PROBABILISTIC MODELS FOR RECOMMENDATION IN SOCIAL NETWORKS

by

SeyedMohsen Jamali

M.Sc., Sharif University of Technology, 2007

B.Sc., Sharif University of Technology, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2013

APPROVAL

Name: SeyedMohsen Jamali

Degree: Doctor of Philosophy

Title of Thesis: Probabilistic Models for Recommendation in Social Networks

Examining Committee:

Dr. Joseph Peters
Chair

Dr. Martin Ester, Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Jian Pei, Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Oliver Schulte,
Associate Professor, Computer Science
Simon Fraser University
SFU Examiner

Dr. Irwin King,
Professor of Computer Science
Chinese University of Hong Kong
External Examiner

Date Approved: January 25th 2013

Partial Copyright Licence

Abstract

Recommender systems are becoming tools of choice to select the online information relevant to a given user. Collaborative filtering is the most popular approach to building recommender systems and has been successfully employed in many applications. However, collaborative filtering based approaches perform poorly for so-called cold start users. With the advent of online social networks, the social network based approach to recommendation has emerged. This approach assumes a social network among users and makes recommendations for a user based on the ratings of the users that have direct or indirect social relations with the given user. As one of their major benefits, social network based approaches have been shown to reduce the problems with cold start users. In this research we propose novel methods to address the recommendation problem in online social networks. To better understand the underlying mechanisms of user behavior in a social network, we first propose a model to capture the temporal dynamics of user behavior based on different effects influencing the behavior of users in rating items and creating social relations (e.g. social influence, social selection and transitivity of relations). Then we propose a memory based approach based on random walk models to perform recommendation in social networks. Matrix factorization is the most prominent model based approach for collaborative recommendation. We extend matrix factorization and propose a model that takes into account the social network as well as the rating matrix. Finally, we present a mixed membership community based model for recommendation in social networks based on stochastic block models. This model is capable of performing both rating and link prediction.

All methods have been experimentally evaluated and compared against state-of-the-art methods on real life data sets from Epinions.com, Flixster.com and Flickr.com. The Flixster data set has been crawled and published as part of the research during this thesis. Experimental results show that our proposed models achieve substantial quality gains compared to the existing methods.

To my love, ***Samaneh****!*

To my ***Parents!***

"We are leaving the age of information and entering the age of recommendation"

— *Chris Anderson,* *The long tail: Why the future of business is selling less of more, 2006*

Acknowledgments

I am very grateful to my supervisor, Dr. Martin Ester. During my PhD, I was blessed with having him as my supervisor. In my research, he is like a father to me. Martin has been extremely nice and kind to me. I have always benefited from his insightful comments and discussions. Not only he helped me with my research projects, but also Martin taught me how to do independent research, how to write papers, and how to define and tackle new problems.

I would also like to thank Samaneh Moghaddam for her great collaborations during my research projects. Together, we learned machine learning and how to employ machine learning in our projects. I feel extremely lucky to have such a great researcher as my lab-mate and my wife. She also did a very comprehensive and detailed proof reading on this thesis which much appreciate.

I like to acknowledge Dr. Jian Pei and Dr. Olive Schulte for their great comments and discussions during my research. Also, I would like to thank the examining committee for reading the thesis and providing valuable comments on my dissertation.

During my PhD, I was also blessed with collaborating with many friends and colleagues who I like to acknowledge in this dissertation: Bo Hu, Mohammad Tayebi, Tianle Huang, Jingbo Fu, Gholamreza Haffari, Bee-Chung Chen, Deepak Agarwal, Liang Zhang, Flavia Moser, etc. Also I want to acknowledge my advisors during my master studies in Sharif university, Dr. Hassan Abolhassani and Dr. Jafar Habibi. I owe the beginning of my research career to both of them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the rapidly growing amount of information available on the World Wide Web, it becomes necessary to have tools to help users to select the relevant part of online information. To satisfy this need, recommender systems have emerged, e.g. there are popular recommenders for movies[1], books[2], music[3], etc.

Recommender systems have become an important research area since the appearance of the first papers on collaborative filtering in the mid-90s [42, 96, 103]. There has been much work done both in the industry and academia on developing new approaches to recommender systems over the last decade [1]. The interest in this area still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users to deal with information overload and to provide them with personalized recommendations of content, services, etc. Examples of such applications include recommending books, CDs, and other products by Amazon [68], and movies by MovieLens [80]. Moreover, some of the vendors have incorporated recommendation capabilities into their commerce servers [93].

Recommender systems can be classified into content-based recommendation and collaborative recommendations. Content-based recommendation has its roots in information retrieval [1]. Collaborative recommendation does not depend on the content of the items and relies only on the past user behavior (ratings). Recommender systems emerged as an independent research area in the mid-90s when researchers started focusing on recommendation problems that explicitly rely on the ratings

[1] http://www.netflix.com

[2] http://www.amazon.com

[3] http://www.last.fm

structure [1]. Note that the main focus of this thesis is also on collaborative recommendation. In its most common formulation, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. Intuitively, this estimation is usually based on the ratings given by this user to other items and on some other information that will be formally described below. Once we can estimate ratings for the yet unrated items, we can recommend to the user the item(s) with the highest estimated rating(s).

Typically in a recommender system, we have a set of *users* and a set of *items*. Each user u rates a set of items by some values. The ratings expressed by users on items are stored in a rating matrix. The rating matrix is typically extremely sparse [1, 69]. The recommender has the task to predict the rating for user u on a non-rated item i or to recommend the top-N items with highest predicted ratings based on the known ratings. Note that "rating" is a general term and includes the real valued item ratings in Epinions[4] and Flixster[5] and binary rating values such as joining a community in Facebook[6] and LiveJournal[7] or adding a photo to your favorite list in Flickr[8]. In other words, any type of user behavior demonstrating his evaluation of an item can be formulated as a rating (binary or real values).

Generally two types of collaborative recommender systems have been investigated: Memory-based and Model-based approaches. Memory-based algorithms (traditional collaborative filtering [37]) explore the user-item rating matrix and make recommendations based on the ratings of item i by a set of users whose rating profiles are most similar to that of user u. Model-based approaches such as matrