest a novel parallel computing framework to scale-up the model building of ComSoc in chapter 7. Finally, in chapter 8, we conclude our works from a unified view, and then we list some future research directions.

Figure 1.4: Thesis Outline
CHAPTER 2

SURVEY ON COMPOSITE NETWORK ANALYSIS

In this section, we survey the works in composite network analysis. Firstly, we review several typical social network analysis tasks: link prediction, community discovery and influence analysis with the main approaches by category and the related applications. Secondly, we will review a series of methods on composite network analysis with multi-relational knowledge, especially on link prediction and community discovery, followed by summarization of several composite network mining approaches which transfer knowledge across networks, where most approaches focus on link prediction. We will also provide some analysis on the properties of these algorithms in these two areas. This research can also be considered as a novel transfer learning paradigm on social network analysis. Finally, we briefly introduce three techniques that will be used to construct models in this research, including mixed membership models, topic models and metric learning.

2.1 Social Network Analysis

Social networks and the infrastructure built around them can support a rich variety of data analytic applications such as search, text analysis, image analysis, and sensor applications, as well as the analysis and evolution of the structure of the social network itself. In this section, we focus on three user-oriented applications: link prediction, community discovery and influence analysis. Let $G = \{U, E\}$ denote a social network, where $U = \{u_i\}_{i=1}^N$ represents the nodes and $E$ indicates the set of observed edges.

2.1.1 Link Prediction

The task of link prediction (Figure 2.1) is to predict how likely an unobserved edge $e_{ij} \notin E$ exists between an arbitrary pair of nodes $(u_i, u_j)$ in the social network $G$. Link prediction has been applied in many real applications. For example, one can apply this technology to predict who are friend and recommend objects, such as movies, music, books, etc., to given users. There are a variety of techniques for the link prediction problem, ranging from graph theory, metric learning, and statistical relational learning to matrix factorization and probabilistic graphical models.
In general, the family of link prediction problems contains link existence prediction (i.e., does a relationship exist?), link classification (i.e., what type of the relationship?), and link regression (i.e., how tight is a relationship?). The link prediction models can mainly fall into three categories in accordance with the intuitions of their solutions: similarity-based approaches, stochastic block models and probabilistic models. Generally, similarity-based approaches try to measure or learn the similarity/distance between two objects, and the prediction is made based on such similarity/distance; stochastic block models capture the relations between nodes through the community structure; probabilistic models try to encapsulate all the elements in data networks, e.g. objects, relationships, into a compact model.

**Similarity-based Approaches**

This kind of methods set their targets as seeking some similarity measurements $Sim(u_i, u_j)$ for each pair of users, and then link prediction is achieved by putting a link between users that are at a similarity from each other larger than a fixed threshold $\delta$. Such similarity measurements could be either defined by human, or determined with the help of machine learning techniques. To compute the similarity, two kinds of approaches can be adopted, one is using local similarity indices, such as Common Neighbors(CN) [75] and its variants [51]. Let $\Gamma(u)$ denote the set of neighbors of $u$, and the simplest measure of this neighborhood overlap is the directed count, namely

$$Sim^{CN}(u_i, u_j) = \frac{|\Gamma(u_i) \cap \Gamma(u_j)|}{\omega(u_i, u_j)} \tag{2.1}$$
where \( \omega(u_i, u_j) \) is some adjustment term depending on \( u_i \) and \( u_j \). For example, in Salton Index [94], 
\[
\omega(u_i, u_j) = \sqrt{d_{u_i} \times d_{u_j}},
\]
where \( d_v \) is the degree of node \( v \). On the other hand, one can explore global similarity indices [46] as well, such as topological patterns. Take Katz Index as an example,

\[
Sim^{\text{Katz}}(u_i, u_j) = \sum_{\ell=1}^{\infty} \gamma^\ell \cdot |\text{paths}^\ell_{u_i, u_j}|
\]

(2.2)

where \( \gamma \) is a free parameter controlling the path weights and \( \text{paths}^\ell_{u_i, u_j} \) is the set of all paths with length \( \ell \) connecting \( u_i \) and \( u_j \). There are also other criteria which are based on random walks, such as SimRank [43].

**Stochastic Block Models**

Stochastic block model is one of the most general network approaches, where nodes are partitioned into groups and the probability that two nodes are connected depends solely on the groups which they belong to. The stochastic block models can capture the community structure [93], role-to-role connections [38] and maybe other factors for the establishing of connections, especially when the group membership plays a considerable role in determining how nodes interact with each other. Among these methods, Mixed Membership Stochastic Blockmodel [3] is a state-of-the-art that generates links based users’ community memberships and community relations.

**Probabilistic Models**

Probabilistic models aim at abstracting the underlying structure from the observed network, and then predicting the missing links by using the learned models. It contains three mainstream methods, respectively called Probabilistic Relational Model (PRM) [31], Probabilistic Entity Relationship Model (PERM) [41] and Stochastic Relational Model (SRM) [121]. Among them, PRMs represent a joint probability distribution over the attributes of a relational dataset, which exploit the dependency between properties with an object and among the related objects; while PERMs consider both relations between properties and between objects. Finally, SRMs model the stochastic structure of entity relationships (i.e., links) via a tensor interaction of multiple Gaussian Processes (GPs), each defined on one type of entities.

### 2.1.2 Community Discovery

It has been shown that most of social networks exhibit strong modular nature or community structure. One example is in Figure 2.2, which contain three communities. An important research
agenda thus is to identify communities of interest and study their behavior over time. We first state some measures of community and then present several typical algorithms.

**Community Measures**

To measure and define communities, here are several important criteria to measure the quality of the extracted clusters on the network, we summarize them in Table 2.1. Let $m_i$ denote the number of edges in the $i$th partition, $n_i$ denote the number of nodes in the $i$th partition, $E(m_i)$ denote the expected number of edges between the nodes in the $i$th partition in a random graph with the same node degree sequence, $c(i)$ denote the number of edges needed to be removed to disconnect the $i$th partition from the rest of the network and $d(i)$ denote the total number of nodes’ degrees in the $i$th partition.

<table>
<thead>
<tr>
<th>Name</th>
<th>Expression</th>
<th>Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity</td>
<td>$\frac{1}{2m} \left( m_i - E(m_i) \right)$</td>
<td>Modularity Ratio</td>
<td>$\frac{m_i}{E(m_i)}$</td>
</tr>
<tr>
<td>Volume</td>
<td>$\sum_i d(i)$</td>
<td>Edges cut</td>
<td>$c(i)$</td>
</tr>
<tr>
<td>Internal Density</td>
<td>$1 - \frac{m_i}{n_i (n_i - 1)/2}$</td>
<td>Cut Ratio</td>
<td>$\frac{c(i)}{n_i (n-n_i)}$</td>
</tr>
<tr>
<td>Normalized Cut</td>
<td>$\frac{c(i)}{2m_i + c(i)} + \frac{c(i)}{2(m-n_i)+c(i)}$</td>
<td>Maximum-ODF</td>
<td>$\max_{u \in V_i} \left{ \frac{\left</td>
</tr>
<tr>
<td>Conductance</td>
<td>$\frac{c_i}{2m_i + c(i)}$</td>
<td>Expansion</td>
<td>$\frac{c(i)}{n_i}$</td>
</tr>
</tbody>
</table>
Discovery Algorithms

According to the different community cluster generative processes, mounts of existed algorithms can be divided into following categories: classical algorithms, spectral methods, hierarchical algorithms, hybrid algorithms and Markov clustering. We describe several core methods briefly below.

Classical Algorithms  The Kernighan-Lin(KL) algorithm [49] is one of the classic graph partitioning algorithms which optimizes the KL objective function, i.e., minimizing the edge cut while keeping the cluster sizes balanced, as shown in Eq.(2.3),

\[
\min \sum_{i \neq j} C(V_i, V_j) \quad s.t. \quad \bigcup_{i=m}^k V_m = V; \quad V_i \cap V_j = \emptyset; \quad |V_i| = |V_j|
\]

(2.3)

where \( k \) is the number of communities and \( C(V_i, V_j) \) denotes the the sum of edge affinities between vertices in \( V_i \) and \( V_j \). It is a cut-based algorithm and follows an iterations style. It starts with an initial partition of the social network. At each iteration, the algorithm searches for a subset of vertices from each part of the social network to reduce the edge cut and swaps them. This process is repeated until convergence.

Spectral Methods  Spectral clustering methods [67] generally partition the social network based on the eigenvectors of matrices, such as the adjacency matrix of the network itself or other related matrices. The nodes of the network are defined as an embedding in a K-dimensional space using the top-K eigenvectors. After that, one can use classical data clustering techniques such as K-means to derive the final partitions of the social network. Formally, let \( A \) be the adjacency matrix of \( G \) and \( D \) is the diagonal matrix with degrees of the nodes along the diagonal. The Laplacian matrix \( L \) can be further defined as \( D^{-1/2}(D - A)D^{-1/2} \). By subsequently decomposing \( L \), the eigenvectors can be obtained, which correspond to the \( K \) smallest non-zero eigenvalues.

Hierarchical Algorithms  Multi-level Graph Partitioning [44] is one such approach and its main idea is to shrink or coarsen the input graph successively so as to obtain a small graph, partition this small graph and then successively project this partition back up to the original graph, refining the partition at each step along the way. It contains three steps. The coarsening is to produce a smaller graph that is similar to the original graph and will repeat several times to obtain a small enough network which can be partitioned quickly. After that, it will partition the coarsest graph using some strategies, such as spectral clustering. Finally, the uncoarsening phase initializes a partition on the
bigger graph and refines the partition using the finer connectivity structure of the graph. This step is repeated until it reaches the initial network.

**Hybrid Algorithms**  Metis+MQI is an effective hybrid method for finding low-conductance cuts, which first exploits fast graph bi-partitioning program Metis [45] to split the network into two equal-sized pieces, and then runs an exact flow-based technique MQI [52] for searching the lowest conductance cut whose small side is contained in one of the two half-graphs chosen by Metis. According to the experiment results in [55], Metis+MQI is good at finding lower conductance cuts.

**Markov Clustering**  This kind of algorithms [108] clusters the nodes in the network via manipulation of the stochastic matrix or transition probability matrix corresponding to the graph. It contains two operations on stochastic matrices, *Expand* and *Inflate*, where \( \text{Expand}(A) = A \times A \) and \( \text{Inflate}(A, r) \) raises each entry in the matrix \( A \) to the inflation parameter \( r \) followed by re-normalizing the columns to sum to 1. These two operations are repeated alternatively until convergence, starting with the initial adjacent matrix \( A \). Finally, the obtained \( A \) represents the similarities between users and then any clustering algorithms can be applied to compute the communities.

### 2.1.3 Social Influence Analysis

Social influence is the behavioral change of one user because of the perceived relationships with other users. One example is shown in Figure 2.3, where \( A \) and \( B \) are two dominating users who will give impacts on their neighbors. Social influence has been a widely accepted phenomenon in social networks for decades. It has been exploited to support many applications, such as marketing, advertisement and recommendations, with the exponential growth of online social network...
services. We describe the statistical measurements related to social influence and then present the research on social influence maximization briefly as follows.

**Influence Related Measures**

At the local level, social influence is a directional effect from node $u_i$ to node $u_j$, and is related to the edge strength from $u_i$ to $u_j$. On a global level, some nodes can have intrinsically higher influence than others due to network structure. There are two kinds of criteria to measure such strength and influence, edge measures [74] and node measures. Since the node measurements, such as Katz centrality, are similar to those described in Section 2.1.1, we skip them here. Similar to the ones used in link prediction, edge measures relate to the influence-based concepts and measures on a pair of nodes. Such measures explain simple influence-related processes and inter-actions between individual nodes. One simplest criterion is to use the common neighbors as shown in Eq.(2.1).

However, this is not enough since they can have no common friends. To address this, another measure based on the hypothesis of triadic closure has been proposed. The motivation is that if strong ties connect $u_i$ with $u_j$ and $u_j$ with $u_k$, then $u_i$ and $u_k$ are likely to be connected by a strong tie as well. This is naturally related to the problem of triangle counting in a network. Let $n_\triangle$ denote the number of triangles in the network, and the clustering coefficient is formally defined as $6n_\triangle/N$.

Another important measure is the edge betweenness [30], which measures the total amount of flow across the edge. The idea is to gradually remove edges of high betweenness scores to turn the network into a hierarchy of disconnected components. These disconnected components will be the clusters of nodes in the network. Then users in the same cluster may have high influence on each other.

**Influence Maximization**

One important aspect of influence analysis is influence maximization [5] which is often motivated by the determination of potential customers for marketing purposes. The goal is to minimize marketing cost and more generally to maximize profit. Previously, the problem has mainly been studied in marketing decision or business management. However, the methods in these fields model the marketing decision process as a “black box”. Several important issues have not been fully solved, including how users influence each other once a set of users have been marketed, how they will influence their neighbors and how the diffusion process will continue. To address these, different models have been introduced.
Diffusion Influence Model  This model assigns each user with one status: active or inactive. All users are considered as inactive at the beginning. Then, the chosen users are activated, who may further influence their friends (neighbor nodes) to be active as well. The simplest model is to quantify the influence of each node with some heuristics, including high-degree heuristic [13], low-distance heuristic [29], degree discount heuristic [19], linear threshold model [47] and general cascade model [47].

Learning to Predict Customers  After learning the influence probability, the second step is to select a set of potentially profitable users. One classical model is to formalize the social network as Markov random fields [60], where each user’s probability of buying is modeled as a function of both the intrinsic desirability of the product for the user and the influence of other users. Beside this, some other works aim to find the optimal marketing strategy by directly maximizing the revenue rather than social influence [86].

Maximizing the Spread of Influence  One way to maximize the profit is to maximize the spread of influence. The proposed linear threshold model and independent cascade model in [47] use a sub-modular function [28] to achieve this by approximating the influence function.

Applications

There are other applications using influence analysis. To perform advertisement targeting, the work in [89] proposes methods for identifying brand-specific audiences without utilizing the user private information. The work in [36] proposes another tractable approach for viral marketing is through frequent pattern mining, which focuses on the actions performed by the users, under the assumption that users are able to see their friends’ actions. In addition, the work in [2] presents a model which describes the influence flow in the influence-graph consisting of all the Blog pages.

2.2 Multi-relational Network Analysis

Although mounts of researches have been done on single-relational social network analysis, humans are interacting with different social networks every day. People may have different friends and play different roles in different social networks. Thus, it is more reasonable to study multi-relational social networks in order to obtain a comprehensive view on users’ social behavior. By now, there are only a few works on multi-relational social network analysis. Most of them study
the link prediction and community discovery problems while others are related to information diffusion, statistical analysis and influence analysis, etc.

### 2.2.1 Link Prediction in Multi-relational Social Networks

This kind of work aims to predict the links based on the multi-relational information. There are two typical methods, where one is to learn a weight vectors which can compose multiple relations into an integrated one, the other is to exploit tensor factorization which can explore the multi-relational information directly.

#### Integration Methods

We introduce two approaches here, where both of which aim to learn the weight of each single relation. Let $\text{Sim}(u_i, u_j)$ denote the similarity between two nodes and it can be defined as

$$
\text{Sim}(u_i, u_j) = \sum_{z=1}^{k} w_z \text{Sim}_z(u_i, u_j)
$$

(2.4)

where $\text{Sim}_z(u_i, u_j)$ is the similarity between $u_i$ and $u_j$ of the $z$th single relation and $w_z$ indicates its importance.

#### Link Classification

MRLP in [24] is a link classification method based on the occurrence of each unique 3-node substructure, which is a weighted extension of the neighborhood methods. Users $u_i$ and $u_j$ form a partial triad with each common neighbor $n \in \Gamma(u_i) \cap \Gamma(u_j)$, and each partial triad provides a probabilistic weight based on the triad census. Let $\text{pattern}(u_i, n, u_j)$ denote the triad pattern among nodes $u_i$, $n$, and $u_j$. Some examples can be found in Figure.2.4.
For any two users, \( u_i \) and \( u_j \), let \( \Gamma(v) \) denote one’s neighbors and then the probability of generating a link with category \( x \) is related to the score

\[
Sim_x(u_i, u_j) = \sum_{n \in \Gamma(u_i) \cap \Gamma(u_j)} \Upsilon_n^{(x)}
\]  

in which \( \text{pattern}(u_i, n, u_j) \) describes the node and edge type pattern of the network path \((u_i, n, u_j)\), and

\[
\Upsilon_n^{(x)} = \frac{\sigma p(\text{edge type}(u_i, u_j) = x | \text{pattern}(u_i, n, u_j))}{p(\text{edge type}(u_i, n)) p(\text{edge type}(u_j, n))}
\]

where the sign is determined by statistical comparison rather than numerical and \( p(x) \) denotes the probability of the link of category \( x \). Finally, it can classify each edge into one category by ranking on these scores. In addition, to increase the performance, it can be extended by adding a regularization term as

\[
Sim_x(u_i, u_j) = \sum_{n \in \Gamma(u_i) \cap \Gamma(u_j)} \frac{1}{\log \phi(u_i, u_j)}
\]

Let \( t_1 \) denote the edge type of \(<u_i, n>\), \( t_2 \) denote the edge type of \(<n, u_j>\) and \( |\Gamma_n(x)| \) denote the number of edges of \( n \) with edge type \( y \). Then the \( \phi(u_i, n, u_j) \) is

\[
\phi(u_i, n, u_j) = \begin{cases} 
|\Gamma_n(t_1)|, & t_1 = t_2; \\
|\Gamma_n(t_1)| + |\Gamma_n(t_2)|, & t_1 \neq t_2.
\end{cases}
\]

**Link Existence Prediction**  
The approach proposed in [80] aims to predict the mobile App installation based on multiple relationships between users. It exploits a memory-based model to predict the link between users and the Apps. Formally, the probability of installing the \( j \)th App by the \( i \)th user with one single relation is defined by

\[
p(I_{ij} = 1) = \frac{1}{|\Gamma(v_i)|} \sum_{u : u \in \Gamma(v_i)} p(I_{uj} = 1)
\]

By considering the multiple networks, this equation can be rewrote as

\[
p(I_{ij} = 1) = \frac{1}{k} \sum_{c=1}^{k} w_c \frac{1}{|\Gamma_c(v_i)|} \sum_{u : u \in \Gamma_c(v_i)} p(I_{uj} = 1)
\]
The details are as follows. First, the conditional probability of the installation of the $j$th App by user $v$ given the probabilities of the neighbors can be defined as

$$p(I_{vj} = 1|I_{uj} = 1 : u \in \Gamma(u)) = 1 - \exp(-s_v - p(I_{vj} = 1))$$

where $\forall v, s_v > 0$. $s_v$ captures the individual susceptibility of Apps, regardless of which App. Then the target becomes to learn a weight vector $w = \{w_c\}_{c=1}^k$ and susceptibility vector $s = \{s_i\}_{i=1}^{|V|}$. Formally, by maximizing the likelihood, the objective function is

$$\max_{w,s} \sum_j \left[ \sum_{v: I_{vj} = 1} \log(1 - \exp(-s_v - p(I_{vj} = 1))) - \sum_{v: I_{vj} = 0} (s_v + p(I_{vj} = 1)) \right]$$

This objective is concave and can be solved effectively. After learning the $w$ and $s$, unobserved links can be predicted.

### Tensor Factorization Methods

For link prediction with single relation, matrix factorization methods decompose the user interaction matrix to obtain users’s latent representation $U$ and then reconstruct the matrix to obtain the unobserved links. Similarly, for multi-relational networks, tensor factorization [96] can be applied. Let $T$ denote a $N \times N \times \ell$ tensor which represents $\ell$ different relations among $N$ users. Then the objective function is

$$\min_{U,S} \| T - U \circ U \circ S \|_F^2 + \gamma(\| U \|_F^2 + \| S \|_F^2)$$

where $U$ is a $N \times d$ matrix which is the latent representation of users, $S$ is a $N \times d$ matrix which represents the types of links and $\gamma$ is a trade-off parameter for regularization terms. The $\cdot$ is element-wise production, where $T_{ijk} = \sum_{c=1}^d U_{ic} \times U_{jc} \times S_{kc}$. This function can be alternatively solved by quasi-Newton methods. For example, by fixing $U$, the Grassmann gradient and the Hessian matrix of the objective function with respect to the $S$ can be computed and used to update $S$ and vice versa. After obtaining $U$ and $S$, the value of the empty element in $T$ can be predicted [33].

As a summary, the integration methods are efficiency since the weight coefficients can be computed fast by regression techniques. However, these methods do not consider an important property: different users would play different roles in different social networks. On the other hand, tensor based approaches can handle the multi-relational information more naturally but they suffer from high computation cost. In addition, they may fail when the tensor is extremely sparse.
2.2.2 Community Discovery using Multiple Relations

Similar to the link prediction, typical community discovery algorithms aim to combine the information from multiple relations into an integrated one, including integration-based method, tensor factorization and Multi-view clustering. Beside, [104] studies the dynamic multi-relational community using evolutionary clustering methods.

Integration-based Methods

There are two ways to perform integration: one is to compose multiple relations into one and adapt existed methods in the single-relational network, and the other is to extend single-relational community measures under the multi-relational context.

Regression-based

The basic idea of regression-based method [15] is trying to find a combined relation which makes the relationship between the intra-community examples as tight as possible and at the same time the relationship between the inter-community examples as loose as possible. The method in [15] applies Ridge Regression to solve this problem. Let $\tilde{I}$ denote the community indicator in the composite network, where $\tilde{I}_{ij} = 1$ indicates that the $i$th and $j$th users belong to the same community. Then its target is to solve the following problem:

$$
\min_{\mathbf{w}} \sum_{c=1}^{k} w_c \mathbf{I}_c - \sum_{c=1}^{k} w_c \mathbf{I}_c
$$

(2.14)

where $\mathbf{I}_c$ is the community indicator with the $c$th single relation. Once the combination coefficients are computed the hidden relation strength any user pair can be predicted. After that, existed community discovery algorithms can be applied. The naive implementation needs to consider every elements in the indicator matrix, say $N(N - 1)/2$ elements. Instead, vectors $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{\ell}$ can be used to replace each indicator matrix in order to accelerate the computing, where the rank of each $\mathbf{v}_i$ is much smaller than $N(N - 1)/2$.

Modularity-based

The method in [105] called Principal Modularity Maximization (PMM) extends the modularity measure by considering multi-relations. It defines a measure called total modularity.

$$
Q = \frac{1}{k} \sum_{i=1}^{k} Q_i = tr\left(S^T \left[\frac{1}{k} \sum_{i=1}^{k} \left\{ \frac{A_i}{2m_i} - \frac{dd^T}{(2m_i)^2} \right\}\right] S \right)
$$

(2.15)
where $A_i$ is the adjacent matrix based on the $i$th single relation, $S$ is the community indicator matrix, $d_i = \sum_j A_{ij}$, $m_i$ is the number of edges in the $i$th network, and $\text{tr}(\cdot)$ is the trace of the matrix. To maximize this criterion, it adopts a two phrases algorithm. The first step is structural feature extraction which computes a low-dimensional embedding using the top eigenvectors of the modularity matrix. The motivation of this operation is that the eigenvectors can be treated as the important structural features extracted from the network. And then it extracts structural features from each single relation and keeps those eigenvectors with a positive eigenvalue only. The second step is cross-relation analysis. It exploits canonical correlation analysis to find a transformation for each set of variables such that the pairwise correlations are maximized. The whole process of detecting communities is as follows.

1. Compute top eigenvectors of the modularity matrix for each $A_i$
2. Select the vectors with positive eigenvalues as $S_i$
3. Compute slim SVD of $X = [S_1, S_2, \ldots, S_k] = UDV^T$
4. Obtain lower-dimensional embedding $\tilde{U} = U(:, k - 1)$
5. Calculate the cluster idx with k-means on $\tilde{U}$

**Tensor-based Methods**

[64] extends the work in [105] by replacing the modularity matrices on each single relation as an integration tensor and proposes two tensor-based multi-view clustering methods, where the first one is MC-OI and the second one is MC-MI. MC-OI aims to solve the following object:

$$
\max_U \sum_{i=1}^k \| U^T I_k U \|_F^2 = \| T \circ U^T \circ U \|_F^2 
$$

(2.16)

where $T$ is the tensor where each slice is a similarity matrix which represents the similarity based on one single relation and $U$ is the joint optimal subspace matrix we which needs to be learnt. MC-MI adopts another decomposition style where

$$
\max_{U,V,W} \| T \circ U^T \circ V^T \circ W^T \|_F^2 
$$

(2.17)

After decomposing, they perform k-means on $U$ and obtain the communities. The whole process of detecting communities is as follows.

1. Build a modularity-based tensor $A$
2. Perform high order SVD (HOSVD) on $A$ to obtain singular vectors $U$
3. Obtain lower-dimensional embedding $\tilde{U} = U(:, k - 1)$
4. Calculate the cluster index with k-means on $\tilde{U}$

In addition, [115] proposes two tensor-based multi-view clustering methods which can be applied on the multi-relation context.

**Multi-view Clustering**

Parallel Integration Clustering Algorithm (PICA) in [37] aggregates information from heterogeneous, incomplete, and potentially discordant single relations. Rather than working on the original data, PICA takes its input as a collection of “base clusterings” constructed independently on each available view. These will typically be generated by applying a partition algorithm such as k-means that will frequently converge to different local minima. We denote the collection of clusterings generated on the the network $G_\ell$ by $C_\ell$, and the complete set of base clusterings for all views by $C = \{C_1, \ldots, C_k\}$. Given the input $C$, PICA follows a two-stage process. The first step produces a set of local models $\{L_1, \ldots, L_k\}$, where $L_i = U_iV$ represents a model, in the form of a “soft” clustering. $U_i$ is a projection matrix which indicates the membership of users in each clusters and $V$ represents the meta-cluster centers. The second step combines the local models to produce a global model by averaging the $U_i$ to generate $\tilde{U}$. After that, one can compute which community is belonged for given users.

As a summary, the integration methods are efficiency since the weight coefficients can be computed fast by regression technique. However, these methods ignore the user-dependent difference in multi-relational networks. On the other hand, matrix or tensor based approaches can handle the multi-relational problem more naturally but they suffer from high computation cost. This is also true for the multi-view clustering based method.

**2.2.3 Other Applications**

Besides link prediction and community detection in multi-relational networks, there are some other research works that are related to multi-relational network analysis.

**Information Diffusion**

Previously, [14] discussed a model for the diffusion of information through heterogeneous social networks from social science perspective. It allows a sender of information to retain that information after telling it to somebody else. Consequently, it allows more actors to have information
during the information diffusion process. The model also provides predictions of diffusion times in a given network at the global, dyadic, and individual levels. The model is applied on an undirected and connected composite network. It is based on a Markov chain directly described by a transition matrix $W$, where $W_{ij} = \frac{\sum_c E_{ij}^c}{\sum_j \sum_c E_{ij}^c}$. This Markov chain has $n$ states, and each state represents one actor who is able to influence others or give information to others. Each entry in $W$ gives the probability that one actor will give information to somebody else while not retaining the information himself. Thus, the process represented by this Markov chain resembles passing a package between the actors in the network according to the probabilities in $W$. The expected time for information to go from one actor to another is based on the expected number of steps this package needs to travel from one actor to another, including the possibility that the package will return to the original actor.

We observe that this method just simply combined all single relations to form an integrated one in order to obtain the transition probability. Recent work [71] improves this by adding a weight to each single relation and suggests a more robust criterion.

$$W_{ij} = \sum_{i=1}^{k} \alpha_i \frac{\Gamma(v_i) \cap \Gamma(v_j)}{A_i + A_j}$$

(2.18)

where $A_i$ and $A_j$ represent the number of actions done by the $i$th and $j$th users, respectively. After that, it performs clustering using $W$ and studies the information flow among clusters.

**Information Extraction**

[57] proposes an approach for egocentric information abstraction. It exploits a vector space representation for composites social networks to identify linear combination of relations as features and computes statistical dependencies as feature values and designs several abstraction criteria to distill representative and important information to construct the abstracted graphs for visualization. The method contains four steps:

1. Feature extraction: a set of features are automatically selected and extracted according to surrounding network substructure, which will serve as the basis of summarization.
2. Statistic dependency measure: statistic dependency measures between the features and the ego node are generated. Based on this measures, vector-based summarization tables can be generated, which each row vector in the table is a summarization of one node in the network.
3. Information distilling: less relevant information will be removed using certain distilling criteria, including local frequency, local rarity, and relative frequency.
4. Abstracted graph construction: an egocentric abstracted graph can be constructed in an incremental manner that allows the users to visualize the results, where the graph is composed of only the distilled linear combination of relations and the corresponding nodes.

Besides, [59] proposes a matrix factorization based method to summarize social activity from rich media social networks. Let each activity theme be a cluster of strongly co-occurring users, actions and terms, and then the problem becomes to a multi-graph clustering problem. Furthermore, by considering with time regularization and different actions, the problem can be solved by a collective matrix factorization algorithm [100].

Network Sampling

To capture the characteristics of very large social networks, graph sampling is an important approach that does not require visiting the entire network. [58] studies whether a sampling method can preserve the node and link type distribution of the composite social networks. After that, it proposes different sampling methods to address these problems.

- Random-Based Sampling. There are two variants. Random Node Sampling (RNS) first picks up a set of nodes into a list. Then it constructs the vertex-induced subgraph by checking if there are edges between the selected nodes in the original graph. Random Edge Sampling (RES) selects edge randomly and includes the connected edges of the head and tail of the selected edge.

- Chain-Referral Sampling. This type algorithm is an exploration-based sampling which is based on the Random Walk (RW) methods. Starting with a random node, RW chooses exactly one neighbor of that start node as the next stop. Next, each new selected node itself is a new ego and the algorithm repeats the same step iteratively until the desired sample size is reached. In addition, it allows multiple egos as the sampling starts at initial to avoid the case that the start ego of ECE falls in a strongly connected component where individuals tend to be connected to those of the same type.

- Respondent-Driven Sampling. It contains two phrases: snowball sampling phase and the Markov chain process. The first phrase is similar to chain-referral sampling, while the second phase helps to generate unbiased samples. As opposed to the conventional sampling methods, the statistics are not obtained directly from the samples, but indirectly inferred from the social network information constructed through them.
On the other hand, the work in [102] analyzes multi-relational social interaction networks in a large-scale commercial massively multiplayer online role-playing game. It studied 6 interactions: friend, whisper, party invitation, trade, mail and shop. After that, it studied three problems: 1. the number of gamers were engaged in each interaction type, and the number of interactions; 2. the clustering coefficients; and 3. Pearson correlation coefficients between the in- and the out-degree of each node, and the reciprocity of interactions.

**Node Classification**

[85] addresses a task of multi-label classification for data organized in a composite network using an iteration algorithm. It performs with two components:

1. Learning step: It consists two parts in the training phrases: a classification function for classifying a node using its content and the labels of its neighbors. This function is learned on the labeled part of the graph using classical machine learning algorithms like perceptron, or SVM. The learned function is called re-estimation function in the following.

2. Inference step: The inference iteratively selects at random unlabeled nodes and computes their new label using the re-estimation function considering the current labels assignment of their neighbors. This step is repeated until convergence or, for a manually chosen number of iterations.

Importantly, the ability of the model to handle complex classification tasks depends on this function. The paper proposes two schemes to achieve this. One is generic propagation scheme (GPS) and the other is label propagation scheme (LPS). For GPS, given a particular label $\ell$, the scores of the neighbors of a node $v_i$ for label $\ell$ will propagate among all relations connecting $v_i$ to its neighbors, and scores are weighted according to the relation type. For the LPS, the weights will depend not only on the relation type, but both on the relation type and on the label. By defining this transition probability, it can propagate the labels through the multi-relational graph structure and perform prediction.

### 2.3 Cross-network Transfer Learning

Most realistic networks, such as social networks, user-item interaction networks, and advertisement click network etc., are sparse, since most users have limited friends, and interact with limited items,
such as movies, music and web page, etc. We call this as “sparsity” problem, which is especially serious when the network is start-up. In recommendation system, it is referred to the “Cold Start” problem. Due to the over-fitting caused by sparsity, many approaches may fail to capture users’ profiles and then perform poorly in the prediction process. One solution here is to exploit transfer learning [79] by borrowing knowledge cross-network. Let $S$ denote the source domain and $T$ denote the target domain. The aim of transfer learning is to exploit knowledge from $S$ to help the learning in the current task $T$. Different from the traditional transfer learning which focuses on classification and clustering based on isolate data where each instance is independent, network based transfer is applied on relational data, where each example is related to others. Most current works focus on transfer link prediction. In addition, graph-based classification and sentiment analysis are also studied.

### 2.3.1 Domain Adaptation

Over the years, transfer learning has received much attention in machine learning research and practice. Researchers have found that a major bottleneck associated with machine learning is the lack of labels or ratings to help train a model. In response, transfer learning offers an attractive solution for this problem. Various transfer learning methods are designed to extract the useful knowledge from different but related auxiliary domains. Recently, transfer learning has been demonstrated to be successfully on many different applications, ranging from text mining [83], sentiment analysis [12], advertising [8], image classification [90], multimedia analysis [117], bioinformatics [112], metric learning [127], collaborative filtering [56] to personalization [65]. Besides the traditional knowledge transfer, different transfer learning settings are also introduced, such as heterogeneous transfer learning [118], which transfers knowledge across different feature spaces [118] or label spaces [98]; multi-view transfer learning [106], where data from different domains are represented from different perspectives; multi-source transfer learning [119], where auxiliary knowledge come from different source domains. As stated in the survey of transferred text mining [83], knowledge transfer can be understood from two views:

- In theory, transfer learning can be considered as a new learning paradigm, where most non-transfer learning methods are considered as a special case when learning happens within a single target domain only, e.g., text classification in Twitter.

- In applications, transfer learning can be considered as a new cross-domain learning technique, since it explicitly addresses the various aspects of domain differences, e.g. data distri-
bution, feature and label space, noise in the auxiliary data, relevance of auxiliary and target
domains, etc. For example, we have to address most of the above issues when we transfer
knowledge from Wikipedia documents to twitter text.

Basically, existing methods can be divided into three categories namely model-based, instance-
based and feature-based transfer. Formally, these methods can be described as follows. Let $D_T$
and $D_S$ represent the source and target data respectively, $\Theta_0$ denote the original model trained
from source domains, $\Theta$ denote the model we aim to build in the target domain, and $\Upsilon$ denote the
feature mapping from original feature space to a new feature space.

1. model-based transfer, which studies on how to reuse a model trained on some auxiliary data

$$\min_{\Theta} \ell(\Theta, D_T) + R(\Theta, \Theta_0)$$  (2.19)

where $\ell$ denotes the loss function to approximate the target data, $R(\Theta, \Theta_0)$ is the regular-
ization term that encodes the model learnt from source domains. A representative algorithm
is Adaptive SVM [117], which transfers the model parameters of a learned SVM in source
domains to a target domain via biased regularization.

2. instance-based transfer, which studies on how to leverage auxiliary data instances

$$\min_{\Theta} \ell(\Theta, D_T, D_S) + \varpi(\Theta, D_T, D_S) + R(\Theta)$$  (2.20)

where $\varpi(w, D_T, D_S)$ is a loss function that aims to minimize the distribution distance be-
tween source and target domains. A representative algorithm is TrAdaboost [23], which adjusts
the weights of source instances through boosting techniques.

3. feature-based transfer, which studies on how to bridge two domains via feature transfor-
mation or learning

$$\min_{\Theta, \Upsilon} \ell(\Theta, \Upsilon(D_T), \Upsilon(D_S)) + \varpi(\Upsilon, D_T, D_S) + R(\Theta)$$  (2.21)

where $\varpi(\Upsilon, D_T, D_S)$ measures the data distance between source and target domains under
the new feature space. A representative algorithm is kMap [129], which searches an optimal
feature space through kernel mapping.
2.3.2 Transfer Link Prediction

Table 2.2: Transfer Link Prediction Approaches Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>User</th>
<th>Item</th>
<th>Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-GP [17], MCF [126], BPR-CL [103], MDUP [65]</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>CBT [56], TagCDCF [99]</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>TCF [81]</td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>CST [82]</td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>

1 √ indicates correspondence. CST requires two source domains to regularize \( U_t \) and \( V_t \) respectively.

Under the network-based context, \( S \) is formed by \( \{M_s, N_s, R_s, I_s \subseteq M_s \otimes N_s \otimes R_s\} \) where \( M_s \) is the set of \( m_s \) users in the source domain, \( N_s \) is the set of \( n_s \) items in the source domain, \( R_s \) is the action set, such as rating, click, etc. and \( I_s \) is the set containing the interactions between users and items, which can be represented as a sparse matrix. The parallel notation for \( T \) is \( T = \{M_t, N_t, R_t, I_t \subseteq U_t \otimes V_t \otimes R_t\} \). At different settings, \( M_s, N_s \) and \( R_s \) can be no-correspondence, partial-correspondence and full-correspondence with \( M_t, N_t \) and \( R_t \). We summarized them in Table 2.2. From the methodology aspect, although most methods propose matrix factorization based methods, we can also classify them into two categories:

1. Regularization-based [56, 81, 82, 99, 103, 65]: it regularizes the factorized matrices of \( I_t \) using \( I_s \)

2. Relativeness-based [17, 126]: it explores relativeness between \( I_s \) and \( I_t \) to make compose prediction

We review several state-of-the-art algorithms belonging to each category respectively.

Regularization-based Transfer

Most regularization-base methods are based on triple matrix factorization. For triple matrix factorization, it aims to factorize matrix \( I_t \) using three latent matrices, \( U_t, V_t \) and \( \Sigma_t \):

\[
I_t = U_t \Sigma_t V_t^T
\]  

where \( U_t \) is a \( m_t \times d \) matrix, \( V_t \) is a \( n_t \times d \) matrix, \( \Sigma_t \) is a \( d \times d \) matrix and \( d \) is the number of latent dimensions. The difference between these approaches is the way to exploit different strategies to
regulate \( U_t, V_t \) or \( \Sigma_t \) using the knowledge from the source domain. Specifically, Codebook Transfer (CBT) [56] decomposes \( I_s \) into \( U_s, V_s \) and \( \Sigma_s \). After that, it obtains the codebook \( B \) using:

\[
B = [U_s^T \Sigma_s V_s] \odot [U_s^T 11^T V_s]
\]  

(2.23)

where \( \odot \) means the element-wise division. After obtaining the codebook, the user-item interaction patterns can be transferred from \( I_s \) to \( I_t \) through minimizing the following object.

\[
\min_{U_t, V_t} \| (I_t - U_t B V_t^T) \circ W_t \|^2
\]  

(2.24)

where \( W_t \) is the indicator to show which element in \( I_t \) is non-zero. Different from CBT, Transfer by Collective Factorization (TCF) proposed in [81] regulates the other two latent matrices \( U_t \) and \( U_s \) and follows collective matrix factorization. Formally, it aims to solve the following object function:

\[
\min_{U, V, \Sigma_t, \Sigma_s} \| (I_t - U \Sigma_t V_t^T) \circ W_t \|^2 + \| (I_s - U_s \Sigma_s V_s^T) \circ W_s \|^2
\]  

(2.25)

It is easy to observe that the target and source domains share the same latent matrices \( U \) and \( V \) and then make the knowledge transfer available. Similar to it, Coordinate System Transfer (CST) [82] adds source knowledge into \( U_t \) and \( V_t \) but apply \( U_s \) and \( U_t \) as priors. It solves the following optimization problem:

\[
\min_{U_t, V_t, \Sigma_t, \Sigma_s} \| (I_t - U_t \Sigma_t V_t^T) \circ W_t \|^2 + \rho_u \| U_t - U_s \|^2 + \rho_v \| V_t - V_s \|^2
\]  

(2.26)

where \( \rho_u \) and \( \rho_v \) are regularization parameters. As shown in Table.2.2, these approaches have different application contexts, and thus lead the different regularization forms. More specifically, CBT requires no correspondence for neither users nor items, TCF requires full correspondence for both users and items but with different rating scales, while CST requires correspondence for users and items from different source domains. In addition, approach in [99] proposes to regulate \( U_t \) and \( V_t \) at the same time by using tags.

Different from the matrix factorization aspect, the method in [65] models users’ preference from two aspects: one is dependent on users and the other is related to domains, based on topic models. It generates the user-item interaction as follows.

1. For each domain \( d \) do
   
   - Generate latent feature vector \( \lambda_d \sim N(0, \sigma_d) \)

2. For all users \( u \) do
• Generate latent feature vector $x_u \sim \mathcal{N}(0, \varphi)$

• For all domains $d$
  
  – Let user-domain feature vector be $\alpha_{du} = \text{lgt}(\lambda_d x_u)$
  
  – Draw user-domain topic distribution, $\theta_{du} \sim \text{Dir}(\alpha_{du})$
  
  – For each observed user-item pair
    
    A. Draw topic $z \sim \text{Multi}(\theta_{du})$
    
    B. Draw item $v \sim \text{Multi}(\phi_z)$

where $\text{lgt}(x) = \log(1 + e^x)$, and $\sigma_d$ and $\varphi$ are variances. The method in [103] adopts a similar idea but is based on matrix factorization, where it separates the latent matrix $U$ into a sum of other two matrices. [21] proposes an algorithm to suggest recipients for an email, which can enhance information flow. It considers each user as a task, and shares hidden variables for prediction across different users. The model is based on topic model and the difference is that it imposes a common prior to each single user and then composites the knowledge from different users.

**Relativeness-based Models**

We introduce two generative models here. Multi-relational Gaussian Process (M-GP) [17] adopts Gaussian process (GP) by applying a domain relativeness kernel. To exploit GP for collaborative filtering, it rewrites the matrix factorization above as $I_t \sim \mathcal{N}(U_t V_t^T, \sigma_t^2 \mathbf{I})$, and then it can be further transformed by marginalizing $p(U_t) = \Pi_i \mathcal{N}(u_i | 0, \beta_{tu}^{-1})$

$$
p(I_t | V_t, \sigma^2, \beta_{tu}) = \Pi_i \mathcal{N}(I_{ti} | 0, \beta_{tu}^{-1} V_t^T V_t + \sigma^2 \mathbf{I})
$$

(2.27)

In addition, to perform non-linear prediction, it replaces $V_t^T V_t$ as a kernel $K_t$, and then rewrites the function again

$$
p(I_t | V_t, \sigma^2, \beta_{tu}) = \Pi_i \mathcal{N}(I_{ti} | 0, K_t + \sigma^2 \mathbf{I})
$$

(2.28)

By now, one can define a kernel to perform collaborative filtering. To cope with the domain relativeness, M-GP explores a new kernel as

$$
C = T \otimes K
$$

(2.29)

where $T$ is a semi-define matrix describing the relativeness among domains and $K$ is the kernel depends on each instance. By incorporating the domain relativeness, knowledge can be transferred. Similarly, MCF [126] defines a matrix $T$ to represent the domain relativeness. However,
this matrix models the relativeness between the latent matrices $U$ in different domains. It converts each $U$ into a vector in a column-wise manner and uses these vectors to generate a matrix $U = \{ \text{vec}(U_1), \ldots, \text{vec}(U_k) \}$. Based on this, its object can be defined as

$$
\min_{U_i, V_i, T} \sum_{i=s,t} \left\{ \| (I_i - U_i V_i^T) \circ W_i \|_2^2 + \rho_u \| U_i \|_2^2 + \rho_v \| V_i \|_2^2 \right\} + \text{tr}(U U^T) \tag{2.30}
$$

where $\text{tr}(\cdot)$ is the trace of one matrix. By treating the $T$ as a bridge, different $U$ can regulate each other.

As a summary, regularization-based matrix factorization is more efficiency than other approaches since its complexity increases linearly with the number of ratings, while the drawback is that it requires two domains being highly correlated and may suffer from negative-transfer if two domains are very different. On the other hand, current relativeness-based models have high computation cost due to the use of kernels and the trace-based optimization but they can adjust the impacts among different domains and then reduce the influence of negative transfer. Finally, the model in [65] can be scaled up while considering the domain difference at the same time.

### 2.3.3 Cross-network Node Classification

#### Graph-based Classification

Despite those classification methods introduced in [79], [40] proposes a graph-based transfer approach which can be applied in network data based on label propagation. Let $S = \{ x_i, y_i \}_{i=1}^{n_s}$ and $T = \{ x_i, y_i \}_{i=1}^{n_t} \cup \{ x_i \}_{i=1}^{n_u}$, where all instances are connected. TRITER [40] first constructs a graph which can represent the data unified. The graph $G$ contains three types of objects: instance, feature and label. It assumes that the source and target domains share the same features and distributions but with different labels. Based on $G$, a random walk based approach can be applied. At the beginning, the nodes which represent labels will be assigned with corresponding values. After that, at each iteration, these labels will be propagated to their neighbors. This process repeats until convergence. At last, each instance node will receive their labels and the prediction is done. Similar to this approach, Margin-Graph [128] is a similar solution to address transfer learning by performing instance selection on $G$.

[120] proposes another approach for graph-based transfer learning based on Gaussian process (GP). It aims to minimize the following object.

$$
\min_{f, \Sigma} = \sum_{J=\{S, T\}} \left( - \sum_{x_i \in J} \log p(y_i, f(i)) + \frac{1}{2} f_j \Sigma^{-1} f_j \right) + \log(|\Sigma|) \tag{2.31}
$$
Table 2.3: Summary of Different Kinds of Social Network Analysis

<table>
<thead>
<tr>
<th></th>
<th>Homogeneous</th>
<th>Heterogeneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single relation</td>
<td>Traditional SNA (Section 2.1)</td>
<td>Works in Section 2.3</td>
</tr>
<tr>
<td>Multiple relations</td>
<td>Works in Section 2.2</td>
<td>Composite SNA (This Thesis)</td>
</tr>
</tbody>
</table>

where $\Sigma$ is the shared parameter between two domains and $f_S$ and $f_T$ denote the labeling models for each domain respectively.

**Opinion Bias Prediction**

Recently, [16] proposes a transfer learning strategy to perform real time sentiment analysis. It identifies a task, opinion holder bias prediction, which is strongly related to the sentiment analysis task; however, in contrast to sentiment analysis, it builds accurate models since the underlying relational data follows a stationary distribution. Instead of learning textual models to predict content polarity, the method in [16] measures the bias of social media users toward a topic, by solving a relational learning task over a network of users connected by endorsements (e.g., retweets in Twitter). It analyzes sentiments by transferring user biases to textual features first and then adopts user bias as the basis for building accurate classification models. The challenge here is to determine a training set. In other words, it should determine the user bias automatically. To address this, it introduces an Opinion Agreement Graph (OAG) where neighbors in the graph tend to hold similar opinions, where the weight of each edge is computed by active and passive similarity. Active similarity between $u_i$ and $u_j$ is defined as the lift, which is the ratio between the frequency of their co-occurrence in activities (e.g., retweets) and the expected frequency. Passive similarity is defined by the lift, which one user endorses another one in a transaction. The final similarity is the combination of these two. After that, a random walk or label propagation model can be applied to perform bias propagation through different opinion graphs over different points in time.

**2.4 Summary**

As shown above, it is crucial to exploit composite information in social network analysis. Most existed works consider only one property of composite networks, either multiple relations or heterogeneous knowledge. The former one is related to multi-relational network analysis which actually considers that there is only one single network but multiple relations can exist between two users; while the later one is related to cross-network transfer learning considers that nodes among
Table 2.4: Summary of Composite Community Discovery and Link Prediction

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Types</th>
<th>Methods</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Prediction</td>
<td>Multiple Relations</td>
<td>Integration</td>
<td>Efficient</td>
<td>Information loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tensor-based</td>
<td>Multi-context reserved</td>
<td>Sparsity, high complexity</td>
</tr>
<tr>
<td>Cross Network</td>
<td>Regularization</td>
<td>Fast, without source data</td>
<td>Negative transfer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relativeness</td>
<td>Domain differences</td>
<td>High complexity</td>
<td></td>
</tr>
<tr>
<td>Community Discovery</td>
<td>Multiple Relations</td>
<td>Integration</td>
<td>Efficient</td>
<td>Information loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tensor-based</td>
<td>Multi-context reserved</td>
<td>High complexity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi-view</td>
<td>More complicated</td>
<td>High complexity</td>
</tr>
</tbody>
</table>

different networks are overlapping but there is only one relation between two nodes. Most current researches focus on community discovery and link prediction. We summarize the corresponding algorithms in Table 2.4. However, to the best of our knowledge, our research is the first work that considers these two properties at the same time, as shown in Table 2.3.

2.5 Background of Techniques

In this section, we will review three basic techniques, which we will use to construct our models in this research, including mixed membership models, topic models and metric learning. Mixed membership models are basic building blocks for the Objective 1 and 2, topic models are used to construct models for Objective 3, and metric learning is exploited to learn the user distance for Objective 4.

Mixed Membership Stochastic Blockmodels  Mixed Membership Stochastic Blockmodels (MMSB) [3] assumes that each user $u_i \in U$ possesses a latent mixture of $K$ roles, which determine the membership of $K$ communities in the network $G$. We denote this role mixture as a normalized $K \times 1$ vector $\pi_i$. In MMSB, these vectors are drawn from some priors $P(\pi)$, such as Dirichlet distribution [3] and Logistic-Normal distribution [32]. MMSB generates a $K \times K$ community relation matrix $B$, which represents the probability of having a connection from a user in a community to another user in another community. Given the vector $\pi_i$ of each user $u_i$, the network edge $e_{ij}$ is generated stochastically as follows:

- For each pair of users $(u_i, u_j) \in E$ in the network $G$:
  - Draw indicator for $u_i$, $z_{ij} \sim \text{Mult}(\pi_i)$
  - Draw indicator for $u_j$, $z_{ji} \sim \text{Mult}(\pi_j)$
Sample the link, \( e_{ij} \sim \text{Bern}(z^T_{ij}B_{zji}) \)

where \( z_{ij} \) and \( z_{ji} \) are two \( K \times 1 \) unit indicator vectors for the sender \( u_i \) and the receiver \( u_j \) respectively. Recently, MMSB has been extended from two aspects: dynamic [32, 42, 20] and nonparametric [116], in order to model dynamical data and release the constraints on the number of communities. In addition, auxiliary data are also considered to improve the model performance [48].

**Topic Models** Topic modeling algorithms [11] are used to discover a set of “topics” from a large collection of documents, where a topic is a distribution over terms and a document is a distribution over topics. They have been used for tasks like corpus exploration, document classification, information retrieval and content personalization [109]. In recommendation system, topic models provide an interpretable low-dimensional representation of the users and items. The simplest topic model is latent Dirichlet allocation (LDA) [11]. This process reveals how the interactions of each user are assumed to come from a mixture of topics: the topic proportions are user specific, but the set of topics is shared by all users. Given a collection, the posterior distribution (or maximum likelihood estimate) of the topics reveals the \( K \) topics that likely generated its users. Assume there are \( K \) topics \( \beta = \beta_{1:K} \), each of which is a distribution over a mixed items. The generative process of LDA is as follows.

- For each user \( u_i \) in the network \( G \),
  - Draw topic proportions \( \theta_i \sim \text{Dir}(\alpha) \).
  - For each interaction \( x_{i,k} \) with item \( v_k \),
    - Draw topic assignment \( z_{i,k} \sim \text{Mult}(\theta_i) \).
    - Draw interaction \( x_{i,k} \sim \text{Mult}(\beta_{z_{i,k}}) \).

where \( \alpha \) is a prior. Multiple variants of LDA have been proposed to incorporate different auxiliary knowledge, such as meta data [72] and relation knowledge [18, 73].

**Metric Learning** Metric learning has been shown to be effective for classification [110], clustering [114] and ranking [69]. Recently, Shaw et al. [97] proposed a graph embedding approach by preserving the network structure. It is designed for general networks, but ignores the community structure that reflects users’ interests in social networks. Boosting technology has also been applied in metric learning [7], which learns combination coefficients for a series of low-rank metric matrices. For networking data, we propose to learn the network metric as follows.

\[
D(u_j, u_k|G) = (x_j - x_k)M(x_j - x_k)^T + \omega r_{jk}^T
\]  

(2.32)
where $M \succeq 0 \in \mathbb{R}^{Q \times Q}$ is a positive semi-definite (PSD) matrix to project node features and $\omega \in \mathbb{R}^{1 \times Z}$ is a $Z$ dimension vector for relational feature mapping. Metric learning for social networks is quite different from metric learning for classification and clustering; social networks contain relational data that violate the classical independence assumption in traditional machine learning and statistics, and do not have any class label information for any instance or user in the networks.
CHAPTER 3

MODELING THE COMPOSITE SOCIAL NETWORK STRUCTURE

3.1 Introduction

Modeling network structure or link structures is an important task in social network analysis (SNA). It can be used for friendship prediction which helps users identify friends and improves their level of activity. Most previous approaches modeling users’ link structures based on historical records, such as existing links, social interactions, etc. However, these research works based on single-network structure modeling may fail to predict correctly due to the data sparsity problem. Each user may have only a few neighbors and this is particularly true for a new network that is starting to form. Thus, each single network may capture only partial aspects of social interests. Learning from such sparse network can cause the model to overfit rare observed links. For example, neighborhood models may suggest friends of friends to the current user. But if a user builds links with only a few other users, most of his/her friendships that rely on unobserved friends cannot be discovered. In addition, information in a single network can be incomplete, but in the same time, different social networks may be correlated with each other. To be exact, the friendship generation process in the current network can be influenced by activities captured by some other networks. For example, if two people become friends in Google+ for similar hobbies on videos, they are likely follow each other on YouTube as well. Without priors from other networks, such links may not be predicted due to the incomplete knowledge in any individual network.

However, the composite property, that users and links across different networks overlap, sheds light on solving the above problems. By using the unified identity, such as Gmail account for Google+ and YouTube, and QQ number for QQ and Tencent’s Microblog, common users can be identified across networks. Consequently, by considering the common users in different networks as the bridge, knowledge in other networks can be transferred to the current network to overcome the “sparsity” challenge. For example, the auxiliary knowledge from other networks can be exploited in auxiliary priors, to model the generating process in the current network. Nevertheless, one cannot simply merge multiple networks due to network differences. First, different networks have different properties, such as different density, degree distributions, clustering coefficient or
diameters. If we merge two networks together, their specific network structures can be destroyed. For example, if we merge a dense network and a sparse network directly, the knowledge in the sparse network will be hidden. For example, as shown in Figure 3.1, Network-A is denser than Network-B. Simply merging may bring unnecessary and even harmful knowledge to each individual network. For example, if we combine networks A and B, the network structure in Network-B is hidden and we cannot discover users’ social interests in Network-B anymore. Second, users play different roles and generate different communities in different networks. Two users holding a link in one network are not necessary neighbors in another network. Specifically, for a given users, his/her neighbors in different single networks can overlap but may not necessary be the same. For example, one person may link with some users on both YouTube and Google+, but he/she may link with some others on YouTube only if they share similar video interests, but will not become friends with them on Google+ as they are not familiar with each other.

Thus, the motivation of the proposed work is to model the current network with other networks together in order to benefit from the enriched knowledge, while resolving the network differences. The challenge here is how to distinguish the shared and specific knowledge across networks, given the sparse data in each single network. We propose a hierarchical mixed membership model to solve the problem. The proposed model integrates friendship knowledge over multiple different social networks collectively to help the prediction in the current network. First, it utilizes an adaptive prior for each user to represent different users’ global interests over all individual networks. Since this prior is related to all nested networks and thus it can encode the common knowledge across multiple networks. Second, it introduces another network-specific prior to encode particular knowledge in the current network, in order to model the network differences. This prior helps avoid the knowledge in a sparser network to be overwhelmed by other denser networks. Finally, these two priors are combined together as a hybrid prior to build mixed membership models in individual networks. Generally, the knowledge in other networks guides the current model building through users’ global characteristics, while the network-specific priors avoid negative impacts due to network differences and simultaneously allow the current network to maintain its own properties.

### 3.2 Challenges

As we stated above, users may have different behaviors across networks and useful knowledge from auxiliary networks may hide behind relational data. Thus, we have to address the following challenges for our studied problem.
1. We have to extract common knowledge from different individual networks, where their network structures can be very different. These knowledge should be domain independent and useful for all individual networks.

2. We have to decide how to utilize the extracted knowledge to each individual network collectively, that means we need to find an appropriate representation for the common knowledge.

3. We have to represent and distinguish the specific properties of each individual network, to allow each network has own network structure.

We observe that these challenges are correlated, and are similar to three fundamental questions in transfer learning; that is, in deciding “what to transfer”, “how to transfer” and “avoiding negative transfer” in transfer learning [79]. Several works have been proposed to handle social networks with multiple relations, such as tensor factorization [33] and triangle patterns counting [24]. However, these works treat each type of relationship as equally important and do not consider network differences. That may bring unnecessary auxiliary regularization to the current network.

### 3.3 Composite Network Structure Modeling

Intuitively, we propose to generate the links in networks by modeling users’ community membership as well as the relationship between communities. Due to the community memberships and community relationship in different networks are related but different, we divide their priors into two parts: one part is to model the common knowledge which can transfer knowledge among different networks, and the other is to model network-specific knowledge. Specifically, the common knowledge can be represented as users’ latent features, which represent users’ social interests and
behaviors, e.g., generating communities with other users. In addition, as shown in Figure 3.3, the number of communities in each network $K_d$ can vary across networks and community structures in different networks can be different. That implies, in different single networks, latent interests play different roles to generate communities. Thus, we introduce a feature-community mapping to transform users’ global latent features into network-dependent community membership. As an example, each user on Google+ and YouTube can be described based on different factors, where some factors represent users’ interests on daily life and others indicate users’ video interests. Then, in Google+, the daily-life interests may play a more important role to generate users’ links, while video interests may be more important in YouTube. Based on this intuition, we propose a model called ComSoc-MMSB based on Mixed Membership Stochastic Block models (MMSB).

### 3.3.1 Model: ComSoc-MMSB

The implementation is to introduce a hybrid prior $\alpha_{id}$ for each user $u_i$ in each network $G_d$. Specifically, the prior is divided into two components: $\lambda = \{\lambda_d\}_{d=1}^D$ and $X = \{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^{1 \times D}$ represents the global interests of user $u_i$ and $\lambda_d \in \mathbb{R}^{K_d \times D}$ represents the network-specific characteristics of $G_d$. Consequently, after utilizing these two priors together, $\alpha_{id} = t(x_i, \lambda_d^T)$, where $t(x) = \log(1 + e^x)$, we can obtain a hybrid prior that indicates users’ specific membership over communities in different single networks. Importantly, this prior encodes the user-dependent and network-specific knowledge, which can exploit the auxiliary knowledge from other networks while
After that, this hybrid prior generates the membership vector $\pi_{id}$ for each user $u_i$ in each network $G_d$ following the Dirichlet distribution, where $\pi_{id}$ represents the user’s membership over communities. At the same time, for each network, the network-specific prior $\lambda_d$ is exploited to generate the community compatibility matrix $B_d$ for each network, which captures the relationship between each community, from a Beta distribution. The motivation is that, if two communities have similar feature mappings, users may have high probability to connect with each other. Then, for each user pairs, ComSoc-MMSB draws two asymmetric community indicator vectors according to the membership vectors for two users following the multi-nominal distribution. Consequently, a variable $z_{ij}^T B_d z_{ji}$ is computed according to the indicator vectors and the community compatibility matrix. The variable indicates whether the two users $u_i$ and $u_j$ are linked or not. Finally, it samples the link $e_{ij}$ between two users $u_i$ and $u_j$ from a Bernoulli distribution based on this variable. Formally, the generative process is as follows,

- For each network $G_d$:
  - Draw a $K_d \times T$ feature matrix $\lambda_d \sim N(0, \sigma_d^2 I)$
  - Draw $B_d$, $B_d^{ij} \sim Beta(t(\lambda_d \lambda_d^T)^{ij}, 1)$
- For each user $u_i \in U$:
  - Draw a $1 \times T$ latent feature vector $x_i \sim N(0, \sigma_u^2 I)$
  - For each network $G_d$:
    - Generate a network-user prior $\alpha_{id} = t(x_i \lambda_d^T)$
    - Draw a membership vector $\pi_{id} \sim Dir(\alpha_{id})$
- For each user pair $(u_i, u_j) \in E_d$ in each network $G_d$:
  - Draw indicator for $u_i$, $z_{i\rightarrow j} \sim Mult(\pi_{id})$
  - Draw indicator for $u_j$, $z_{j\rightarrow i} \sim Mult(\pi_{jd})$
  - Sample the link, $e_{ij} \sim Bern(z_{i\rightarrow j}^T B_d z_{j\rightarrow i})$

The graphical representation of ComSoc-MMSB is shown in Figure 3.2. Obviously, ComSoc-MMSB captures the network-specific and user-dependent knowledge to model common knowledge and allows different users to own different community memberships in different networks. Specifically, common knowledge from other networks is embedded in each user’s global prior $x_i$ and another prior $\lambda_d$ adjusts users’ community memberships and community-community relations $B_d$ in different networks.
Figure 3.3: Illustration of Composite Network. Different colors represent different communities. The size of each node presents each user’s membership of the community, e.g., larger size indicates higher probability of the user belonging to this community.

3.3.2 Inference

The inference process has two scenarios: one is to construct MMSB using Gibbs sampling and the other is to learn the hierarchical priors using a Metropolis-Hastings sampler. As joint inference is intractable due to the non-conjugate between hierarchical priors and the latent factors in MMSB, we perform alternate inference by calling Gibbs sampling and Metropolis-Hastings sampler in turns.
and then the pairwise community assignments in the pairwise posterior of the community distribution on the matrix including each user’ prior in the posterior distribution and thus we propose a Gibbs sampling method instead. First, let \( \rho_u = t(\lambda_u \lambda_d^T) \), and then the pairwise community assignments in the \( d \)-th network can be written as

\[
p(z^d|\alpha_d, \rho_d, E_d) \propto p(E_d|z^d, \rho_d)p(z^d|\alpha_d)
\]

\[
= \int p(E_d|z^d, B_d)dp(B_d|\rho_d) \int p(z^d|\pi_d)dp(\pi_d|\alpha_d)
\]

\[
= \prod_{k,k'} B(\rho_{d,k,k'}) \prod_{u} \Gamma(\alpha_{i,d} + n_{i,d} - 1) \prod_{u} \Gamma(\alpha_{j,d} + n_{j,d} - 1)
\]

where \( B(\omega) = \prod \Gamma(\omega_k) / \prod \Gamma(\omega_k - 1) \), \( n_{i,d} \)-th network can be defined as

\[
p(z^d_{i\rightarrow j} = k, z^d_{j\rightarrow i} = k'|\alpha_d, \rho_{d,k,k'}, z^d_{\sim(i,j)}, E_d)
\]

\[
\propto \prod_{k,k'} \Gamma(n_{i,k}^d + 1) \Gamma(n_{j,k'}^d + 1) \frac{B(n_{i,k}^d + 1 + \rho_{d,k,k'})}{B(\rho_{d,k,k'})}
\]

\[
= \prod_{k,k'} \Gamma(n_{i,k}^d + 1) \Gamma(n_{j,k'}^d + 1) \frac{\Gamma(n_{i,k}^d + 1 + \rho_{d,k,k'})}{\Gamma(\rho_{d,k,k'})}
\]

where \( z^d_{\sim(i,j)} \) denotes the set of community assignments without two assignments over the link between \( u_i \) and \( u_j \), \( n_{i,k}^d \) represents the number of user \( u_i \) picking community \( k \) in the \( d \)-th network, and \( n_{k,k',y}^d \) represents the total number of links in type \( y \) with \( (k, k') \) as the participating communities in the \( d \)-th network. In addition, \( y_{(i,j)} \) denotes the sign of the link \( e_{ij} \), where \( y_{(i,j)} = 1 \) represents that \( u_i \) and \( u_j \) are friends and \( y_{(i,j)} = -1 \) represents that \( u_i \) and \( u_j \) will not build a link between each other. Then, we can use this equation to update the community assignments iteratively.

**Constructing Mixed Membership Models** We notice that although each MMSB model in each single network is similar to the one described in [3], we assign a prior on \( B \) to restrict the probability of generating links between two communities, while the previous work does not. In addition, the original paper solved the MMSB using Variational EM, which may not approximate the true posterior distribution and thus we propose a Gibbs sampling method instead. First, let \( \rho_u = t(\lambda_u \lambda_d^T) \), and then the pairwise community assignments in the \( d \)-th network can be written as

\[
p(z^d|\alpha_d, \rho_d, E_d) \propto p(E_d|z^d, \rho_d)p(z^d|\alpha_d)
\]

\[
= \int p(E_d|z^d, B_d)dp(B_d|\rho_d) \int p(z^d|\pi_d)dp(\pi_d|\alpha_d)
\]

\[
= \prod_{k,k'} B(\rho_{d,k,k'}) \prod_{u} \Gamma(\alpha_{i,d} + n_{i,d} - 1) \prod_{u} \Gamma(\alpha_{j,d} + n_{j,d} - 1)
\]

where \( B(\omega) = \prod \Gamma(\omega_k) / \prod \Gamma(\omega_k - 1) \), \( n_{i,d} \)-th network can be defined as

\[
p(z^d_{i\rightarrow j} = k, z^d_{j\rightarrow i} = k'|\alpha_d, \rho_{d,k,k'}, z^d_{\sim(i,j)}, E_d)
\]

\[
\propto \prod_{k,k'} \Gamma(n_{i,k}^d + 1) \Gamma(n_{j,k'}^d + 1) \frac{B(n_{i,k}^d + 1 + \rho_{d,k,k'})}{B(\rho_{d,k,k'})}
\]

\[
= \prod_{k,k'} \Gamma(n_{i,k}^d + 1) \Gamma(n_{j,k'}^d + 1) \frac{\Gamma(n_{i,k}^d + 1 + \rho_{d,k,k'})}{\Gamma(\rho_{d,k,k'})}
\]

where \( z^d_{\sim(i,j)} \) denotes the set of community assignments without two assignments over the link between \( u_i \) and \( u_j \), \( n_{i,k}^d \) represents the number of user \( u_i \) picking community \( k \) in the \( d \)-th network, and \( n_{k,k',y}^d \) represents the total number of links in type \( y \) with \( (k, k') \) as the participating communities in the \( d \)-th network. In addition, \( y_{(i,j)} \) denotes the sign of the link \( e_{ij} \), where \( y_{(i,j)} = 1 \) represents that \( u_i \) and \( u_j \) are friends and \( y_{(i,j)} = -1 \) represents that \( u_i \) and \( u_j \) will not build a link between each other. Then, we can use this equation to update the community assignments iteratively.

**Sampling Hierarchical Priors** Different from the original MMSB in [3], we need to derive two priors: \( \lambda_d \) for each network and \( x_i \) for each user \( u_i \). To find the optimal values, we first define the
Then, we compute the acceptance ratio as

\[
p(z, \lambda, x) = \Pi_{d,k} \frac{\Gamma(\sum_k t(x_i) \lambda_{d,k}^T)}{\Gamma(\sum_k t(x_i) \lambda_{d,k})} \frac{\Gamma(t(x_i) \lambda_{d,k}^T + n_{i,t}^d)}{\Gamma(t(x_i) \lambda_{d,k}^T)}
\]

Note that this equation is similar to Eq.(4) in [65] and Eq.(1) in [72], but with very different terms, since we apply \( \lambda_d \) as a prior to the compact matrix \( B_d \). Let \( dt(x) = \partial_x t(x) \) be the derivative of the transform function. The derivative of the log of likelihood with respect to \( \lambda_{d,k,t} \) for a given community \( k \) and the feature \( t \) is

\[
\partial_{\lambda_{d,k,t}} \log p(z, \lambda, x) = -\frac{\lambda_{d,k,t}}{\sigma_d^2} + \sum_{d,t} x_{i,t} dt(x_i) \lambda_{d,k}^T
\]

We exploit a standard L-BFGS optimizer [61] using the above equation to update each \( \lambda_d \). Similarly, the derivative of the log of likelihood with respect to the parameter \( x_{i,t} \) is similar:

\[
\partial_{x_{i,t}} \log p(z, \lambda, x) = -\frac{x_{i,t}}{\sigma_u^2} + \sum_{d,k} x_{i,t} \lambda_{d,k,t} dt(x_i) \lambda_{d,k}^T
\]
Algorithm 1 Inference for ComSoc-MMSB

1: **Input**: Nested social networks: \( G \), Number of iterations \( I \), Number of user features \( T \) and Number of communities \( K \)
2: **Output**: Compact matrix \( B_d \) and membership vector \( \pi_id \)
3: Generate \( \lambda_d \) and \( x_i \) randomly
4: Initialize the compact matrices and membership vectors
5: **for** \( k = 1 \) to \( I \) **do**
6: Perform Gibbs sampling using Eq.(3.2)
7: Update membership vector \( \pi_id \) and compact matrix \( B_d \)
8: **IF** \( k\%10==0 \)
9: \( \lambda_d \) using L-BFGS using Eq.(3.4)
10: \( x_i \) using Metropolis-Hastings in Eq.(3.7)
11: **IF** the algorithm is convergent **Break**
12: **end for**
13: **Return** \( B_d \) and \( \pi_id \)

where \( \sigma \) is related to the learning rate. Finally, we update \( x_{i,t} \) to the new value \( \bar{x}_{i,t} \) with probability \( \min(r, 1) \).

**Algorithm** The complete inference process is described in Algorithm 1. Generally, we alternatively optimize the user and network priors, and latent variables in MMSB as well as the community assignments. Specifically, we randomly initialize the priors according to the assigned distributions. Then we use these priors to sample the latent variables of MMSB, including \( \pi \) and \( B \). Consequently, we alternatively update the community assignments and latent variables in each MMSB model, and the network-specific and user-dependent priors. This process is repeated until convergence. To avoid overfitting and reduce the computational cost, we update both \( \lambda \) and \( x \) every 10 iterations and set the number of iterations for L-BFGS as 10. At each iteration, Gibbs sampling needs to look up all \( m \) links in nested networks and then update \( \pi \) with time \( O(NKn) \) where \( N \) is the number of networks and \( n \) is the number of users. In addition, L-BFGS takes \( O(NKT) \) to update \( \lambda \) and Metropolis-Hastings sampler spends \( O(Tn) \) on updating \( x \). Typically, \( NKT \) is much smaller than \( NKn \). Thus, with \( I \) iterations, the time complexity is \( O(I(m + Tn + NKn)) \) which linearly increases with the number of links \( m \) and and number of users \( n \).

### 3.4 Experiments

We show that the proposed algorithm is better than merging all networks simply, and better than only considering single networks. Thus, we introduce two baselines based on MMSB. One is to learn an MMSB on each individual network and the other is MMSB-C which performs MMSB on a combined network, of which the edge set is the union of all edge sets from all individual net-
works. In addition, we introduce two other baselines that are applied on multi-relational networks: TF [33] and MRLP [24], where TF formulates multiple relationships as a tensor and then performs factorization and MRLP performs link prediction based on triangle patterns. The evaluation task is to predict which users will build links between each other. To test the performance, we select 10% of the links in the whole dataset according to the temporal information as the hold-out set $T$. In addition, to test whether ComSoc-MMSB can solve the data sparsity issue, we remove those popular users whose numbers of degrees are higher than the averaged degree plus one standard deviation. The numbers of user latent features and the communities in single networks are set as 25 and we will test their effectiveness. The results are evaluated by mean average precision (MAP).

$$MAP = \frac{1}{|U|} \sum_{u \in U} \frac{\sum_{(u,v) \in T_u} \frac{1}{T} \sum_{r \in [1,T]} pre_r}{|T_u|}$$ \quad (3.8)

where $pre_r$ is the precision on top $r$ predictions. Generally, MAP measures how well the algorithms rank the links in the hold-out set against the non-existing links. As there are only positive links in the dataset, we sample non-existing links in the same magnitude randomly in the training process to construct negative examples.

<table>
<thead>
<tr>
<th>Networks</th>
<th>MMSB</th>
<th>MMSB-C</th>
<th>TF</th>
<th>MRLP</th>
<th>ComSoc-MMSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tencent</td>
<td>QQ</td>
<td>0.4885</td>
<td>0.5011</td>
<td>0.5034</td>
<td>0.5473</td>
</tr>
<tr>
<td></td>
<td>MB</td>
<td>0.2053</td>
<td>0.2351</td>
<td>0.2387</td>
<td>0.2521</td>
</tr>
<tr>
<td>Douban</td>
<td>Offline</td>
<td>0.7213</td>
<td>0.7312</td>
<td>0.7343</td>
<td>0.7521</td>
</tr>
<tr>
<td></td>
<td>Online</td>
<td>0.6346</td>
<td>0.6541</td>
<td>0.6567</td>
<td>0.6793</td>
</tr>
<tr>
<td>Epinion</td>
<td>Trust</td>
<td>0.6831</td>
<td>0.6923</td>
<td>0.6933</td>
<td>0.7312</td>
</tr>
<tr>
<td></td>
<td>Distrust</td>
<td>0.6629</td>
<td>0.6679</td>
<td>0.6687</td>
<td>0.7089</td>
</tr>
<tr>
<td>Facebook</td>
<td>Link</td>
<td>0.7078</td>
<td>0.7143</td>
<td>0.7166</td>
<td>0.7365</td>
</tr>
<tr>
<td></td>
<td>Wall</td>
<td>0.6681</td>
<td>0.6761</td>
<td>0.6781</td>
<td>0.6939</td>
</tr>
<tr>
<td>Renren</td>
<td>Footprint</td>
<td>0.4912</td>
<td>0.4981</td>
<td>0.5088</td>
<td>0.5435</td>
</tr>
<tr>
<td></td>
<td>Visiting</td>
<td>0.1932</td>
<td>0.2045</td>
<td>0.2145</td>
<td>0.2431</td>
</tr>
<tr>
<td></td>
<td>Talk</td>
<td>0.2621</td>
<td>0.2777</td>
<td>0.2875</td>
<td>0.3189</td>
</tr>
<tr>
<td></td>
<td>Buddy</td>
<td>0.3388</td>
<td>0.4055</td>
<td>0.4027</td>
<td>0.4481</td>
</tr>
<tr>
<td>Twitter</td>
<td>Forwarding</td>
<td>0.6613</td>
<td>0.6741</td>
<td>0.6734</td>
<td>0.6923</td>
</tr>
<tr>
<td></td>
<td>Mention</td>
<td>0.7432</td>
<td>0.7686</td>
<td>0.7742</td>
<td>0.7921</td>
</tr>
<tr>
<td>Weibo</td>
<td>Forwarding</td>
<td>0.5357</td>
<td>0.5713</td>
<td>0.5817</td>
<td>0.6101</td>
</tr>
<tr>
<td></td>
<td>Mention</td>
<td>0.6489</td>
<td>0.6715</td>
<td>0.6887</td>
<td>0.7011</td>
</tr>
<tr>
<td>Github</td>
<td>Following</td>
<td>0.7236</td>
<td>0.7414</td>
<td>0.7501</td>
<td>0.7901</td>
</tr>
<tr>
<td></td>
<td>Collaborating</td>
<td>0.7911</td>
<td>0.8065</td>
<td>0.8078</td>
<td>0.8265</td>
</tr>
<tr>
<td>StackOverflow</td>
<td>Answering</td>
<td>0.8101</td>
<td>0.8288</td>
<td>0.8298</td>
<td>0.8532</td>
</tr>
<tr>
<td></td>
<td>Commenting</td>
<td>0.7576</td>
<td>0.8076</td>
<td>0.8012</td>
<td>0.8189</td>
</tr>
<tr>
<td></td>
<td>Voting</td>
<td>0.7732</td>
<td>0.7965</td>
<td>0.8054</td>
<td>0.8187</td>
</tr>
</tbody>
</table>
3.4.1 Performance Comparisons

We compare ComSoc-MMSB with other baselines on friendship prediction, and the results are summarized in Table 3.1. It is evident that ComSoc-MMSB achieves better performance on most datasets, except the Microblog network in Tencent collection. Due to the “sparsity” problem, MMSB in single networks fails to capture the true users’ distribution over communities under a uniform prior. For example, in Tencent collection, it achieves MAP just no more than 0.49 on
QQ-IM network and no more than 0.21 on Microblog network. By considering the information from both networks, MMSB gets better performance which improves the MAP in sparser networks, such as online network of Douban collection and Link network of Facebook collection. However, it assigns a uniform prior to all single networks and thus does not tackle with network differences. This harms the performance on many other networks, such as Microblog network in Tencent collection and Offline network in Douban collection, which are comparatively denser. In addition, for those networks that are totally different, networks in Epinion collection, MMSB-C performs significantly worse than MMSB. Thus, from another side, these results support the necessity to consider network differences. Nonetheless, ComSoc-MMSB outperforms MMSB in all datasets. It achieves up to 0.06 higher in MAP than the baseline methods on QQ-IM network in Tencent collection. Furthermore, the improvements of ComSoc-MMSB on sparser networks, such as QQ-IM in Tencent collection and online network in Douban collection, are higher than other two networks. That means ComSoc-MMSB can bring more benefits to sparser networks. The better performance of ComSoc-MMSB over MMSB-C can be ascribed to the hybrid prior. As analyzed, this prior considers the user-dependent and network-specific knowledge at the same time to get benefits from
auxiliary knowledge, while avoiding the harm of network differences. In addition, this prior is personalized for each user, and then makes each user choose the regularization strength from other networks adaptively. More importantly, ComSoc-MMSB performs better than both TF and MRLP which deal with multi-relational networks. The reason is that these two methods treat each network uniformly and do not tackle with network differences. For example, in the Epinion dataset, where the link information in two individual networks can be very different, ComSoc-MMSB outperforms them significantly.

**Long-tail Users** To examine whether knowledge in other networks can help solve the data sparsity problem in the current network, we evaluate and illustrate the results on the long-tail users, the number of whose neighbors is smaller than 10 and suffer from the harm of “sparsity” most. As shown in Figure 3.4∼3.7, ComSoc-MMSB improves MMSB on these users more than the average level. One reason is that, for those users who have large amount of friends, MMSB has gained enough knowledge to infer their community memberships, and thus their friendships can be predicted correctly. For the long-tail users, ComSoc-MMSB can exploit knowledge from other networks to enhance the prediction. This also explains the larger improvements of ComSoc-MMSB on sparser network, such as QQ-IM of Tencent collection. From the empirical perspective, this provides justification to the argument that ComSoc-MMSB can enrich the knowledge in each single network.

### 3.4.2 Performance Analysis

We perform extensive experiments on Douban dataset to address the following questions: (1) Does ComSoc-MMSB capture network differences? (2) How do the model parameters, the corresponding ratio between two networks, and the level of data sparsity affect ComSoc-MMSB’s performances? (3) What is the time cost of ComSoc-MMSB?

**Network Adaptation** We illustrate the network differences captured by ComSoc-MMSB in Figure 3.8(a) which represents the $5 \times 5$ network-specific prior matrix $\lambda$ in each network, where the grayscale indicates the values of elements in the matrices (after normalization). Each row can be considered as a community distribution over one user feature. Obviously, communities’ distributions are different between two networks. That implies ComSoc-MMSB works well in capturing the network differences. It also explains why C-MMSB fails to achieve good results because C-MMSB assigns a uniform prior over two networks. This result is also consistent with the example networks in Figure 3.3(c) and (d), where network structures in online and offline networks can be
very different, where community sizes are different and the tie strengths between different communities are also different across networks. In addition, we test the effectiveness of the corresponding ratio between two networks or the number of users who exist in two networks at the same time. Figure 3.8(b) presents the results. We see that, ComSoc-MMSB’s performance becomes worse if fewer corresponds are provided across networks. However, a more important observation is that it still outperforms each MMSB in single networks consistently as long as there are enough corresponding users between two networks. Figure 3.8(c) shows the improvement of ComSoc-MMSB
Parameter Analysis  We analyze the effectiveness of two parameters: one is the number of user features $T$ in the hybrid prior and the other is the number of communities $K$ in each single MMSB, on the Douban dataset. The former one indicates the complexity of user-network relations, and the latter one reflects the complexity of users’ representation in each single network. We fix $K = 25$ and varies $T$ from 5 to 100. The results on $T$ are illustrated in Figure 3.9(b). We observe that ComSoc-MMSB is not sensitive to $T$. We fix $T = 25$ and let $K$ increase from 5 to 100. The results on $K$ are shown in Figure 3.9(c). Different from $T$, the performance of ComSoc-MMSB goes up first and then drops down when $K$ keeps increasing. That means $K$ should be tuned to avoid overfitting.

Efficiency Analysis  As analyzed in the end of Section 2.3, the computational time of ComSoc-MMSB increases linearly with the number of interactions between users and the number of communities $K$. We evaluate these empirically as shown in Figure 3.10. Figure 3.10(a) illustrates the
computational time of MMSB-C and ComSoc-MMSB on the Douban dataset, with different ratio of links. We observe that the computational time increases linearly with larger data size. For our experimental setting, every round of inference takes about 25 seconds in our computer, of which memory is 16G and CPU is 3.2Gz. ComSoc-MMSB not only has better prediction performance as shown in Table 3.1, but also has similar time cost with MMSB-C. Figure 3.10(b) shows the computational time with different number of communities. Clearly, the time cost of ComSoc-MMSB increases linearly with $K$ as well. This suggests that ComSoc-MMSB can scale-up to handle large-scale datasets.

3.5 Summary

In this chapter, we solved the network structure modeling problem across multiple nested networks, where users in different networks overlap. Each individual network is sparse and has different properties. To utilize the shared knowledge and avoid the harm due to network differences, we proposed a hierarchical Bayesian framework by introducing adaptive and hybrid priors. Unlike prior works, the proposed method ComSoc-MMSB considers the network-specific and user-dependent knowledge together to generate users’ membership of communities. This is formulated into a hybrid prior that balances auxiliary knowledge and network differences. In addition, ComSoc-MMSB models users’ differences adaptively with personalized priors. The proposed algorithm is flexible in that it can be extended to any number of networks and the priors can be automatically adjusted with respect to the network differences, e.g., different densities. Empirical studies were conducted on eight large-scale and real-world datasets, both of which are composed by two or more different networks, where ComSoc-MMSB improves previous algorithms without taking network differences and auxiliary knowledge into account by up to 0.11 in MAP.

Applicability Discussion The proposed ComSoc-MMSB is designed under the assumption that most users will express same or similar kinds of interests in each individual networks or users behaviors are consistent in one given network. For example, in YouTube, we assume that all users want to watch movies. Under this case, users’ interests on movies can be active by the interest-community mapping for YouTube in ComSoc-MMSB. Thus, for some users who may behavior very dissimilar with other users, ComSoc-MMSB cannot infer their interests and community memberships well. For example, one user may link with some other users in one network but disconnect with them in another network on purpose, due to some privacy issues. Then ComSoc-MMSB cannot capture such behavior based only the observed links.
**Future Works** In the future, we consider exploit users’ behaviors together with relation interactions in different networks for building more accurate models. We also aim to design a model that can work more effectively under partial user correspondence among networks. Finally, we plan to apply a full Bayesian model in order to avoid overfitting on the learning of hierarchical priors.
CHAPTER 4

MODELING THE DYNAMICS OF COMPOSITE SOCIAL NETWORKS

4.1 Introduction

As an important research topic, modeling the dynamics of social networks can help people understand the evolution of network structures, e.g., the community evolution, the change of network statistics, e.g., diameter and clustering coefficient, and the shifting of users’ preferences, etc. In addition, considering the dynamic properties helps improve the performance of other tasks, such as link prediction [107] and community detection [77].

Most existing research works only study the dynamics of individual networks and do not consider their intra-network correlations, but in practice, different networks can be highly correlated. As shown in Figure 4.3(a), the ratio of common links between Tencent’s Instant Messaging and Microblog networks (QQ vs. Tencent Microblog) is much higher than the one between two random graphs (0.143 vs. 0.007 respectively). In reality, individual networks reflect only partial aspects of users’ social activities, thus the information captured by individual networks may be incomplete. For example, one user builds more links on LinkedIn and interact less with her friends on Facebook when she just graduates. If we model the dynamics in each network independently, we cannot infer the graduation activity correctly, as people have different reasons to be inactive on Facebook and use LinkedIn when they just want to change jobs. Consequently, we may fail to model this user’s social dynamics, e.g., her social interactions on Facebook in the future, without knowing the fact of graduation. In addition, the reasons that cause the network to change can come from other networks. Two users without any common neighbors may follow each other on Twitter as they are familiar on Facebook, and one user may mention someone as she has forwarded many tweets of the user. Without considering the cross-network knowledge, one cannot correctly model these network dynamics. Furthermore, the data in an individual network may be very sparse, as users may have limited interactions when a network is just beginning to form. Data sparsity makes models overfit the rare observations and thus have poor generality. Modeling the dynamics of composite networks collectively can lead to more accurate and comprehensive results, and help understand the interactions across different networks.
However, simply considering these nested networks as a single one does not work either, as different networks have different properties. For example, different networks reflect users’ different social interests. Users in different networks may have overlapped but different social circles. Thus, different types of networks have their own growth patterns, e.g., different shrinkage ratios of diam-
Figure 4.2: Illustration of Community Dynamics. Different community relation strength can change over time.

parameters, degree distributions, etc. For example, the average number of users’ neighbors in Microblog network is much higher than instant messaging network (QQ). As shown in Figure 4.1(a)∼(d), communities in Microblog are larger than those in QQ and the evolution process in QQ is more stable. If we simply combine these two networks together, the specific growth patterns can be ignored.

To solve the aforementioned problems, we propose a nonparametric Bayesian model which integrates network evolution over composite social networks directly. On one hand, it utilizes an adaptive and time-dependent prior for each user to denote different users’ latent interests that decide users’ community memberships globally. Since this prior is related to all nested networks and thus it can be considered as a bridge to capture the cross-network dynamics. On the other hand, it introduces temporal network-specific factors to encode growth patterns in each individual network, which adjusts users’ concrete time-evolving community memberships. These two priors are combined together to generate a hybrid prior and build dynamic mixed membership models in each individual network, where users’ community memberships as well as the community relations are exploited to generate links between them.
Figure 4.3: Illustration of Dynamic and Composite Properties. (a) presents the ratio of common links across networks, where the ratio is much higher than that between two random graphs, implying individual networks influence each other. (b) illustrates the correlation between the number of degrees and the number of new links generated by the users.

4.2 Challenges

Our problem setting is novel and challenging in terms of co-evolution. In particular, we enumerate the following challenges for the problem setting.

1. How to make use of the existing correspondences among users from different individual networks, given that such relationships are important and can serve as a bridge across networks.

2. As networks have their own specific growth patterns that do not exist in other networks but each network can influence each other, we should firstly balance these two parts and represent these knowledge appropriately to maximize the knowledge transfer.

3. How to model the data-dependent effectiveness when sharing the data-independent knowledge, as data in different networks may follow different distributions and have different semantic meanings.

To the best of our knowledge, no previous research work has been done to model the cross-network evolution, while these dynamic properties are important to build more comprehensive real-world social applications.
4.3 Modeling Composite Network Dynamics

We propose to capture the dynamics of composite social networks based on MMSB. The proposed model is called ComSoc-IT. Intuitively, users’ community memberships can change over time, as one user may join and leave one community and communities are changing. Similar to ComSoc-MMSB, we also introduce the composite modeling technique to model networks’ common and specific properties separately. To capture the network dynamics, on one hand, we allow users’ latent interests to be dynamic, as users’ preferences are affected by users’ other factors that can change over time as well. As these interests capture networks’ common knowledge, their dynamics can reflect the network co-evolution process. On the other hand, networks’ interest-community mappings vary over time too, as topics in networks are always changing. Thus, the changes of mappings can reflect networks’ specific growth patterns. In addition, as the number of community structures can be different across networks over time, we introduce an infinite modeling process to determine the community numbers in each network at each time stamp automatically.

In the following, we present the details of three extensions: infinite, dynamic and composite modeling. To help understand the motivation of the proposed model, we visualize two sub networks from Tencent. These networks contain users who are 2-hops away from the first author of the paper, as well as their relationships in Tencent’s instant messaging network QQ and Microblog network Weibo, as shown in Figure 4.1.

4.3.1 Model: ComSoc-IT

As MMSB only models static and individual networks with fixed number of communities, as follows, we evolve MMSB step-by-step to model the dynamics of composite networks based on evidences and observations from real-world social networks [132].

Infinite Modeling  In reality, with the evolution of networks, communities can come and go. For example, in social networks, e.g., Facebook, a set of users can form a community when they join the same school and get familiar with each other; in social media, e.g., Twitter, a discussion group may dismiss when a hot topic is out-of-date. An example can be found in Figure 4.1(a) and (b), where the number of communities as well as the community sizes are different over time in Tencent Weibo. Different from previous research works [32, 20], which keep the community structure unchanged in different time stamps, ComSoc-IT allows communities to vary over time and determines the number of communities automatically. Inspired by Chinese Restaurant Process [9] and
its successful applications on topic models [10] and MMSB [116], we introduce a stick-breaking prior on each $\pi_i$ to let $\sum_{k=1}^{\infty} \pi_{ik} = 1$. The construction process is as follows:

$$\pi_{ik} = v_{ik} \Pi_{j=1}^{i-1} (1 - v_{ij})$$

(4.1)

where $v_{ij}$ is the latent factor that needs to be estimated. This process can be understood as follows. Let $\pi_{i1} = v_{i1}$ and $1 - \pi_{i1}$ be the remainder of the stick after chopping off the length $v_{i1}$. To calculate the length $\pi_{i2}$, draw $v_{i2}$ randomly and chop off this fraction of the remainder of the stick, giving $\pi_{i2} = v_{i2}(1 - v_{i2})$. This process is repeated and thus $v_{ik}$ is the fraction to chop off from the stick’s remainder, and $\pi_{ik}$ is the length of the stick that was chopped off. Thus, the probability to generate a community $K + 1$ can be calculated as $1 - \sum_{k=1}^{K} \pi_{ik}$. In addition, $v_{ik}$ evolves over time, and thus a community $k$ can die if $v_{ik}$ becomes too small for each user $u_i$. To avoid overfitting and incorporate domain knowledge, we apply a logistic-normal prior on $v_i$. Formally, we have

$$\epsilon_i \sim N(\mu, \lambda^{-1}I) \quad v_i = \frac{1}{1 + \exp(-\epsilon_i)}$$

(4.2)
where \( \mu \) is the mean of all \( \epsilon_i \) and \( \lambda \) is used to control the precision. As stated below, the logistic-normal prior allows us to incorporate global, network-specific and time-dependent knowledge naturally by adjusting \( \mu \) over time and networks.

**Composite Modeling** Common knowledge can be transferred by overlapping users across individual networks while each individual network contains specific factors. For example, Facebook reflects users’ friendship in daily life, while Twitter is more about users’ interests on daily news. But at the same time, the same users in different individual networks can also have similar neighbors as they have similar interests over different dimensions like friendship, movies, etc. This phenomenon can be formulated as: different individual networks in a composite network may have different community structures, and each user follows community memberships over different networks while keeps some intrinsic features unchanged. One example can be found in Figure 4.1(a) and (c) and Figure 4.3(b), where QQ and Weibo networks have different community structures and users’ community memberships but share many common links. To model these observations, we decompose \( v_i \) into two parts, one representing each user’s latent features and the other reflecting network-specific factors. Let \( x_i \in \mathbb{R}^{1 \times D} \) denote the latent features of user \( u_i \) and \( \omega_d \in \mathbb{R}^{D \times K_d} \) denote the network-specific factors of the \( d \)-th individual network, where \( K_d \) is the corresponding number of communities. \( x_i \) encodes each user’s latent interests while \( \omega_d \) maps users’ latent interests to network-dependent communities. For each user \( u_i \) in \( G_d \), we set \( \mu_{di} = x_i \omega_d \). Thus, cross-network influence can be captured by \( x_i \) while network-dependent properties can be described by \( \omega_d \). To avoid overfitting, we assign Gaussian priors to \( x_i \) and \( \omega_d \), the generation process of \( \pi_i^d \) is as follows:

\[
\begin{align*}
\omega_d &\sim \mathcal{N}(0, \lambda_d^{-1} I) \\
x_i &\sim \mathcal{N}(0, \lambda_u^{-1} I) \\
\epsilon_{di} &\sim \mathcal{N}(x_i \omega_d, \lambda^{-1} I) \\
v_{di} &= \frac{1}{1 + \exp(-\epsilon_{di})} \\
\pi_{dik} &= v_{dik} \prod_{\ell=1}^{K_d-1} (1 - v_{d\ell}) \quad k = 1, \ldots, K_d 
\end{align*}
\]

The number of communities is network-dependent and the communities with the same index in different networks are not necessary identical. This is consistent with the real-world data in Figure 4.1, where the number of communities and community sizes of QQ and Weibo are different.

**Dynamic Modeling** One important property of social networks is its dynamics, where communities can come into being and phase out, users’ community membership can change, and users can form and deform links. Examples can be found in Figure 4.2 shows the different community
structures and their relation strengths over time; Figure 4.1 (a) and (b) present the different community memberships over time; Figure 4.3 (b) shows that the levels of users’ activities to generate links are closely related to their degrees, which evolve over time as well. In mixed membership models, these can be manipulated by changing the community compatibility $B$ and users’ community memberships $\pi$, and varying the number of users’ links in different time stamps respectively. Firstly, with the shifting of users’ interests, the connection between different communities changes over time. Considering that older interactions have relatively smaller impacts, we use exponential decay to modify the priors of $B$. Let $n_{ij}^t$ denote the times that users build links between the $i$-th and the $j$-th communities at time $t$ and $n_{ij}^0$ denote the number of user pairs that do not interact with each other but select the $i$-th and $j$-th communities at time $t$. With a kernel parameter $\kappa$, we have

$$\hat{\gamma}_{ij}^t = \sum_{h=1}^{t-1} \exp \left( \frac{h-t}{\kappa} \right) n_{ij}^h$$

$$\hat{\gamma}_{ij}^0 = \sum_{h=1}^{t-1} \exp \left( \frac{h-t}{\kappa} \right) n_{ij}^h$$

(4.3)

Then the generation of the community compatibility matrix at time $t$ is $B^t \sim Beta(\gamma_0 + \hat{\gamma}_{ij}^0, \gamma_1 + \hat{\gamma}_{ij}^1)$. Note that $\gamma_0$ and $\gamma_1$ here enforce that the connection between new and existing communities can appear as time goes by. Secondly, users’ community memberships shift over time as well. This can be understood from two aspects: (1) users’ latent interests $x_i$ change over time and (2) the mapping from latent interests to communities changes with the evolution of community structures.

Motivated by state-space models for series data, we let

$$\omega^t_i \sim N(\omega^{t-1}_i, \lambda^{-1}_i I)$$

$$x^t_i \sim N(x^{t-1}_i, \lambda^{-1}_u I)$$

(4.4)

That means, users’ latent interests as well as the interest-community mapping at time $t$ are shifted from those in the last time stamp. Finally, users’ activity levels can be very different in different time stamps. According to the study in [54], the probability that one user generates one link with time gap $\delta$ is $\delta^{-\alpha_d} \exp(-\beta_d n_i^d \delta)$, where $n_i^d$ is the number of degrees of user $u_i$ in the network $G_d$, and $\alpha_d$ and $\beta_d$ are two network-dependent parameters. In other words, in unit time, the expected number of interactions for user $u_i$ is $\exp(\beta_d n_i^d)$. Instead of setting the number of interactions explicitly, we introduce a sparsity parameter $\rho_i^d = \frac{\exp(\beta_d n_i^d)}{N_d}$ to characterize the source of interaction, where $N_d$ is the number of users in $G_d$. Then, we down-weight the probability of successful interaction as

$$e^d_{ij} \sim Bern(\rho_i^d \rho_j^d z^T_{ij} B_d z_{ji})$$

(4.5)

Note that the number of node degree varies over time, thus $\rho_i^d$ changes with time as well.
Model Summary  We now put all the pieces together and give the full generative process of ComSoc-IT as follows. The graphical model in one time stamp can be found in Figure 4.4.

- For each user $u_i \in U$: Sample $x_{0i} \sim N(0, \omega^{-1}_u I)$
- For each network $G_0^d \in G$: Sample $\lambda_0^d \sim N(0, \omega^{-1}_d I)$
- Set each $n_{0ij} = 0$ and $n_{0ij}^1 = 0$
- From $t = 1$ to $T$
  - For each network $G_t^d$:
    * Draw a $D \times K_t^d$ feature matrix $\lambda_t^d \sim N(\lambda_t^{t-1}_d, \lambda_t^{-1}_d I)$
    * Draw $B_t^d \sim Beta(\gamma_0 + \tilde{\gamma}_t^0, \gamma_1 + \tilde{\gamma}_t^1)$
  - For each user $u_i \in U$:
    * Draw a $D$ latent feature vector $x_t^i \sim N(x_t^{t-1}, \sigma^2_u I)$
    * Set sparsity parameter $\rho_t^d = \frac{\exp(\beta_d n^d_t)}{N_d}$
    * For each network $G_t^d$:
      
      \[
v^t_{di} = \frac{1}{1+\exp(-e^t_{di})}, e^t_{di} \sim N(x_t^i \lambda_t^d, \lambda^{-1}_d I)
      \]
    * Draw $\pi_t^d$: $\pi_t^d_{ik} = v^t_{dik} \prod_{\ell=1}^{K-1} (1 - v^t_{dil})$
  - For each pair $(u_i, u_j) \in E_t^d$ in $G_t^d$:
    * Draw indicator for $u_i$, $z^t_{dij} \sim Mult(\pi_t^d)$
    * Draw indicator for $u_j$, $z^t_{dji} \sim Mult(\pi_t^d)$
    * Sample the link, $e^t_{dij} \sim Bern(\rho_t^d \rho_t^j z^t_{dij} B_t^d z^t_{dji})$

4.3.2 Inference

Given observed links for some or all node pairs, we employ a Sequential Monte Carlo (SMC) method to draw samples from the latent variables’ posterior distribution and a Maximum a posteriori (MAP) method to estimate the hyperparameters. SMC incrementally runs a fast batch sampling method over the data at epoch $t$ given the state at earlier epochs. As the priors and posteriors are not conjugate, we divide the inference process into three phrases to reduce the computational cost. It firstly samples the community assignments of each user for every links by assuming the community membership $\pi$ is given, then infers users’ membership over communities, and finally estimates the hierarchical priors.
Sampling Community Assignments  Due to the challenge of infinite number of communities, we employ retrospective sampling [84] to approximate the true infinite model. Let $\gamma_{d0} = \gamma_0 + \zeta_{d0}$ and $\gamma_{d1} = \gamma_{d1} + \zeta_{d1}$, and then the pairwise community assignments in the $d$-th network at time $t$ can be written as

$$p(\mathbf{z}_d|\pi_d, \gamma_{d0}, \gamma_{d1}, E_d) \propto p(E_d|\mathbf{z}_d, \gamma_{d0}, \gamma_{d1})p(\mathbf{z}_d|\pi_d)$$

$$= \int p(E_d|\mathbf{z}_d, \mathbf{B}_d)d\mathbf{B}_d p(\mathbf{B}_d|\gamma_{d0}, \gamma_{d1})p(\mathbf{z}_d|\pi_d)$$

$$\propto \prod_{k,k'} B(\gamma_{d0} + \gamma_{d1}, \mathbf{n}_{d,k,k'}) \prod_{u_i} \pi_{dik} \prod_{u_j} \pi_{djk'}$$

where $B(\omega) = \frac{\Gamma(\omega_k)}{\Gamma(\omega_k)}$ and $\mathbf{n}_{d,k,k'}$ denotes the total number of pairwise community assignments between community $k$ and $k'$ at time $t$ in the $d$-th network. As the conjugate property between Beta and Multinomial distributions, $\mathbf{B}_d$ can be marginalized. The posterior of community assignments can be defined as

$$p(\mathbf{z}_{dij} = k, \mathbf{z}_{dji} = k'|\pi_d, \gamma_{d0}, \gamma_{d1}, \mathbf{z}_{d,-(i,j)}, E_d)$$

$$\propto \pi_{dik}^{t_{dij}} \pi_{djk'}^{t_{dji}} \frac{(n_{d,k,k',0} + \gamma_{d0})^{1-y_{dij}} (n_{d,k,k',1} + \gamma_{d1})^{y_{dij}}}{n_{d,k,k',0} + \gamma_{d0} + n_{d,k,k',1} + \gamma_{d1}}$$

where $\mathbf{z}_{d,-(i,j)}$ denotes the set of community assignments without two assignments over the link between $u_i$ and $u_j$, and $n_{d,k,k',y}$ represents the total number of links in type $y$ with $(k, k')$ as the participating communities in the $d$-th network at time $t$. In addition, $y_{dij}$ denotes the sign of the link $e_{dij}$, where $y_{dij} = 1$ represents that $u_i$ and $u_j$ are linked at time $t$ and $y_{dij} = 0$ represents that $u_i$ and $u_j$ will not build a link between each other. Importantly, assuming the number of current communities is $K$, if $1 \leq k \leq K$ then $\pi_{dik} = \pi_{dik}$; otherwise $\pi_{dik} = 1 - \sum_{k=1}^{K} \pi_{dik}$ and $n_{k,k',0}^{t_{d,-(i,j)}} = n_{k,k',1}^{t_{d,-(i,j)}} = 0$. Eq.(4.6) represents the posterior probability of selecting community $k$ if $k \leq K$ but represents the aggregate posterior probability of the infinite “tail” of communities with indexes greater than $K$ if $k > K$.

Sampling the Parameter  Given the community assignments, we employ the Metropolis-Hastings algorithm to sample $v$ for community memberships independently. Given users’ latent features $x$ and network factors $\omega$, to sample a new community activation $v_n^s$ for each user $u_i$ in the $d$-th network, we define the accept ratio of $v_n^s$, $A(v_n^s, v_{di})$, as

$$A(v_n^s, v_{di}) = \prod_{u_j} p(\mathbf{z}_{dij}|v_n^s) p(\mathbf{z}_{dji}|v_n^s) p(\mathbf{z}_{dij}|v_{di})$$

$$= \prod_{k=1}^{K} \frac{\pi_{dik}}{\pi_{dik}} D_{dik}$$

66
where \(D_{dik}\) denotes the total number of indicators attached to user \(u_i\) assigned to community \(k\) in the network \(G_d\). Then, at each time, we sample \(v_{di}^*\) from \(\sim N(v_{di}^*|x_i\omega_d, \lambda^{-1}I)\) and accept it with probability \(\min(1, A(v_{di}^*, v_{di}))\).

**Sampling \(x\) and \(\omega\)** Given \(v\), for each time stamp \(t\), we can infer the hierarchical priors: \(\omega_d^t\) for each network and \(x_i^t\) for each user \(u_i\). To find the optimal values, we first define the union distribution over \(v\) and priors as

\[
p(v', \omega', x) = \prod_{d=1}^L \prod_{i=1}^N N(v_{di}'|x_i'\omega_d', \lambda^{-1}I) \prod_{i=1}^N N(x_i'|x_i^{-1}, \lambda_u^{-1}I) \prod_{d=1}^L N(\omega_d'|\omega_d^{-1}, \lambda_d^{-1}I)
\]

where \(x_i^0 = 0\) and \(\omega_d^0 = 0\). By calculating the derivatives of the log of likelihood with respect to \(\omega_d^t\) and \(x^t\) and set them as zero, we obtain the sampling equations as

\[
\omega_d^t \sim N((\lambda_d I + x'^T x)^{-1} x'^T v' + \omega_d^{-1}, \lambda_d^{-1}I)
\]

\[
x^t \sim N(v'^T \omega_d^t(\lambda_u I + \sum_d \omega_d^T \omega_d)^{-1} + x'^{-1}, \lambda_u^{-1}I)
\]

where \(x^t\) is a \(N \times D\) matrix, each row of which is \(x_i^t\).

**Parameter Estimation** As follows, we propose a maximum a posteriori (MAP) method to estimate the hyperparameters, including \(\lambda\), \(\lambda_u\) and \(\lambda_d\). To avoid overfitting, we place Gamma priors so that their prior distributions are conjugate with the likelihoods. For \(\lambda\), we have

\[
p(\lambda|v, x, \omega, a, b) \propto \text{Gam}(\lambda|a, b) \prod_{t=1}^T \prod_{d=1}^L \prod_{i=1}^N N(v_{di}'|x_i'\omega_d', \lambda^{-1}I) \propto \text{Gam}(\lambda|\frac{1}{2}NT\ell K + a, \frac{1}{2} \sum_{k,t,d,i} (v_{dik} - [x_i'\omega_d'])^2 + b)
\]

Similarly, we obtain the sampling equations for \(\lambda_u\) and \(\lambda_d\):

\[
p(\lambda_u|x, a_u, b_u) \propto \text{Gam}(\lambda_u|\frac{1}{2}NTD + a_u, \frac{1}{2} \sum_{f,t,i} x_{if}^2 + b_u)
\]

\[
p(\lambda_d|\omega, a_d, b_d) \propto \text{Gam}(\lambda_d|\frac{1}{2}T\ell K + a_d, \frac{1}{2} \sum_{k,t,d} \omega_{dk}^2 + b_d)
\]
Algorithm  We put all the components above into an SMC framework. Firstly, we set priors as arbitrary values, then initialize the latent variables and finally sample the community assignments. Consequently, it alternatively updates community assignments, community memberships, priors and hyperparameters in each round of inference:

- Update community assignments using Eq.(4.6)
- Update community activations $v$ using Eq.(4.7)
- Update network-specific priors $\omega$ using Eq.(4.9)
- Update user priors $x$ using Eq.(4.10)
- Update hyperparameters using Eq.(4.11), (4.12) and (4.13)

This process is repeated until convergence. Besides, the sparsity ratio $\rho$ can be estimated independently by maximizing likelihood [54]. To avoid overfitting and reduce the computational cost, we update latent variables periodically. Specifically, we update $v$ every 5 iterations, both $\lambda$ and $x$ every 10 iterations, and hyperparameters every 20 iterations. In addition, as in most real-world datasets, we observe only positive links. For each user, we randomly sample other users who are more than two-hops away and build negative links to the current user. The number of negative links is kept in the same order to that of positive links.

Time Complexity  At each iteration, SMC needs to look up all $M$ links in nested networks to pick community assignments and then update $v$ with time $O(T\ell N K D)$ where $N$ is the number of users. Metropolis-Hastings sampler spends $O(T N D)$ on updating $x$ and $O(T \ell K)$ on updating $\omega$. Consequently, the updating of hyperparameters takes $O(T \ell N K)$. Typically, $T \ell K$, $T N D$ and $T \ell N K$ are much smaller than $T \ell N K D$. Thus, with $I$ iterations, the whole complexity is $O(I(MK + T \ell N K D))$ which linearly increases with the number of links $M$ and users $N$. Following the similar idea in [130], that partitions data into multiple machines for parallel computing and then combines pieces of results, the inference process can be implemented on proposed parallel framework in Chapter 7 straight forward to cope with large datasets efficiently.

4.4 Experiments

The tasks include (1) link prediction, that predicts who will interact with each other in a given time stamp, and (2) macro-evolution, which predicts changes of networks’ statistics, e.g., clustering coefficients and degree distribution, etc. We adopt Mean Average Precision (MAP) as the evaluation criterion. It measures how well the algorithm ranks new links above non-existing links. We set $D = 5$, $\kappa = 1$ and other hyperparameters as 1.0. The number of iterations is 1000. In the link
prediction task, we compare ComSoc-IT with Mixed Membership Stochastic Blockmodels (MMSB) [3], dynamic MMSB (dMMSB) [32], Nonparametric Metadata Dependent Relational Model (NMDR) [48], dynamic Infinite Relational Model (dIRM) [42] and Tensor Factorization (TF) [34]. MMSB is the basic baseline, dMMSB is a dynamic extension of MMSB, NMDR is a nonparametric extension of MMSB, dIRM is a dynamic and nonparametric extension of MMSB, and TF models multiple relations and time factors together but it does not model the network differences and in it the number of communities is fixed. For MMSB, dMMSB and TF, we set the number of communities $K$ as 50 while NMDR, dIRM and ComSoc-IT can determine $K$ automatically. We set other parameters as the default values in original papers. For macro-evolution, we introduce the Microscopic Evolution (ME) model [54] as the baseline to show how well ComSoc-IT approximates the real evolution process. In the link prediction task, the networks at the previous $T−1$ time stamps are considered as the model input and the output is the prediction at time $T$. To simulate the evolution process, we consider the network series $\{G_t\}_{t=1}^{T/2}$ and evolve it from $t = T/2, \ldots, T$.

### 4.4.1 Interaction/Friendship Prediction

Table 4.1 summarizes the MAP of all baselines in the link prediction task. ComSoc-IT consistently outperforms other baselines on MAP in all networks except Tencent’s Microblog against TF. The MAP of ComSoc-IT is at least 0.02 higher than MMSB. If we overlook the model differences, ComSoc-IT on average achieves 0.042, 0.028, 0.027, 0.026 and 0.021 higher MAP as compared to MMSB, dMMSB, NMDR, dIRM and TF respectively. The better performance of ComSoc-IT over baselines with one individual network can be ascribed to the fact that, ComSoc-IT considers more knowledge from auxiliary networks and captures more aspects of users’ interests. Due to the data sparseness, not all users’ interests can be reflected by one individual network. Thus some knowledge will be missed if only one network is considered. On the other hand, although models with simply combined network, such as MMSB-C, exploit knowledge from multiple networks to improve accuracies, they do not consider network differences, i.e., users actually make different friends in different networks. Thus, they may introduce unnecessary regularization from unrelated networks. However, ComSoc-IT adaptively captures the cross-network influence and keeps the network-specific growth patterns, hence solves the above problems.

We have other interesting discoveries. First, comparing the performance of other baselines and MMSB, infinite and dynamic modeling can help improve the prediction precision. Second, due to the network differences, combining knowledge may not always be helpful. For example, the performance on the combined networks degrades as the trust and distrust relations are different across
networks, thus, simply combined different networks bring negative impacts. Thirdly, the prediction precisions in Github and Stackoverflow are higher than in other networks. The reason is that, users in Github and Stackoverflow are programmers, who may have more consistent interaction patterns. Last but not the least, network influence is not symmetrical. For example, in Xiaonei collection, the improvement in friend application network is higher than that in other networks, as friend applications are much fewer than other interactions but others are highly dependent on the actions of friend application.

### 4.4.2 Network Evolution

Figure 4.5 and Figure 4.6 summarizes the evolution results of ME and ComSoc-IT in Tencent and Epinion networks on degree distribution and clustering coefficient respectively. Both networks are relational networks which are suitable for the settings of ME, and thus the comparison is fair. Overall, the performance of ME and ComSoc-IT are close on the evolution of degree distribution in the last time stamp. They approximate the true distribution of networks well in both collections. Nonetheless, on the Tencent dataset, ComSoc-IT performs slightly better than ME. The reason is
that ME models network evolution anonymously, that ignores users’ personalities. When a network is large, the estimated distribution from a set of anonymous users is stable. However, when data is insufficient, estimation may overfit the rare observations. On the other hand, ComSoc-IT models users’ activities in person and hence provides more accurate modeling. In addition, ComSoc-IT collectively exploits knowledge from two networks, where knowledge in different networks can regularize the evolution process in each individual network and thus alleviate the overfitting issue. The performances of ME and ComSoc-IT on clustering coefficient are similar as well. But for long-tail users, e.g., whose number of neighbors is less than 10, the estimation of ComSoc-IT is more accurate than ME. That ascribes to the fact that ComSoc-IT considers more knowledge from multiple networks.

4.4.3 Performance Analysis

We perform extended experiments on the Tencent dataset to answer three questions: (1). does ComSoc-IT capture the network differences and reflect specific growth patterns? (2). how do the model parameters affect the performance of ComSoc-IT? (3). what is the effectiveness of
corresponding ratio among individual networks? (4). what is the effectiveness of each component in ComSoc-IT?

We plot the community relation matrices of QQ and Weibo in Figure 4.7(a), where nodes are communities and the thickness of edges represent the relational strength between communities. Obviously, the number of communities and community relations are different (43 in QQ vs. 33 in Weibo). Communities in Weibo are more centralized while those in QQ are more diverse. This is consistent with real applications, as people on Weibo follow others based on interests while people on QQ may interact with only close friends. This suggests that different networks have different community structures and ComSoc-IT can capture this difference.

We analyze the effects to change the number of features $D$ (5 to 100). The results are illustrated in Figure 4.7(b). The performance of ComSoc-IT is not sensitive to the number of $D$ but decreases with very large $D$. The reason is that $D$ denotes the representation complexity of each user. Thus, if $D$ exceeds a threshold (e.g., 50), the model is too complex and results in overfitting. At this point, it is less helpful to improve the model performance by increasing $D$. In practice, $D$ can be tuned through cross-validation techniques. For $\kappa$, we vary it from 1 to 10, the results are shown in Figure 4.7(c). The performance of ComSoc-IT drops down slightly with larger $\kappa$ but it is not
sensitive nonetheless.

As many people may only use parts of online services, i.e., they may exist in one individual network but not in all networks, we test the effect of the correspondence ratio between different individual social networks, e.g., the number of users who exist in both networks. Figure 4.7(d) presents the results with different ratios between the QQ and MB networks. ComSoc-IT’s performance becomes worse if fewer correspondences are provided across networks. In addition, it outperforms MMSB consistently if there are corresponding users, implying that ComSoc-IT successfully uses the overlapping users as bridges to capture cross-network influences.

To test the effectiveness of each component, i.e., composite, infinite and dynamic in ComSoc-IT, we perform experiments on the Tencent dataset by removing specific components in ComSoc-IT and produce three baselines. Com considers only composite network knowledge but fixes the number of communities and ignores the temporal information, ICom is the same as ComSoc-IT but ignores the temporal information and TCom fixes the number of communities. As shown in Figure 4.8, the composite modeling contributes most improvements while the infinite and dynamic extensions improve the model performance further.

### 4.4.4 Efficiency Analysis

As analyzed in the end of Section 2.6, the computational time of ComSoc-IT increases linearly with the number of interactions between users. In addition, the computational time is closely related to the convergence property of the inference algorithm. We evaluate these empirically as shown in Figure 4.9. Figure 4.9(a) illustrates the computational time of MMSB-C, dMMSB-C, NMDR-C, dIRM-C and ComSoc-IT on the Tencent dataset, with different ratio of links. We observe that the computational time increases linearly with larger data size. For our experimental setting, every
round of inference takes about 45 seconds in our computer, of which memory is 16G and CPU is 3.2Gz. ComSoc-IT not only has better prediction performance as shown in Table 4.1, but also has similar time cost with dIRM-C and dMMSB-C. Figure 4.9(b) shows the convergence property of ComSoc-IT. Clearly, ComSoc-IT becomes convergent after about 400 iterations. Combining with the time cost of each round, the inference algorithm of ComSoc-IT takes about five hours to build a model.

### 4.5 Summary

In this chapter, we studied a new problem on dynamics analysis of composite social networks. We defined a composite social network as a set of nested individual networks, where users and links in different individual networks overlap. The dynamical process of each individual network can influence each other while keeping specific evolution patterns. To model this co-evolution process, we proposed a nonparametric Bayesian model, ComSoc-IT, by capturing cross-network influences adaptively. The main idea is to model users’ latent interests, which carry the common knowledge embedded in multiple networks and use network-specific factors to describe the network dependent growth patterns on community structures. These two kinds of knowledge are then encoded as a hybrid prior for a dynamic mixed membership model to generate the links between users in each time stamp. To allow communities vary over time and networks, we construct nonparametric models. Unlike prior works, the proposed model can capture the microscopic and macroscopic aspects of network dynamics, including the links of each specific user and the global changes of networks properties. The proposed model is flexible in that, it can be extended to any number of networks and the number of communities can be determined automatically. We conducted large-scale empirical studies on eight real-world network collections, where ComSoc-IT outperforms...
several state-of-the-art baselines on link prediction by as high as 0.11 on MAP efficiently and estimates the evolution of degree distribution and clustering coefficient accurately.

**Applicability Discussion** The assumption of ComSoc-IT is similar to the one of ComSoc-MMSB, where we assume that users’ social interests and behaviors are consistent. That means most users will do similar kinds of behaviors in one network.

**Future Works** The proposed model requires the users across networks to be identical. We propose to relax this restriction in the future. Besides modeling the network dynamics and predicting users’ behaviors [130], the rich data embedded in composite social networks can benefit many other applications. Firstly, users’ characteristics and their interactions can be exploited to build an accurate user distance measure. In addition, by considering multiple networks together, common patterns can be extracted from users’ behaviors/interactions and these patterns can be used to infer their characteristics.
Table 4.1: Performance Comparisons on MAP (ComSoc-IT)

| Networks | QQ MB | Tencent | Tencent QQ | Tencent QQ MB | Epinion Trust | Epinion Distrust | Facebook | Facebook Link | Facebook Wall | Renren Record | Renren Browse | Renren Chat | Renren Friend | Twitter Forward | Twitter Mention | Weibo Forward | Weibo Mention | Github Follow | Github Col. | SO Answer | SO Comment | SO Vote |
|----------|-------|---------|------------|--------------|---------------|----------------|-------------|---------------|---------------|---------------|---------------|------------|-------------|---------------|----------------|---------------|-------------|-------------|-------------|------------|----------|-----------|---------|
|          |       |         |             |               |               |                |             |               |               |               |               |            |             |               |                |               |             |             |            |           |           |           |         |
| MMSB     | 0.5097 | 0.5034  | 0.5135      | 0.5211        | 0.5131        | 0.5055         | 0.5132      | 0.5127        | 0.5241        | 0.5599        | 0.5251       | 0.2549     | 0.7375      | 0.7116        |                |               |             |             |            |           |           |         |
| MMSB-C   | 0.2154 | 0.2151  | 0.2255      | 0.2397        | 0.2293        | 0.2239         | 0.2339      | 0.2337        | 0.2541        | 0.2549        | 0.2549       | 0.2549     | 0.2549      | 0.2549         |                |               |             |             |            |           |           |         |
| dMMSB    | 0.7235 | 0.6937  | 0.7282      | 0.7112        | 0.7251        | 0.7010         | 0.7205      | 0.7005        | 0.7300        | 0.7375        | 0.7375       | 0.7375     | 0.7375      | 0.7375         |                |               |             |             |            |           |           |         |
| dMMSB-C  | 0.7020 | 0.6738  | 0.7151      | 0.6864        | 0.6996        | 0.6829         | 0.6945      | 0.6937        | 0.7001        | 0.7116        | 0.7116       | 0.7116     | 0.7116      | 0.7116         |                |               |             |             |            |           |           |         |
| NMDR     |        |         |             |               |               |                |             |               |               |               |               |            | 0.7469      |               |                |               |             |             |             |            |           |           |         |
| NMDR-C   |        |         |             |               |               |                |             |               |               |               |               |            | 0.7029      |               |                |               |             |             |             |            |           |           |         |
| dIRM     |        |         |             |               |               |                |             |               |               |               |               |            | 0.7469      |               |                |               |             |             |             |            |           |           |         |
| dIRM-C   |        |         |             |               |               |                |             |               |               |               |               |            | 0.7029      |               |                |               |             |             |             |            |           |           |         |
| TF       |        |         |             |               |               |                |             |               |               |               |               |            |            |               |                |               |             |             |             |            |           |           |         |
| ComSoc-IT|        |         |             |               |               |                |             |               |               |               |               |            |            |               |                |               |             |             |             |            |           |           |         |
CHAPTER 5

USER BEHAVIOR PREDICTION ACROSS SOCIAL MEDIAS

5.1 Introduction

An important challenge for user behavior modeling in social media is how to push the right information to the right users at the right time. Accurate user-behavior models enable us to predict users’ actions and intents based on data collected from online product purchasing, webpage browsing and advertisement clicking history, etc. Different approaches of user-behavior modeling, such as collaborative filtering, have been proposed [95]. A major challenge to use these methods lies in the “sparse” nature of historical data, whereby users have interactions with only few other users and items; users may click on only one or two advertisements and listen to a few songs during a time period. For example, the dataset from Tencent has more than 99.9% empty entries in the user-item interaction matrix. The problem with sparse data makes existing methods overfit, and thus fails to model users’ behaviors accurately. We observe that many people are members of several social networks in the same time, such as Facebook, Twitter and Tencent’s QQ. Importantly, their behaviors and interests in different networks influence one another. This gives us an opportunity to leverage the knowledge of user behaviors in different networks in order to alleviate the data sparsity problem, and enhance the predictive performance of user modeling.

While interactions in a single social network can be sparse, there is an opportunity to exploit data in multiple networks by considering overlapping users as bridges. Several previous research works have considered combining multiple networks to form a larger network, in which a holistic user model could ideally be built. For example, Pan et al. [80] has proposed to combine multiple user relationships to recommend Apps to users, where the sub networks are treated uniformly, such that each user-user relationship holds the same importance. However, we observe that in a composite network, the solution by treating each relationship uniformly may likely result in poor prediction performance, due to the following two reasons. First, different networks have their own properties, such as density, degree distributions, clustering coefficient or diameters. If we combine dense and sparse networks together, the information in the sparse network will be overwhelmed inside the dense one. Second, users play different roles and get different levels of influence from their neigh-
bors in individual networks. Some users may prefer Facebook over Twitter for one application, while others may use Twitter more frequently instead. Hence, neighbors in different networks may have different impacts on a given user. To illustrate the idea, a synthetic example is drawn in Figure 5.1, where there are four users (John, Sara, Bob, Alice) and their preferences on four hobbies: watching movie, listening to music, reading book and playing games. In this figure, “1” represents “like” and “−1” denotes “dislike”. Based on the interaction matrix in Figure 5.1(a), we compute the users’ similarities with cosine distance [95], as shown in Figure 5.1(b). Clearly, John is similar to Bob, Bob is similar to John and Sara, and Sara is similar to both Bob and Alice. Figure 5.1(c) summarizes their relations under a given relationship, such as the friendship on Facebook while Figure 5.1(d) shows another one, such as the following relationship on Twitter. Clearly, both of the friendship and following relationships can only partially reflect the similarities in interests. If we simply merge them together, as shown in Figure 5.1(e), the naively combined network does not reflect the correct similarity among them. For example, John and Alice are dissimilar, but there is still a link between them.

We present a hierarchical Bayesian model to cope with the above challenges. It transfers knowledge of behavior models across individual networks in a composite network, specifically adapted and parameterized for different users. The common users across individual networks form bridges to enable the transfer. To uncover the common knowledge across different individual networks, we apply transfer learning techniques [79] to exploit both the topological and topical knowledge in social networks. The basic idea is to integrate relationships adaptively, where different users will be influenced by their neighbors in each individual network with different levels of importance using personalized weights.

5.2 Challenges

To address the adaptive knowledge transfer setting as shown above, we need to decide which network is useful for a given user. Specially, we have to answer three fundamental questions:

1. What kinds of knowledge in social network are useful to help predict users’ behaviors or profiles;

2. How to model the social relations and users’ behavior logs in a principled way;

3. When to regularize the user behavior modeling with social relations, such long-tail users and head users.
Figure 5.1: Synthetic example on how to exploit Composite Social Network adaptively. (a) the complete user-item interaction matrix - most items are empty and need to be modeled by recommendation system; (b) the user similarity based on user-item interactions; (c) “Facebook” relationship among them; (d) “Following” relationship on Twitter; (e) the naive combination of (c) and (d); (f) the adaptive combination reflecting the correct user similarity in (b).

As these three problems correlate with each other, we propose to solve them together in a unified framework. Although the previous work exploited the knowledge from multiple networks to help recommend Apps [80], it use all knowledge from social networks and apply them anyway without consider the network and user differences.
We propose a composite social topic model (ComSoc-Adap) for selecting individual networks in a composite social network, in order to adaptively transfer the relational knowledge. Intuitively, we assume that each user’s behaviors are generated from some distributions based on users’ interests or topics. As users’ interests are also reflected by their social relations, modeling users’ social relations and behaviors together can make the behavior modeling be regulated by the social relational knowledge. In addition, considering that different networks may have different impacts on different users, the key point of our intuition is that, the regularization from different networks is adjusted adaptively depending on the user. This is consistent with the observation that, someone may use Facebook more often than Twitter and hence the neighbors on the Facebook shall have higher level of influence for the given user, while others may be influenced more by their followers on Twitter, as they prefer Twitter over Facebook.
5.3.1 Model: ComSoc-Adap

The complete generating process of ComSoc-Adap is as follows. Firstly, ComSoc-Adap models users’ interests as topics and then each user is represented as a distribution over topics, $\Pr(t|u)$. In addition, ComSoc-Adap adds one constraint on $\Pr(t|u)$ by enforcing that users’ distributions over topics can be used to generate links between users. Intuitively, common interests tend to create relationships among people. For each user in each individual network $G_d$, ComSoc-Adap assumes $\Pr(t|u)$, a Multinomial distribution, is generated by a Dirichlet distribution with a prior $\alpha_d$. Subsequently, a topic indicator is drawn from this distribution for each user. Then every link in each network is generated by a Bernoulli distribution, of which the prior is determined by the topic indicators of the users and a topic compatibility matrix $B^d$ related to $G_d$, where $B^d$ represents the topic connections. After generating the user-topic distribution $\Pr(t|u)$ for each user in every social networks, to transfer knowledge from social relationships adaptively, we introduce a term of users’ distribution over networks, $\Pr(G_d|u)$, in order to represent the probability of how much a user is influenced by a given network. ComSoc-Adap models this user-network distribution in a hierarchical manner. At first, for each user $u_i$, a network proportion $\rho_i$ is drawn from a Dirichlet distribution with a prior $\nu$. Then, for every interaction of a given user, a social network is drawn from a Multinomial distribution according to the network proportion. The user-topic distribution of the selected network, $\Pr(t|u)$, is picked to generate a topic assignment. Finally, ComSoc-Adap generates each interaction according to the item-topic distribution, $\Pr(v|t)$, over the chosen topic. We formulate this process as follows.

- For every user $u_i$ in each network $G_d$:
  - Draw topic proportion $\theta_{id} \sim \text{Dir}(\alpha_d)$

- For every link of $(u_i, u_j)$ in each network $G_d$:
  - Draw topic indicator $z_{ij} \sim \text{Mult}(\theta_{id})$, $z_{ji} \sim \text{Mult}(\theta_{jd})$
  - Draw binary link indicator $e_{ij}^d \sim \text{Bern}(z_{ij}B_d z_{ji})$

- For each user $u_i$:
  - Draw network proportion $\rho_i \sim \text{Dir}(\nu)$
  - For each interaction $a_{i,k} \in A$ with item $v_k$:
    * Draw a network indicator $d \sim \text{Mult}(\rho_i)$
    * Draw assignment $z_{i,k} \sim \text{Mult}(\theta_{id})$
    * Draw interaction $a_{i,k} \sim \text{Mult}(\beta_{z_{i,k}})$
We observe that, each user $u_i$ has her/his own network proportion $\rho_i$. This means that we choose different individual networks to transfer knowledge adaptively for different users. The graphical representation can be found in Figure 5.2.

5.3.2 Inference

The basic idea is to update topic assignments from the posterior distribution. To guarantee sampling efficiency, the posterior distribution is represented in collapsed format [88]. We first define a number of statistics. Let $n_{t,v}$ denote the number of the topic $t$ related to an item, and $n^s_{u,t}$ denote the number of the topic $t$ assigned to a user in the $s$-th network. Formally, for each topic assignment $z_{i,k}$,

$$n_{t,v} = \sum_{i,k} \{z_{i,k} = t; v_k = v\}; n^s_{u,t} = \sum_k \{z^s_{i,k} = t\}$$ \hspace{1cm} (5.1)

In addition, we denote $n_t = \sum_v n_{t,v}$, where $n_t$ stores the number of times that items are assigned to the topic $t$, and $n_u$, which maintains the number of items interacted with the user $u$. Lastly, we define another statistic $n_{u,d}$, which is the number of times that a user $u$ utilizes subnetwork $G_d$ in order to regularize her/his distribution over topics.

We express the full joint probability of the interaction data in $A$ with network indicators as:

$$\text{Pr}(A, z, s | \alpha, \beta, \nu) = \left[ \prod_{i=1}^N \frac{\prod_{j=1}^{K} \Gamma(\alpha_s + n_{u=i,t=j})}{\prod_{j=1}^{K} \Gamma(\alpha_s)} \right] \times \left[ \prod_{i=1}^N \frac{\prod_{j=1}^{V} \Gamma(\beta_t + n_{t=i,v=j})}{\prod_{j=1}^{V} \Gamma(\beta)} \right] \times \left[ \prod_{i=1}^N \frac{\prod_{j=1}^{\ell} \Gamma(\nu_{\ell} + n_{u=i,d=j})}{\prod_{j=1}^{\ell} \Gamma(\nu)} \right]$$

We notice that, different from the inference in LDA [88], the above equation has one additional term to model, which network to be selected, in order to regularize users’ distributions over topics. Following the inference in [88], let $z_{-(i,k)}$ denote the topic assignments without the current topic on $a_{i,k}$, and then we rewrite the above equation to obtain the following un-normalized probability, in order to resample the topic $z_{i,k}$ for $a_{i,k}$ between user $u_i$ and item $v_k$:

$$\text{Pr}(z_{i,k} = t, d = s | A, z_{-(i,k)}) \propto (n^s_{i,t,-(i,k)} + \alpha_s) (n_{t,s,-(i,k)} + \nu) \frac{n_{t,-(i,k)} + \beta}{n_{t,-(i,k)} + T\beta}$$ \hspace{1cm} (5.2)

The resulting equation is used to update the topic assignments for user-item interactions. In addition, we need to sample users’ pairwise topic assignments for user-user relationships in each
subnetwork $G_s$, which regularizes the user-topic distributions for user-item interactions. In each network, we first define the full joint probability as:

$$
\Pr(\tilde{z}, E_s | B^s, \alpha_s) = \left[ \prod_{i=1}^{N} \prod_{j=1}^{K_s} \frac{\Gamma(\alpha_s + n_{u=i,t=j})}{\alpha_s K + n_{u=i}} \right] \times \left[ \prod_{i=1}^{N} \prod_{j=1}^{K_s} \frac{\Gamma(\alpha_s + n_{u=i,t=j})}{\alpha_s K + n_{u=i}} \right]
$$

Following the same inference process, the un-normalized probability for resampling the pairwise topic for user relationship between $u_i$ and $u_j$ is

$$
\Pr(\tilde{z}_{i,j} = t_1, \tilde{z}_{j,i} = t_2 | \alpha_s, B^s, E_s, \tilde{z}_{-(i,j)}) \propto \prod_{t_1, t_2} \Gamma(n_{i,t_1} + \alpha_s) \Gamma(n_{j,t_2} + \alpha_s) \propto (n_{i,t_1,-(i,j)} + \alpha_s)(n_{j,t_2,-(i,j)} + \alpha_s)
$$

This equation is used to update the pairwise topic assignments for user-user relations. In summary, in each iteration of Gibbs sampling, we update the topic assignments for user-item interactions and user-user relations in each sub network alternatively using Eq.(5.2) and Eq.(5.3) respectively until convergence. We revisit the synthetic example in Figure 5.1. After learning the model, to predict the preference of each user, it first selects an appropriate social relationship. For example, Bob adopts the Facebook relationship in Figure 5.1(c) while Alice adopts the “Following” relationship in Figure 5.1(d). Then, an adaptively combined network is constructed, as shown in Figure 5.1(f). The neighbors’ knowledge in this network is exploited to help pick a topic. Finally an item is selected to recommend to the given user according to the item proportion of this topic.

The inference framework of ComSoc-Adap can be found in Algorithm 2. At the beginning, it randomly picks topics for user-item interactions and user-user relationships. Afterwards, in each iteration, Eq.(5.2) and Eq.(5.3) are utilized to update the topic assignments from user-item interactions and user-user relations in each social network. The iteration is repeated until convergence. In addition, a burn-in process is introduced in the first several iterations to remove unreliable sampling results. The parameters $\theta_s, \phi$ and $\psi$ are used to predict user behaviors. We analyze the time complexity as follows. Suppose the process needs $I$ iterations to reach convergence. In each iteration, it requires to go through all user-item pairs ($Q$) in the interaction network and user-user pairs ($M_i$) in each sub social network. In addition, for each pair, it requires $O(K)$ operations to compute the posterior distribution and sample topics, and constant cost to update the statistics. At last, it needs $O(N + T)$ operations to generate topic proportions. Thus, the whole time complexity is $O(I(Q + \sum_{i=1}^{I} M_i)K + N + T) = O(I(Q + \sum_{i=1}^{I} M_i)K)$.
Algorithm 2 Inference of ComSoc-Adap

1: **Input**: Composite network: $G$, Interactions: $A$, # of iterations $I$, # of burn-in, $I_b$, Priors: $\alpha$, $\beta$ and $\nu$
2: **Output**: User-topic distribution $\theta_s$ for each sub network $G_s$, topic-item distribution $\phi$ and user-network distribution $\psi$
3: Assign topics for user-item pairs randomly
4: Assign topics for user-user pairs in each sub network
5: Initialize the user-network assignments randomly
6: Initialize the $\theta_d$, $\phi$ and $\psi$ with zeros
7: for $i = 1$ to $I$ do
8: Update user-item topic assignments using Eq.(5.2)
9: Update user-user topic assignments using Eq.(5.3)
10: IF $I > I_b$ Update $\theta_s$, $\phi$, $\psi$ as
11: \[
\theta_{s,i,j}^+ = \frac{1}{I - I_b} \frac{\alpha_s + n_{s,i,t} + \lambda_{s,j}}{\alpha_s K + n_{s,i,t}}
\]
12: \[
\phi_{i,j}^+ = \frac{1}{I - I_b} \frac{\beta + n_{i,j,v}}{\beta V + n_{i,j,v}}
\]
13: \[
\psi_{i,j}^+ = \frac{1}{I - I_b} \frac{\nu_{s} + n_{i,s,j}}{\nu_{s} + n_{i,s,j}}
\]
14: end for
15: Return $\theta_s$, $\phi$ and $\psi$

### 5.4 Experiments

We evaluate the proposed methods on two data collections, one is from Douban\(^1\), and the other is from Tencent; both contain composite social networks and rich user behavior information. We compare the proposed approach with three state-of-the-art methods: latent Dirichlet allocation (L-DA) [11], LinkLDA [27] and relational topic models (RTM) [18]. For LinkLDA and RTM, we implement two variations: one is to build models with only one single social network, and the other is to build models with a naive combined social network, of which the edge set is the union of all edges in sub networks. For the model parameters, we set the priors for topic models as $\alpha = 0.5$, $\beta = 50/K$ and $\nu = 0.5$. In addition, we set the number of topics as $K = 50$. We will study the effectiveness of this parameter later. The results are evaluated based on two criteria: perplexity (perp) and mean average precision (MAP) on the hold-out set $\mathcal{T}$.

\[
\text{perplexity} = \exp \left\{ - \frac{\sum_{(u,v) \in \mathcal{T}} \log \Pr(u,v)}{|\mathcal{T}|} \right\} \tag{5.4}
\]

\[
\text{MAP} = \frac{1}{N} \sum_{u \in U} \sum_{(u,v) \in \mathcal{T}_u} \frac{1}{|\mathcal{T}_u|} \text{pre}_r \tag{5.5}
\]

where $\text{pre}_r$ represents the precision on top-$r$ predictions. Behavior records in $\mathcal{T}$ are represented as a list of user-item pairs. For example, for the music listening prediction, one user-item pair represents one user will listen to the given music. Then the task is to predict if such behaviors happen or not, in other words, if the corresponding user-item pairs exist in $\mathcal{T}$ or not. Generally, perplexity measures how precisely the algorithm generates the behaviors in the hold-out set, and

\(^1\)http://www.douban.com
Table 5.1: Summary of Data Characteristics

<table>
<thead>
<tr>
<th>Collections</th>
<th>Tencent</th>
<th>Douban</th>
<th>Collections</th>
<th>Tencent</th>
<th>Douban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networks</td>
<td>QQ</td>
<td>Microblog</td>
<td>Online</td>
<td>Offline</td>
<td>Behaviors</td>
</tr>
<tr>
<td>#User</td>
<td>∼1M</td>
<td>∼50K</td>
<td>∼80M</td>
<td>∼32M</td>
<td>∼5M</td>
</tr>
<tr>
<td>#Edge</td>
<td>∼80M</td>
<td>∼32M</td>
<td>∼5M</td>
<td>∼4M</td>
<td>#Interaction</td>
</tr>
</tbody>
</table>

MAP measures how well the algorithms rank the behaviors in the hold-out set above non-existing behaviors. In addition, to evaluate the significance in improvement of the proposed model, we randomly split the hold-out set into 10 subsets and then calculate the error bar. We notice that, the observed behaviors are implicit, where we only know which items are interacted by a given user.

5.4.1 Dataset Description

The Douban collection contains two sub networks. One is the online contact network, representing who pay attention to whom, and the other is the offline physical network, indicating who are familiar with whom in the real world. Since the website does not provide APIs to collect the data of offline relations, we construct the offline network using users’ co-occurrence in social gatherings, where people physically meet. If two users take part in the same gatherings more than 20 times, we consider that they know each other. Finally, the task is to predict users’ behaviors on movie, music and book domains using their historical behavior logs, as well as the information in the online and offline networks. This dataset can be downloaded.

The Tencent collection contains two sub networks. One is QQ, which contains about 1 billion users and 80 billions links, making it the biggest instant messaging network in the world, and the other is Tencent’s Microblog network, which contains more than 200 millions users, making it one of the biggest Microblog networks in China. We introduce two subtasks: the first one is to predict which songs users will collect in the future, which can be applied to radio applications, and the second one is to predict users’ profiles, which is crucial to display online advertisements. To generate users’ profiles, we recorded the words from three big Web sites: QQ.com, one of the biggest portals in China, qzone, the biggest social blogging service in China and wenwen, one of

---

2http://www.cse.ust.hk/TL/dataset/Douban-50000.zip
4http://t.qq.com/?lang=en_US
5http://fm.qq.com/
6http://qzone.qq.com
7http://wenwen.soso.com/
Table 5.2: Performance Comparisons on Douban Collection (ComSoc-Adap)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Movie</th>
<th>Music</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LDA</td>
<td>LinkLDA-On</td>
<td>LinkLDA-Off</td>
</tr>
<tr>
<td>Movie</td>
<td>perp</td>
<td>7140.3</td>
<td>7311.1</td>
</tr>
<tr>
<td></td>
<td>MAP</td>
<td>0.8407</td>
<td>0.8407</td>
</tr>
<tr>
<td>Music</td>
<td>perp</td>
<td>43676.8</td>
<td>43506.4</td>
</tr>
<tr>
<td></td>
<td>MAP</td>
<td>0.7492</td>
<td>0.7492</td>
</tr>
<tr>
<td>Book</td>
<td>perp</td>
<td>47541.0</td>
<td>54646.1</td>
</tr>
<tr>
<td></td>
<td>MAP</td>
<td>0.7084</td>
<td>0.7088</td>
</tr>
</tbody>
</table>

the biggest QA systems in China. Subsequently, for a given user, 5,000 words with commercial values, such as insurance, loans, etc., were extracted from webpages that the user has browsed. These words are used to describe users and then each user is represented with a $?/1$ vector, where 1 represents a particular user browsing the corresponding word, and the question mark (?) represents unknown. As users’ browsing contents can change over time, the task here is to predict which words the user will browse in the future.

Table 5.1 presents the statistics of the datasets. In summary, the Douban collection contains three types of user behavior: movie watching, music listening, and book reading, while the Tencent collection contains two other types: music collection and user profile modeling. We observe that the user-item interactions and user-user relations in each single network are very sparse. The degree distributions of the social networks in both Douban and Tencent datasets are plotted in Figure 1.3. We observe that degree distribution of each network follows the power-log distribution, which means each sub network is still sparse and most users have limited interactions. Hence we should utilize the knowledge from multiple networks together.

**Data Sampling Process** For the Douban dataset, we crawled the data through the APIs provided by the website. We randomly retrieved 30 senior users, who registered more than 4 years ago and are still active. Then we performed breadth-first-search based on their online contacts to collect more users iteratively, until the number of users reached a threshold (50,000 in this chapter). Subsequently, the contact relations and the social gathering co-occurrence among these users, and their behaviors on music, book and movie domains were recorded. For the Tencent dataset, we firstly picked a user whose number of neighbors is around 100 randomly. Then, this user as well as his/her neighbors on QQ and Microblog networks are retrieved as a seed set. Similar to the Douban data, we performed breadth-first-search on these two networks to extend the user set until the number of users exceeded a threshold. Finally, corresponding social networks of these users, their music collection records and webpages browsing logs within two months were extracted as the experimental data. Behavior data in the final week are hold out as the test set.
Table 5.3: Performance Comparisons on Tencent Collection (ComSoc-Adap)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>LDA</th>
<th>LinkLDA-QQ</th>
<th>LinkLDA-MB</th>
<th>LinkLDA-C</th>
<th>RTM-QQ</th>
<th>RTM-MB</th>
<th>RTM-C</th>
<th>ComSoc-Adap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>1233.1</td>
<td>1217.9</td>
<td>1189.7</td>
<td>1187.1</td>
<td>1231.2</td>
<td>1198.3</td>
<td>1188.1</td>
<td>1150.7</td>
</tr>
<tr>
<td>MAP</td>
<td>0.5745</td>
<td>0.5839</td>
<td>0.5857</td>
<td>0.5861</td>
<td>0.5885</td>
<td>0.5861</td>
<td>0.5866</td>
<td>0.6064</td>
</tr>
<tr>
<td>Music</td>
<td>21118.8</td>
<td>21041.2</td>
<td>21032.1</td>
<td>21020.3</td>
<td>21016.8</td>
<td>21042.3</td>
<td>21080.2</td>
<td>20942.3</td>
</tr>
<tr>
<td>MAP</td>
<td>0.8497</td>
<td>0.8580</td>
<td>0.8582</td>
<td>0.8677</td>
<td>0.8559</td>
<td>0.8591</td>
<td>0.8681</td>
<td>0.8795</td>
</tr>
</tbody>
</table>

5.4.2 Performance Comparisons

Table 5.2 and Figure 5.4 summarize the perplexity and MAP of LDA, LinkLDA, RTM and the proposed ComSoc-Adap on the three datasets of the Douban collection. For the Movie dataset, as highlighted in bold, ComSoc-Adap consistently outperforms LDA, LinkLDA and RTM on MAP and perplexity. The MAP of ComSoc-Adap is at least 0.017 larger than LDA. If we overlook the model differences, compared to its competing methods, ComSoc-Adap on average achieves 0.017, 0.016 and 0.007 higher MAP as compared to LDA, LinkLDA and RTM respectively. The better performance of ComSoc-Adap over RTM and LinkLDA with one sub network, e.g., RTM-Online, etc., can be ascribed to the fact that, ComSoc-Adap considers more knowledge from auxiliary networks, and hence, captures more aspects of users’ interests. For example, users can share their interests through different sub networks so some regularization knowledge will be missed if only one network is considered. On the other hand, although RTM-C and LinkLDA-C explore knowledge from multiple networks to improve their accuracies, they assign uniform regularization on users without considering the network differences to each user, i.e., different users actually receive different levels of influence from their neighbors in different social networks. In this sense, RTM-C and LinkLDA-C may introduce unnecessary regularization from unrelated networks. For example, the offline network is less useful to infer users’ interests. As shown in Table 5.2, LinkLDA-Off and RTM-Off do not improve the performance of LDA. However, ComSoc-Adap adaptively selects network and hence solve the above problems. Moreover, in all cases, ComSoc-Adap outperforms LDA on all three datasets, implying that when the historical interaction knowledge is sparse, social network information can actually improve the prediction performance.

Table 5.3 and Figure 5.3 summarize the results on the two datasets from Tencent. Similar to the performance in the Douban collection, ComSoc-Adap consistently outperforms RTM, LinkLDA and LDA on both perplexity and MAP for both datasets. For example, on the Profile dataset, ComSoc-Adap gets MAP higher than 0.605 while other baselines do not exceed 0.587. In addition, on the Music dataset, the improvement of ComSoc-Adap is as high as 0.03 on MAP comparing to LDA. Importantly, the improvement of ComSoc-Adap on Tencent collections is more significant. One reason is that, users share more contents through both QQ and Microblog networks, and hence
the links in these two networks can reflect more users’ interests. These justify the effectiveness of the proposed methods empirically.

5.4.3 Performance Analysis

We conduct five extensive experiments on the Douban collection to answer the following questions:
(1). Does ComSoc-Adap transfer knowledge from composite social networks to different users adaptively? (2). Does ComSoc-Adap enrich the knowledge in the user-item interaction network and therefore reduce the “sparsity”? (3). How does the correspondence ratio between composite social network and user-item interaction network affect ComSoc-Adap’s performance? (4) What is
the effectiveness of the model parameters?

We answer the first question by examining the network proportions of different users in the Movie dataset captured by ComSoc-Adap. We pick 100 users who have the most behavior data, as these users have the highest confidence to derive the weights of different networks. The results are presented in Figure 5.5(a), where the y-axis represents the proportion difference between online and offline networks of a given user $u$: $Pr(\text{network=online}|u) - Pr(\text{network=offline}|u)$. The differences of most users are not equal to zero, meaning different networks have different impacts to different users and ComSoc-Adap utilizes the knowledge in each network adaptively.

We evaluate the results on long-tail users, whose number of behaviors are smaller than 10. These users suffer from the harm of “sparsity” the most, since each of them interacts with only $\leq 0.01\%$ items. The results are presented in Figure 5.5(b). We observe that, the improvement of ComSoc-Adap on long-tail users is at least twice more than the average level. For example, on the Book dataset, the improvement of ComSoc-Adap is smaller than 0.01 on MAP on the average level, while larger than 0.03 on long-tail users. One reason is that, for those users who have a large number of interactions, LDA has gained enough knowledge to infer their distributions over topics; but for the long-tail users, ComSoc-Adap can significantly exploit knowledge from the composite social network to help enhance the prediction. This provides justification to the argument that ComSoc-Adap can enrich the knowledge in the user-item interaction network.

In real world, many people may only use parts of on-line services, i.e., they may exist in one sub network but not in all networks. We test the effect of the correspondence ratio between the composite social network and the user-item interaction network, e.g., the number of users who exist in both social and interaction networks. Figure 5.5(d) presents the results with different ratios between the offline/online networks and the Movie interaction network. ComSoc-Adap’s performance becomes worse if fewer correspondences are provided across networks. An important observation is that it outperforms LinkLDA consistently if there are corresponding users among networks, implying that ComSoc-Adap successfully use the overlapping users as bridges to transfer knowledge.

We analyze the effect to change the number of topics $K$ on the Douban collection. We change $K$ from 5 to 100. The results are illustrated in Figure 5.6(a). We observe that the performance of ComSoc-Adap improves with the increase of $K$ but the increment becomes smaller. The reason is that $K$ represents the model complexity. Thus, when $K$ is too small, the model has limited ability to describe the data. On the other hand, when $K$ exceeds a threshold (e.g., 50, the default value in the experiment), the model is complex enough to handle the data. At this point, it is less helpful to improve the model performance by increasing $K$. In practice, $K$ can be tuned through
cross-validation techniques [131].

5.4.4 Efficiency Analysis

As analyzed above, the computational time of ComSoc-Adap increases linearly with the number interactions between users and items, the number of links in the composite networks, and the number of topics $K$. We evaluate this empirically as shown in Figure 5.6(b)∼(c). Three subfigures from left to right illustrate the computational time of LinkLDA-C, RTM-C and ComSoc-Adap on the Movie dataset, with different ratio of interactions and links, and different number of topics respectively. We observe that the computational time linearly increases with larger data size and number of topics. For our experiment setting, each iteration takes about 80 seconds in our computer, of which memory is 4G and CPU is 2.0Gz. Although ComSoc-Adap has better prediction performance as show in Table 5.2 and Table 5.3, it is faster than RTM and has similar time cost with LinkLDA.
Figure 5.6: Parameter and Efficiency Analysis (ComSoc-Adap)

5.5 Summary

In this chapter, we studied a new problem on how to transfer knowledge from composite social network for user behavior prediction, where the behavior data (or user-item matrix) of the target application can be more than 99.9% empty. We defined a composite social network as a set of nested single networks, where users and links in different sub networks overlap. Each network reflects partial aspects of users’ interests and has different properties. For different users, individual networks have a diverse range of influences. Thus, we proposed a hierarchical Bayesian model to transfer knowledge for different users adaptively. The knowledge from the composite social network is utilized to regularize users’ distributions over topics. Unlike prior works, the proposed model considers knowledge transfer adaptively. This is formulated into a parametric model that controls the probability of how one user is influenced by a given network. In addition, different from the previous relational topic models, ComSoc-Adap exploits user-topic distributions to generate users’ social relations. The proposed model is flexible in that, it can be extended to any number of networks. We proposed a large-scale implementation based on Map/Reduce to cope with massive amount of data. We conducted empirical studies on two data collections from Tencent Inc. and Douban, where ComSoc-Adap outperforms other baselines by as high as 0.03 on MAP.
**Applicability Discussion** The success of knowledge transfer from users’ composite social networks to modeling their behaviors relies on their closeness. Thus, we assume that the provided social networks should reflect users’ interests that are related to their behaviors. For example, we cannot use users’ relationship on LinkedIn to infer users video interests on Youtube.

**Future Works** In this chapter, we only build models to predict whether a user will interact with a given item or not. In future work, we plan to extend the proposed model to handle different kinds of behavior, such as rating, multi-class prediction, etc. We will extend the experiments over multiple social networks, such as Facebook + Twitter + MSN. We will consider incorporating heterogeneous auxiliary knowledge together, such as item profiles, user behaviors in auxiliary domains, etc.
CHAPTER 6

COMPOSITE USER DISTANCE MODELING

6.1 Introduction

A fundamental challenge in social network analysis is how to measure users’ distance or latent similarity. Accurate user distance measure is important to ensure success of these analytical tasks, including link prediction [66], community detection [76], social influence [113], and etc., that are used to build social network applications.

Social media applications capture users’ information from various and heterogeneous aspects: node features, link information and community structures. For example, users have profiles (age, gender, education, etc.), as well as behavioral data (songs listened, movies watched, etc.); each user pair can be described by different criteria, such as number of common neighbors, number of interactions etc.; and multiple users can also form communities. Thus, one could potentially adopt metric learning [114] to learn a comprehensive and yet single distance measure for a social network, by unifying different types of information from heterogeneous information sources. However, data in an individual network can be quite sparse; each user may just have a few connected neighbors - in a dataset from Tencent, there are more than 99.9% empty entries in the user-user relation matrix and only a few users provide their profiles. The rare observations in an individual network can make models overfit. It is difficult to measure the distance between two users in a given network if they do not directly share anything in common yet.

An important observation is that as different service providers offering complementary services, people are now engaged in many networks simultaneously. For example, people use Facebook to communicate with their friends, post information on Twitter and browse videos uploaded by others on YouTube - all simultaneously. For a given target network, other related networks provide auxiliary knowledge that can help solve the sparsity problem. However, one cannot simply merge data from multiple networks due to their intrinsic difference. First, different networks have different properties, such as density, degree distributions, etc. If we merge a dense network and a sparse one directly, the knowledge in the sparse network will be hidden inside the dense one. Second, distances between two users could be different for individual networks, as they have a variety of all kinds of interests in provided services. Two users may be close on YouTube as they share similar interests
in video, but may not necessarily become friends on Facebook as they are not familiar with each other in real life. As an illustration, users $a$ and $b$ in Figure 6.1(b) are closer in network B than A, as they share more common friends in B. Thus the motivation is to learn an adaptive distance measure for a target network, by leveraging knowledge in multiple related social networks and handling the network difference in the same time.

We propose an adaptive metric learning framework, formulated as a convex optimization problem with a boosting-based solution. The main idea is to embed the target network into a latent feature space based on node features and link information. In the latent space, the topological and communal structures are preserved, by exploiting the knowledge in multiple networks collectively such that user distance can be accurately measured. In addition, irrelevant data from auxiliary networks are eliminated in each round of boosting to avoid possible negative impacts brought by network differences and useless information.

6.2 Challenges

As users’ distance in different networks can be very different but related and the obtained data can be heterogeneous, we observe three fundamental challenges as follows:

1. Users’ distances can be very different in different networks but data in a single network is not enough to build an accurate measure. Thus, we have to adaptively select relational knowledge from other networks.

2. Distance knowledge are encoded in different formats, including social relations, personal behaviors and social interactions. Thus, another challenge is how to transform these knowledge into a unified representation.
3. The proposed algorithm should be scalable as the number of users in real-world networks can be extremely big.

As far as we know, although some previous works studied the problem of distance measure [22], they did not consider so comprehensive data in multiple networks.

### 6.3 Adaptive User Distance Modeling

In this section, we firstly design a metric learning approach that measures the distance between two users by exploiting heterogeneous social media data. Then we propose a boosting based framework to solve the data sparsity problem by importing data from auxiliary networks. Intuitively, users’ behavior and profile records can be considered as features of users and thus each user can be represented as a feature vector. Then metric learning can measure the distance between two users based on their features. In addition, analogous to the class information in machine learning task, we can use users’ link information and community structures to regularize the modeling process of metric matrices. The most important point of our intuition is to transform users’ links into a set of constraints, which provide a unified representation to encode the relational knowledge across networks. Then the auxiliary constraints from other networks can be adaptively selected for solving the data sparsity problem in the current network.

#### 6.3.1 Relational Metric Learning

In order to construct a comprehensive distance measure, we need to exploit data from various information sources, including the network topology, users’ features, users’ interactions as well as community structures. However, these data have completely different representations. Thus we need to embed them in a unified feature space before exploiting their knowledge. To achieve this, we propose an algorithm called RML (Relational Metric Learning). RML constructs $M$ and $\omega$ in Eq.(2.32) to satisfy two constraints: (1) the network topology is preserved; (2) users maintain their communities in the embedded feature space. The motivations are that (1) for a given user, her/his distances to all disconnected users must be larger than the distance to the farthest connected neighbor; (2) people in the same community are closer to each other and share more common interests, so if the community structures can be preserved after embedding, users’ distance can be measured more accurately. Formally, we define the following constraints: $\forall i, j, k$

$$D(u_i, u_j) > (1 - A_{ij}) \max_k \left( A_{ik} D(u_i, u_k) \right) \quad (6.1)$$
Recall that in Eq.(2.32), learning the metric matrix $M$ is equivalent to finding a linear embedding on the input features $LX$, where $M = LL^T$ and $L \in \mathbb{R}^{f \times f}$. Then,

$$D(u_i, u_j) = (x_i - x_j)M(x_i - x_j)^T + \omega r_{ij}^T$$

(6.2)

$$= (x_i - x_j)LL^T(x_i - x_j)^T + \omega r_{ij}^T$$

$$= (x_iL - x_jL)(x_iL - x_jL)^T + \omega r_{ij}^T$$

This provides the right formulation to incorporate the community structure knowledge. We preserve the community structure in the embedding space by maximizing the normalized modularity [76]. Let $d_i$ denote the degrees of user $u_i$ and $m$ denote the number of links. Formally,

$$\max_{L, \omega} \frac{1}{4m} \sum_{ij} \left( A_{ij} - \frac{d_i d_j}{2m} \right) (x_iL)(x_jL)^T = \frac{1}{4m} \text{Tr}\left( (XL)^T B(XL) \right)$$

(6.3)

where $\text{Tr}$ is the trace, $B$ is the modularity matrix and $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$. By combining Eq.(6.1) and Eq.(6.3) together, we obtain the following objective

$$\max_{L, \omega} \frac{1}{4m} \text{Tr}\left( (XL)^T B(XL) \right) - \beta \left( ||L||_2 + ||\omega||_2 \right)$$

(6.4)

$$s.t. \forall i, j, k, D(u_i, u_j) > (1 - A_{ij}) \left( A_{jk}D(u_i, u_k) \right)$$

$$M = LL^T, \quad M \succeq 0$$

By now, we observe that users’ features and interactions are used to calculate the distance while users’ link information and community structure are utilized to construct $M$ and $\omega$. However, this objective is non-convex and can be hard to optimize. Thus, we derive another equivalent convex objective function. For all distance constraints $(i, j, k)$ of each user $u_i$, where $A_{ij} = 1$ and $A_{ik} = 0$, we denote their aggregations $\sum_{u_i \in U} \left( \sum_{j,k} \right)$ as $\sum_{i,j,k}$.

$$\min_{M, \omega} \sum_{ij} A_{ij} \left( D(u_i, u_j) - \omega r_{ij} \right) + \lambda \sum_{i,j,k} \left( \text{Tr}(C^{(i,j,k)} X M X^T) + \omega(r_{ij} - r_{ik}) \right)$$

(6.5)

$$+ \beta/2(||M||_2 + 2||\omega||_2)$$

s.t. $M \succeq 0$

where each $C^{(i,j,k)}$ is a sparse matrix and corresponds to a constraint $\{(i, j, k), A_{ij} = 1, A_{ik} = 0\}$

$$C^{(i,j,k)}_{jj} = +1, C^{(i,j,k)}_{ik} = +1, C^{(i,j,k)}_{ki} = +1$$

$$C^{(i,j,k)}_{ij} = -1, C^{(i,j,k)}_{ji} = -1, C^{(i,j,k)}_{kk} = -1$$

Although this objective uses hard margin, soft margin can be adopted using similar techniques as in [97]. We show two properties of the objective in Eq.(6.5) as follows.
Algorithm 3 Projected SGD for RML

1: Input: Constraint Matrices: $S = \{C^{(i,j,k)}\}; I; N$
2: $M \leftarrow I$
3: For $i = 1$ to $I$
4:   Learn $\omega$ using SGD based on Eq.(8)
5:   $C = \{0\}^{n \times n}$
6: For $o = 1$ to $N$
7:   Sample matrix $C^{(i,j,k)}$ from $S$
8:   IF $D(u_i, u_j) - D(u_i, u_k) + 1 > 0$
9:       $C = C + C^{(i,j,k)}$
10:   Update $M$ using Eq.(6.7) and project $M$ into a PSD cone
11: IF the algorithm is convergent Break
12: Return Metric Matrix $M$, $\omega$

Theorem 1. Maximizing objective in Eq.(6.4) is equivalent to minimizing objective in Eq.(6.5).

Proof. According to the definition of $C^{(i,j,k)}$,

$$\text{Tr}(C^{(i,j,k)}XMXT)$$

$$= (x_i - x_j)M(x_i - x_j)^T - (x_i - x_k)M(x_i - x_k)^T$$

$$= D(u_i, u_j) - D(u_i, u_k) - \omega(r_{ij} - r_{ik})$$

Using Lagrange multiplier, constraints can be transformed to a term plus a trade-off parameter in the objective. Then the first constraint in Eq.(6.4) becomes the third term in Eq.(6.5). Based on the conclusion in [122], maximizing the modularity equals to finding the min-cut, i.e.,

$$\max_L 1/4m \text{Tr}((XL)^TB(XL)) \sim \min_L \text{Tr}((XL)^T\mathcal{L}(XL)),$$

where $\mathcal{L} = D - A$ is the Laplacian matrix, $D$ is a diagonal matrix and $D_{ii} = \sum_{j} A_{ij}$. Consequently,

$$\text{Tr}((XL)^T\mathcal{L}(XL)) = \sum_{ij} A_{ij}(x_iL - x_jL)(x_iL - x_jL)^T$$

$$= \sum_{ij} A_{ij}(x_i - x_j)LL^T(x_i - x_j)^T$$

$$= \sum_{ij} A_{ij}(x_i - x_j)M(x_i - x_j)^T$$

$$= \sum_{ij} A_{ij}(D(u_i, u_j) - \omega r_{ij})$$

Finally, $||M||_2 = ||L||_2 + ||L^T||_2$. Thus the objective in Eq.(6.4) becomes the one in Eq.(6.5). □

Theorem 2. Objective in Eq.(6.5) is convex w.r.t $M$ and $\omega$.

Proof. First, the second-order derivative of the $L_2$-norm for a matrix is positive and thus the term $||M||_2$ is convex. The term $\text{Tr}(C^{(i,j,k)}XMXT)$ has been shown to be convex in [97]. The second-order derivative of the term $\sum_{ij} A_{ij}(D(u_i, u_j) - \omega r_{ij})$ is 0. For $\omega$, its second-order derivative is $2\beta > 0$. Thus, the objective is jointly convex w.r.t $M$ and $\omega$. □
For efficiency concerns, instead of employing semi-definite programming (SDP), we derive a projected stochastic gradient descent algorithm to optimize Eq.(6.5). We compute its sub gradients at $M$ and $\omega$ as follows:

$$\nabla M = \beta M + \lambda X^T C X + \sum_{i,j,A_{ij}=1} Y_{ij} \tag{6.7}$$

$$\nabla \omega = 2 \beta \omega + \lambda \sum_{i,j,k} (r_{ij} - r_{ik}) - \sum_{i,j,A_{ij}=1} r_{ij} \tag{6.8}$$

where $Y_{ij} = (x_i - x_j)^T (x_i - x_j)$ and $C = \sum_{i,j,k} C^{(i,j,k)}$. In practice, we optimize $M$ and $\omega$ alternatively. The details to learn $M$ and $\omega$ can be found in Algorithm 3, where $S$ is the set of constraints. We initialize $M$ as an identity matrix and then in each iteration, we compute the matrix $C$ and update $M$ using the sub gradient in Eq.(6.7) with learning rate $\nu$. Finally, we project $M$ into a PSD cone to satisfy the PSD constraints. The objective function is convex and its function value decreases in each iteration, so Algorithm 3 can converge to a global optimal solution.

### 6.3.2 Adaptive Relational Metric Learning

We propose an adaptive metric learning algorithm to reduce the “sparsity” in the target network by utilizing the knowledge from the auxiliary network and eliminating irrelevant data. Let $S_a$ and $S_t$ denote the sets of constraints in the auxiliary and target networks respectively. The idea of adaptation is to remove those constraints from the auxiliary network $G_a$ that cannot be satisfied, as they are different from the knowledge in the target network and cannot contribute to the target distance modeling. We firstly learn $\omega$ and then construct $M$ based on the fixed optimal $\omega$. Thus, we aim to optimize the following objective

$$\min_M \sum_{i,j} A_{ij} D_M(u_i, u_j) + \lambda \sum_{i,j,k} \text{Tr}(C^{(i,j,k)} XMX^T) + \beta/2 ||M||_2 + o \tag{6.9}$$

$$+ \quad s.t. \quad M \succeq 0$$

where $o$ is a constant given by the optimal value of $\omega$ and $D_M(u_i, u_j) = (x_i - x_j)^T M (x_i - x_j)^T$. As follows, we first formulate the objective of Eq.(6.9) as a boosting problem, and then propose a novel weight updating rule to incorporate auxiliary knowledge. A PSD matrix $M$ can be decomposed as $M = L \Sigma L^T = \sum_k \alpha_k L_k L_k^T$, where $\Sigma$ is a diagonal matrix, in which the $k$-th element is $\alpha_k$ corresponding to the $k$-th eigenvalue, and $L_k$ is the $k$-th column of $L$ corresponding to the $k$-th orthonormal eigenvector. Then $D_M(u_i, u_j)$ becomes

$$D_M(u_i, u_j) = (x_i L - x_j L) \Sigma (x_i L - x_j L)^T \tag{6.10}$$
Thus, the distance under metric $M$ can be considered as the distance in an embedding space under metric $\Sigma$. In addition, since $M \succeq 0$, we have $\forall k, \alpha_k \succeq 0$, and $M = \sum_k \alpha_k Q_k$ where matrices $Q_k = L_k L_k^T$ are rank-one matrices. In general, the matrix $M$ can be represented as a weighted combination of rank-one matrices. In this formulation, we consider each rank-one matrix as a weak learner, and exploit a boosting-based algorithm to learn the combination coefficients. We define a weak learner based on each rank-one matrix $Q_k$ as

$$h_k(x_i, x_j) = (x_i - x_j)L_k L_k^T(x_i - x_j)^T \quad (6.11)$$

Then the loss for each triple constraint $(i, j, k)$ is defined as $h(x_i, x_j) - h(x_i, x_k)$. In practice, we truncate this loss to make it into $(-1, 1)$. From the boosting perspective, we aim to minimize the following exponential loss

$$E = \sum_{i,j,k} \exp \left\{ \sum_q \alpha_q (h_q(x_i, x_j) - h_q(x_i, x_k)) \right\} \quad (6.12)$$

where this objective can be minimized stepwise. In the $q$-th iteration, we optimize this objective w.r.t $\alpha_q$ and $h_q$ only, and fix the previous $q-1$ weak learners and coefficients. Then the objective becomes $E = \sum_{i,j,k} w^q_{i,j,k} \exp \left\{ \alpha_q (h_q(x_i, x_j) - h_q(x_i, x_k)) \right\}$, where $w^q_{i,j,k} = \exp \left\{ \sum_{c=1}^{q-1} \alpha_c (h_q(x_i, x_j) - h_q(x_i, x_k)) \right\}$, and can be considered as constants since they are independent from $h_q$ and $\alpha_q$. Then, following the deduction process in [70], we can minimize $E$ w.r.t $h_q$ as

$$h_q = \arg \min_{h_c} \sum_{i,j,k} w^q_{i,j,k} [h_c(x_i, x_j) - h_c(x_i, x_k)] \quad (6.13)$$

where $[\pi] = 0$ if $h_c(x_i, x_j) < h_c(x_i, x_k)$ and 1 otherwise. This function can be determined by minimizing the objective in Eq.(6.4), in which it replaces $L$ with a vector $L_q$ and sets $\Sigma = I$ with a projected gradient descent algorithm in Algorithm 3. The gradient w.r.t $L_q$ is

$$\nabla L_q = 2\beta L_q + \lambda \sum_p X^T C_p X L_q + \sum_p \sum_{i,j} A^p_{ij} (x_i - x_j)(x_i - x_j)^T L_q \quad (6.14)$$

After learning $h_q$, we also need to find the optimal $\alpha_q$. By computing the deviation of $E$ w.r.t $\alpha_q$ and rearranging the terms, we obtain the solution of $\alpha_q$ as

$$\alpha_q = \log \left( \frac{1 + \epsilon_q}{1 - \epsilon_q} \right)/2 \quad \epsilon_q = \frac{\sum_{i,j,k} w^q_{i,j,k} (h_c(x_i, x_k) - h_c(x_i, x_j))}{\sum_{i,j,k} w^q_{i,j,k}} \quad (6.15)$$

Finally, after learning every $\alpha_q$ and $h_q$, the weight of each constraint $(i, j, k) \in S_t$ is updated as

$$w^{q+1}_{i,j,k} = w^q_{i,j,k} \exp \left\{ \alpha_q (h_q(x_i, x_j) - h_q(x_i, x_k)) \right\} \quad (6.16)$$
Algorithm 4: Boosting for ComSoc-ML

1: **Input:** Constraint Matrices: $S_t, S_a; f$
2: Initialize the weights $w^0_i$ for each constraint as $\frac{1}{|S_a| + |S_t|}$
3: For $i = 1$ to $f$ do
4: Learn $L_i$ using Algorithm 3 with weights $w^{i-1}_i$
5: Compute $\alpha_{t,i}^f$ in the target network using Eq.(6.15)
6: Update weights for constraints in $S_t$ using Eq.(6.16)
7: For $(i,j,k) \in S_a$ do
8: $w^{i}_{a,i,j,k} = w^{i-1}_{a,i,j,k} [h_q(x_i, x_j) - h_q(x_i, x_k)]$
9: Normalize $w^i$ so that $w^i$ is a distribution
10: Return Metric Matrix: $M = \sum_i \alpha_{t,i}^f L_i^T L_i, \omega$

For the auxiliary constraints in $S_a$, we set their weights as 0 once they are violated, to eliminate any useless data, but keep their weights as the same if they can be satisfied. The complete algorithm can be found in Algorithm 4, where $S_a$ and $S_t$ are constraint sets in auxiliary and target networks respectively. In each iteration, a weak learner related to one rank-one matrix is built and then its weight is computed. Consequently, the weights of every target and auxiliary constraints are updated using different strategies. One advantage of ComSoc-ML is that: as the knowledge in different networks are extracted and formulated as constraints, it conceals users’ identities, and thus does not require the user correspondence between networks. This enables ComSoc-ML to borrow knowledge from the auxiliary network without entity resolution. We discuss the error bound and convergence of ComSoc-ML as follows.

**Lemma 1.** In Algorithm 4, the expected weight of each constraint in $S_a$ in the $f$-th iteration is:

$$E_{(i,j,k) \in S_a} \{w^f_{a,i,j,k}\} = \frac{1}{|S_a| + |S_t|} \prod_{t=1}^{f} (1 - \epsilon_t) \geq \frac{(0.5 + \gamma)^f}{|S_a| + |S_t|} \tag{6.17}$$

**Proof.** The proof is straightforward. Since in each iteration, the training error is $\epsilon^t$ and the initial weight is $\frac{1}{|S_a| + |S_t|}$. Recall the updating equation in Algorithm 4, we obtain the equation. Finally, let $\gamma_t = 0.5 - \epsilon_t$ and $\gamma \geq \gamma > 0$, we obtain the Lemma.

**Theorem 3.** The training error of the final model $h$ in the target network, is bounded by

$$\epsilon_t(h) \leq e^{-2\gamma^2 f \left( \frac{|S_a| + |S_t|}{|S_t|} - \frac{(0.5 + \gamma)^f|S_a|}{|S_t|} \right)} \tag{6.18}$$

where $|S_a|$ and $|S_t|$ are the number of constraints in the auxiliary and target networks respectively. **Proof.** The proof is analogous to that in [70] but with different terms. Let $Z_t = \sum_p \sum_{i,j,k} w^t_{p,i,j,k} \exp \{ \alpha_t^p (h_t(x_i, x_j) - h_t(x_i, x_k)) \}$ denote the normalization term in the $t$-th iteration and $\epsilon(h) = \sum_p \sum_{i,j,k} w^0_{p,i,j,k} [h(x_i, x_j) - h(x_i, x_k)]$ denote the training error of the final
model $h$, where $\ell$ is the number of networks. As $[\pi] \leq \exp(\pi)$, we have

$$\varepsilon(h) \leq \sum_{p} \sum_{i,j,k} w^0_{p,i,j,k} \exp \left\{ h(x_i, x_j) - h(x_i, x_k) \right\}$$

$$= \sum_{p} \sum_{i,j,k} w^{f+1}_{p,i,j,k} \Pi_{t=1}^f Z_t = \Pi_{t=1}^f Z_t$$

According to the conclusion in [70],

$$\Pi_{t=1}^f Z_t = \Pi_{t=1}^f \left\{ 2 \sqrt{\varepsilon_t(1 - \varepsilon_t)} \right\} = \Pi_{t=1}^f \left\{ \sqrt{1 - 4\gamma_t^2} \right\}$$

$$\leq \exp \left( -2 \sum_{t=1}^f \gamma_t^2 \right) \leq e^{-2\gamma^2 f} \quad (6.20)$$

Since the training error is contributed by both the auxiliary and the target networks, the training error in the target network is bounded as

$$\varepsilon_t(h) \leq \frac{|S_a| + |S_t|}{|S_t|} \left\{ \Pi_{t=1}^f Z_t - \sum_{i,j,k} w^f_{a,i,j,k} [h(x_i, x_j) - h(x_i, x_k)] \right\}$$

$$\leq \frac{|S_a| + |S_t|}{|S_t|} \left\{ \Pi_{t=1}^f Z_t - \Pi_{t=1}^f Z_t \sum_{i,j,k} w^f_{a,i,j,k} \right\}$$

$$\leq \frac{|S_a| + |S_t|}{|S_t|} \left\{ \Pi_{t=1}^f Z_t - \Pi_{t=1}^f Z_t \frac{(0.5 + \gamma)^f |S_a|}{|S_a| + |S_t|} \right\}$$

$$\leq e^{-2\gamma^2 f} \left( \frac{|S_a| + |S_t|}{|S_t|} - \frac{(0.5 + \gamma)^f |S_a|}{|S_t|} \right)$$

**Generalizability** Recall the definition of function $h_k$ in Eq.(6.11), it can be expanded and rewritten as $h_k(x_i, x_j) = \sum_{f,g} L^f_k L^g_k (x_{if} - x_{jf})(x_{ig} - x_{jg})$. It can be considered as a quadratic polynomial classifier. Let us denote its VC-dimension as $d_V$. Based on the result in [70], with high probability, the generalization error of ComSoc-ML in the target network is at most

$$\varepsilon_t(h) + O\left( \sqrt{\frac{d_V}{|S_t|}} \right).$$

**Time Complexity** For a given network, ComSoc-ML requires $I$ iterations before convergence, and in each iteration, it samples $N$ constraints. For each constraint, it requires $O(f \cdot f)$ time to compute the distances between user pairs, as well as an additional $O(f \cdot f)$ time to update the metric matrix $M$. Given all these, the time complexity of RML is $O(I \cdot N \cdot f \cdot f)$. ComSoc-ML requires an outer loop with $T$ iterations to build low-rank metric matrices, and thus the overall time complexity is $O(I \cdot N \cdot f \cdot f \cdot T)$. In practice, $N$ is chosen as the order of the number of observed links in the network, that means the proposed algorithm can be scaled up to large datasets.
Table 6.1: Summary of Data Characteristics

<table>
<thead>
<tr>
<th>Data</th>
<th>Tencent QQ-IM</th>
<th>Tencent Microblog</th>
<th>Douban On-line</th>
<th>Douban Off-line</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Nodes</td>
<td>~1M</td>
<td>~50K</td>
<td></td>
<td></td>
</tr>
<tr>
<td># of Node Features</td>
<td>50</td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td># of Edges</td>
<td>~80M</td>
<td>~32M</td>
<td>~5M</td>
<td>~4M</td>
</tr>
</tbody>
</table>

6.4 Experiments

We evaluate the proposed metric learning models on the link prediction task - one of the most widely employed social learning tasks, by examining whether the linked users are closer than others.

6.4.1 Experimental Setting

We introduce four baselines: SVM, MMSB [3], SRW [4] and SPML [97]. SVM learns a classification model to predict whether one link exists between two users or not, MMSB builds models using only link information, SRW is a random walk based model designed for link prediction specifically, and SPML is a metric learning model for single networks without considering community structure. We show that the proposed algorithm, ComSoc-ML, performs better than both simply merging all networks and building models in a single network. Thus, for all baselines, we implement two variations. One is to learn a model on each single network and the other is to perform modeling on a naive combined network, of which the edge set is the union of edge sets in all single networks.

We exploit two real-world datasets. The first dataset is from Tencent\(^1\), which contains two social networks: one is an instant messaging network, QQ, and the other is Tencent’s Microblog network. In this dataset, the 50 tags used most often by users are regarded as node features. The second dataset is from Douban\(^2\) which contains two networks as well: one is an on-line network that represents users’ visual contact relations and the other is an off-line network which represents users’ physical relations. In addition, users’ ratings on top 50 movies, 50 songs and 50 books are extracted as node features. To accelerate the computation, for each observed link we select 5 no-linked user pairs as the negative examples. In addition, for each user pair, we introduce four features: time length, number of hops, number of common neighbors and Katz Index [46]. Table 6.1 presents the statistics for different networks. The evaluation task is to predict who will build links with others. We select 10% of the links in the whole dataset according to the temporal information

\(^1\)http://tencent.com/en-us/index.shtml
\(^2\)http://www.douban.com/
Table 6.2: Performance Comparisons on MAP and AUC (ComSoc-ML)

<table>
<thead>
<tr>
<th>Data</th>
<th>Criterion</th>
<th>Tencent QQ</th>
<th>Tencent Microblog</th>
<th>Douban On-line</th>
<th>Douban Off-line</th>
<th>Tencent QQ</th>
<th>Tencent Microblog</th>
<th>Douban On-line</th>
<th>Douban Off-line</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.5302</td>
<td>0.5467</td>
<td>0.6215</td>
<td>0.2851</td>
<td>0.3748</td>
<td>0.3311</td>
<td>0.4248</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-SVM</td>
<td>0.5316</td>
<td>0.5565</td>
<td>0.5889</td>
<td>0.3265</td>
<td>0.3883</td>
<td>0.3406</td>
<td>0.4012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMSB</td>
<td>0.5293</td>
<td>0.5465</td>
<td>0.6173</td>
<td>0.2777</td>
<td>0.3625</td>
<td>0.3253</td>
<td>0.4037</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-MMSB</td>
<td>0.5432</td>
<td>0.5509</td>
<td>0.5748</td>
<td>0.3235</td>
<td>0.3719</td>
<td>0.3275</td>
<td>0.3635</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPML</td>
<td>0.5429</td>
<td>0.5755</td>
<td>0.6220</td>
<td>0.3101</td>
<td>0.3953</td>
<td>0.3906</td>
<td>0.4940</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-SPML</td>
<td>0.5447</td>
<td>0.5976</td>
<td>0.5935</td>
<td>0.3125</td>
<td>0.4044</td>
<td>0.4146</td>
<td>0.4288</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRW</td>
<td>0.5589</td>
<td>0.6005</td>
<td>0.6496</td>
<td>0.3394</td>
<td>0.4053</td>
<td>0.4138</td>
<td>0.5133</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-SRW</td>
<td>0.5698</td>
<td>0.6204</td>
<td>0.6212</td>
<td>0.3434</td>
<td>0.4189</td>
<td>0.4228</td>
<td>0.4883</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RML</td>
<td>0.5623</td>
<td>0.5990</td>
<td>0.6235</td>
<td>0.3310</td>
<td>0.3978</td>
<td>0.4170</td>
<td>0.5006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ComSoc-ML</td>
<td><strong>0.6308</strong></td>
<td><strong>0.6346</strong></td>
<td><strong>0.6658</strong></td>
<td><strong>0.3553</strong></td>
<td><strong>0.4255</strong></td>
<td><strong>0.4534</strong></td>
<td><strong>0.5302</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

as the hold-out set, to simulate the real-world scenario.

In addition, to evaluate the significance in improvement of the proposed model, we randomly split the hold-out set into 10 subsets and then calculate the error bar. The results are evaluated through area under the curve (AUC) and mean average precision (MAP). We set $\lambda = 1.0$ and the number of iterations as $f$ (# of features), and also study the effects of these parameters.

### 6.4.2 Performance Comparisons

Table 6.2, Figure 6.2(a) and Figure 6.2(b) present the results for each composite social network dataset of each baseline. It is evident that ComSoc-ML achieves better performance on each dataset. Clearly, SVM and MMSB consider data from only one aspect and thus perform poorly in all networks. In addition, due to the “sparsity” problem, algorithms in single networks fail to capture the true users’ latent interests. To be specific, MMSB achieves AUC and MAP just no more than 0.62 and 0.41 respectively. Although SRW and SPML improve SVM and MMSB by considering user features and link information together, they do not consider community structures and link features but only exploits knowledge in single networks, and thus cannot provide satisfying performance, either. By further incorporating community structure information and features of user pairs, RML achieves better performance. It improves SPML on MAP by 0.02 in both Tencent’s QQ and Douban’s on-line networks. This suggests that we need to incorporate various data sources to achieve superior performance. Another important observation is that, simply merging different networks does not work well. In the off-line network of Douban, the performance of naive combination is even worse than using only an individual network. Thus, from another preservative, it
shows the necessity to consider the individual difference of each network. Nonetheless, the proposed approach that collectively and adaptive learns over composite networks, ComSoc-ML, has the highest AUC and MAP in all datasets. Specifically, it achieves at least 0.01 higher AUC and 0.02 higher MAP than all baseline methods in all networks. The improvement of ComSoc-ML in QQ is higher than other networks. That means ComSoc-ML achieves higher improvement to sparser networks. The better performance of ComSoc-ML over C-SPML and C-SRM can be ascribed to the instance weighting strategy in the boosting-based framework which benefits from auxiliary knowledge, while avoiding the harm of network differences.

### 6.4.3 Performance Analysis

In addition, we evaluate ComSoc-ML and SPML models on the long-tail users, the number of whose neighbors are smaller than 10. They suffer from “the curse of sparsity” the most. The results are presented in Figure 6.3(a). We observe that, ComSoc-ML improves SPML on these users more than the average level. One reason is that, for those users who have large amount of interactions, SPML has gained enough knowledge to infer distance metric, but for the long-tail users, ComSoc-ML can exploit knowledge from other networks to help enhance the prediction. We illustrate the computational time of SRW-C, SPML-C, RML and ComSoc-ML on Douban networks in Figure 6.3(b) to evaluate their efficiencies. For the chosen experiment setting, ComSoc-ML takes about 10 seconds for each iteration in our computer, of which memory is 16G and CPU is 3.2Gz.
6.4.4 Parameter Analysis

We analyze the effects of two parameters: the trade-off parameter $\lambda$ and the number of iterations, on the Tencent collection. The former indicates the trade-off between the importance of community structure and graph topology terms in the objective. We vary $\lambda$ from 0.5 to 1.5 with step 0.25. As shown in Figure 6.3(c), the performance of RML goes up first and then drops down when $\lambda$ keeps increasing. The graph topology effectiveness is overly emphasized when $\lambda$ is too small, but when $\lambda$ becomes large, the effectiveness of community structure is ignored. In practice, this parameter can be tuned through cross-validation [131]. However, an important observation is that, in most cases, RML outperforms SPML. This indicates the importance to model and maintain community structure in the embedding space, as it intrinsically measures users’ latent similarity. Figure 6.3(d) presents the results with different numbers of iterations. With increasing number of iterations, ComSoc-ML’s performance increases and then converges. By inheriting the advantage of Adaboost, ComSoc-ML resists overfitting.
6.5 Conclusion and Future Works

We comprehensively studied the adaptive user distance modeling problem in social networks by simultaneously taking users’ features, link information and network structures into account. We first proposed a novel relational metric learning algorithm (RML) that preserves both network topology and community structure and exploits knowledge from heterogeneous information sources in a unified framework. As data in individual networks are sparse, where a single network reflects only some aspects of users and has different properties. As a result, each network by itself does not fully describe the users’ relationships and interests. To resolve the sparsity challenge, we utilized link knowledge from multiple networks collaboratively, as well as other user profiles together, to learn individual metric particularly adapted for the target network. We proposed an optimization and boosting-based framework for network-adaptive metric learning based on our proposed RML, called ComSoc-ML. It chooses only relevant constraints from auxiliary networks to help learn the metric for the target network. We analyzed its convergence and generalizability. To evaluate the effectiveness of the constructed distances, experiments were conducted on the link prediction task with two large-scale data collections, where ComSoc-ML outperforms state-of-the-art baselines as high as 0.07 on MAP and 0.09 on AUC.

Applicability Discussion The assumption of the proposed metric learning approaches is that users’ behaviors and profiles are closely related to users’ social interests and can affect their decisions on friendship. If the obtained behaviors and profiles do not have discriminative ability on their social relations, then the constructed distance can be inaccurate.

Future Works Besides link prediction, we plan to apply the proposed method on other social network analysis tasks in the future, such as community detection and influence analysis. As an example, after embedding the social graph into a corresponding latent space, one can detect community by maximizing modularity (Eq.(6.4)). For some social networks that are initiating, where there is no clear community as many users have not yet connected to others, community detection would fail to uncover meaningful clusters due to the cold-start problem. When this happens, we can incorporate additional networks that share users’ and/or links’ features, to enable better learning for distance measures.
CHAPTER 7

SCALABLE COMPOSITE NETWORK MINING

As we know, nowadays the size of social data becomes extremely huge. For example, the number of social links in Tencent’s QQ network can exceed tens of billions and the interaction logs can be more than one billion per day. Thus, it is not realistic to construct models using these huge data on a single machine, where the computing resources are limited. In this section, we firstly introduce Map/Reduce [25], a data parallel based computation framework, which is simply but effective to scale up many computational tasks. However, the original primitives, Map and Reduce, are not ideal for machine learning algorithms which require many iterations or recursions to construct models. If we invoke a Map/Reduce operation in each iteration, the cost of initializing and I/O operations will become unaffordable. To solve this, we borrow and implement another primitive, AllReduce, from MPI [78], another parallel computation framework. AllReduce merges the computational results from all single machines and then broadcast. As we will demonstrate, this operation makes the implementations of iterative algorithms more efficient and thus benefit the modeling build process of ComSoc a lot, as the inference or optimization algorithms for ComSoc require lots of iterative operations.

7.1 Background of Map/Reduce

Map/Reduce [25] is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. This takes advantage of the parallel processing power of distributed systems, and also reduces network bandwidth as the algorithm is passed around to where the data lives, rather than a potentially huge dataset transferred to a client algorithm. Typically, an execution process of Map/Reduce jobs can be found in Figure 7.1.

Take the training process of logistic regression as an example and suppose that we use gradient descent to construct the model. Then we can design an iterative framework. In each iteration, we implement a Map and a Reduce function, where the Map function computes the sub gradients...
obtained by each data instance and the Reduce function combines all sub gradients and updates the model. This process is carried on until convergence. Formally, we have

- **Map**: Take \( k_i, (x_i, y_i) \) where \( k_i \) is the ID of the \( i \)-th instance and \((x_i, y_i)\) is the correspondent instance, and emit \( 0, \nabla w_i \) where \( \nabla w_i \) is the gradients computed by the \( i \)-th instance.

- **Reduce**: Take \( 0, \nabla w_i \), compute \( \nabla w = \frac{1}{N} \sum_{i=1}^{N} \nabla w_i \), update model \( w \leftarrow w + \eta \nabla w \), where \( \eta \) is the learning rate and emit \( w \).

Recently, Map/Reduce has been implemented under different environments. The most famous one is Hadoop [111], which was designed for large computing clusters and has been applied on many large-scale applications. In this research, we also build our models parallel based on Hadoop. Besides, many other different implementations have been proposed. For example, Ranger et al. implemented an in-memory Map/Reduce system for multi-core machines [91]; Plimpton et al. designed a specific Map/Reduce framework based on MPI for graph algorithms [87]; and He et al. built a Map/Reduce system using GPUs instead of CPU to simplify the GPU computing program design [39].

Figure 7.1: Map/Reduce Illustration (from Hadoop Tutorial of Yahoo!).
7.2 A Map/AllReduce Framework for ComSoc

As shown in the last section, we need to call Map/Reduce functions multiple times in an iterative algorithms. In this process, the Map/Reduce environment should be set up and the data are needed to be loaded from hard disks in the Map phrase and exchanged among machines in the Reduce phrase multiple times. These extra costs may slow down the whole model building process if there are many iterations. This is true for ComSoc as it resorts to MCMC [35] and gradient-based algorithms to build models.

Although some previous works have been done to adjust Map/Reduce for iterative algorithms, such as Twister [26], Spark [125] and Vowpal Wabbit [1], they are all independent from Hadoop and do not have all Map/Reduce properties, such as fault tolerance and distributed file system. Thus, we propose to design and implement a novel Map/Reduce framework that can be integrated with Hadoop closely and accelerate the iterative computing. Generally, we introduce an operation called AllReduce, which merge the pieces of results from all single machines. For programming, the programmer can firstly save data in memory during the Map phrase, then start the iterative process. Take logistic regression as an example, in each iteration, the sub gradients are computed by the data subset in each single machine and then call the AllReduce operation to obtain the whole gradients and update the model maintained in each machine. In this research, we implement this Map/AllReduce framework in Java and apply it to scale up the model building process of ComSoc. A open-source version has been developed ¹.

7.3 Map/AllReduce Implementations of ComSoc

The main idea of implementing ComSoc models based on Map/AllReduce is to compute the posterior distribution and resample the topic/community assignments (for ComSoc-MMSB, ComSoc-IT and ComSoc-Adap), or subgradients (for ComSoc-Adap) on partial data in each single machine and then aggregate pieces of results. To simply the discussion, we store the data in user-wise, where the data related to the same user will be loaded into the same machine.

We firstly take ComSoc-Adap as an example to illustrate the discussing the main idea of distributed implementation. Recall that sampling equations, such as Eq.(5.2) and Eq.(5.3), several statistics will be updated in the sampling process, including the user-topic count of each user in each single network, the user-network count, the topic count, the item-topic count of each item

¹https://github.com/purlin/MapAllReduce
and the topic assignment for user-item pair. We partition the data by users. For example, if there are $D$ machines, then the whole dataset will be divided into $D$ groups and each group will contain parts of users as well as their relationship and interaction data. As stated in [101], user-topic count, user-network count and topic assignment are entirely local to each user and thus do not need to be shared. They can be written to the disk after resampling. In addition, topic count and item-topic count change slowly, and hence, a delay in obtaining their up-to-date representations will not affect the sampler significantly. Furthermore, the number of topic statistics is small. Thus we can keep them in the memory permanently. These observations accomplish the local sampling without losing too much precision. In summary, the distributed implementation of each inference algorithm is also an iterative approach, but in each iteration, it resamples topic assignments/computes gradients in a distributed manner. As follows, we present the specific implementation of each ComSoc model. Before the start of iterations, the inference algorithm performs distributed initialization. Then, we use the Map operation to distribute the data into all single machines.

### 7.3.1 Implementation of ComSoc-MMSB and ComSoc-IT

As both ComSoc-MMSB and ComSoc-IT require neighbors’ statistics to update model and sample community assignments. At the beginning, we have to duplicate users’ statistics for their neighbors’ use. This can be done in a Map function. After that, related information, including users’ previous community assignments, statistics and neighbors’ statistics are all ready to update the current statistics and new community assignments. After the update is done, we can employ the AllReduce operation to broadcast to obtain the updated results. Formally, the process for inference of ComSoc-MMSB is as follows:

- **Map:** Map all data in the form of $(user, statistics)$ and duplicate the statistics for users’ neighbors.
- **Reduce:** begin the sampling iteration
  - Update the statistics
  - Resample the community assignments
  - Broadcast the updated statistics using AllReduce

### 7.3.2 Implementation of ComSoc-Adap

For ComSoc-Adap, the item-topic count of every items are maintained in every machines while the user-topic count of each user in each social network are duplicated for each related users and
maintained locally. For each user, the related user-topic count will be copied multiple times for its neighbors in each single network, and each duplicated statistics is used for the sampling of each neighbor. The second step is used to resample the topic assignments for user-item interactions and for user-user relations in each sub network. All information related to users and items are represented as \( \langle \text{key}, \text{value} \rangle \) pairs, where \text{key} refers to user identity and \text{value} can be statistics, topic assignments, item interactions and user relations information. Formally, the process is as follows:

- **Map**: Map all data in the form of \( \langle \text{user, statistics} \rangle \) and duplicate the statistics for users’ neighbors.
- **Reduce**: begin the sampling iteration
  - Update the statistics
  - Resample the topic assignments
  - Broadcast the updated statistics using AllReduce

### 7.3.3 Implementation of ComSoc-ML

Different from previous models, ComSoc-ML utilizes a boosting framework with gradient descent embedded. Considering the boosting procedure is sequential, we focus on paralleling the gradient descent part. The implementation is also an iterative algorithm. In each iteration, we design a Map function, where we used to compute sub gradients and update models distributive. Inside the Map function, we firstly load data into memory and then compute the gradients for the partial model. Consequently, we use AllReduce to broadcast the gradients and aggregate the partial results. Finally, we used the aggregated gradients to update model in boosting for next round. The process is as follows:

- **Begin boosting**
- **Map**: to build a base model
  - Store related data, including all triple constraints and the related users’ features
  - Compute the sub gradients
  - Broadcast using AllReduce
  - Update the base model
- **Compute the weight of the model on the master machine**
7.4 Efficiency Experiments

We compare our Map/AllReduce framework with standard Map/Reduce approach Hadoop on the efficiency of ComSoc implementations on the Tencent dataset. The cluster contains eight slave machines and a master, where each node contains two CPUs (8 cores, 2 GHz, 20M cache) and 64G RAM. The results can be found in Figure 7.2. We can observe that, Map/AllReduce has a better speedup ratio than Map/Reduce when the data can be cached into distributed memories. In addition, with more machines, the advance of Map/AllReduce becomes larger. In addition, Map/AllReduce is more suitable for gradient-based optimization algorithms.
CHAPTER 8

CONCLUSION AND FUTURE WORKS

Composite social network analysis is a new and exciting research area, which also has close relationships with real industry applications, including social recommendation, behavior targeting, and personalization etc. In this thesis, we have done the following works,

1. We surveyed social network analysis works, w.r.t. link prediction, community detection and user influence, and composite network mining w.r.t multi-relational network analysis and cross-network transfer learning.

2. We gave a formal definition for composite social network analysis inspired from real-world applications.

3. We presented four novel research objectives in composite social networks, and then design our solutions based on hieratical Bayesian models correspondingly,
   - Composite Network Structure Modeling (ComSoc-MMSB)
   - Modeling Dynamics of Composite Network and Network Co-evolution (ComSoc-IT)
   - Adaptive User Behavior Prediction (ComSoc-Adap)
   - Adaptive User Distance Measure (ComSoc-ML)

4. We designed a novel parallel computing framework called Map/AllReduce for ComSoc and derive efficient implementations.

**Discussions** This thesis focuses on exploiting knowledge from multiple single networks to benefit from the enriched data while taking care of the network differences. However, there are some situations where this idea is not applicable. Firstly, for those network-orient tasks, introducing multiple networks would not provide complementary knowledge but may bring unnecessary noises. If one would like to study users’ careers, then combining Twitter with LinkedIn would not help. Secondly, for those independent networks, such as QQ and Facebook, which share limited users and users on these networks have very different behaviors, combining them together cannot exchange useful knowledge but increases the model complexity and then raises the risk of overfitting. Finally, privacy issues may also affect the applications of this thesis work, as not all networks can be aligned.
Future Works  In the future, we plan to develop this exciting and fertile multidisciplinary area in the following two directions: (1). is there any new problems brought by composite social networks? and (2). how to extend the existed applications from single to composite social networks? Following these two questions, we state several potential research issues.

- New problems

1. How to measure users’ influence and track information flow cross-network? Considering that users can post their favorites and messages through different social networking applications and information can be transferred from one network to another, modeling the knowledge in a single network cannot retrieve the complete information flow and measure users’ true influence path.

2. How to measure and identify the differences and similarities among different social networks, such as the diameters, degree distributions, and number of communities etc? This can guide us to develop algorithms with composite information while prevent the negative impacts from network differences.

3. How to model the properties of the composite networks, such as the community structure, influence path, node and edge distribution, etc., by a mathematical model? This model can help us to build relationships among different networks.

4. How about the co-influence process between composite social networks and users’ online behaviors? Which parts of one domain will be influenced by others? What factors perform similar roles on evolving the different networks? This can offer us an insight of the networks’ interaction and promote the composite network applications.

5. How to link people across different networks? Although several works have been proposed to solve the entity resolution task, when personal information of a given user are not available, we may need to align users by exploiting the network structure.

- New applications

1. Composite social recommendation. As stated above, recommendation based on collaborative filtering may suffer from the sparsity problem. One way to solve this is to transfer knowledge from other domains. The relationships between users in social networks offer one alternative solution. In addition, people play different roles in different social networks, which means one user will receive different impacts from the neighbors in different networks. Thus, one kind of relations can only help the recommendation for
some but not all users. To exploit the power of composite social networks for recommendation, one should consider to give different weights on each social networks for different users.

2. Co-Transfer in composite networks. As shown above, several cross-network transfer learning approaches have been proposed. However, both the learning tasks in source and target networks are homogeneous, e.g., all on link prediction. Now, we can consider a more general idea, could we transfer knowledge cross-network to help heterogeneous tasks. For example, could we transfer knowledge for both community discovery and link prediction simultaneously? On one hand, if two users are in the same groups, then they may know each other with high probability. On the other hand, if users have more links with others in the same community, then this community is easier to be detected.

3. Composite social marketing. As different networks can reflect different users’ interests and different users may have different ranks in different networks. By considering the relational and interaction knowledge from multiple networks, we can find target users for social marketing more accurately.
REFERENCES


127


[123] Nicholas Jing Yuan, Fuzheng Zhang, Defu Lian, Kai Zheng, Siyu Yu, and Xing Xie. We know how you live: Exploring the spectrum of urban lifestyles. In *ACM Conference on Online Social Networks*, COSN ’13, New York, NY, USA, 2013. ACM.


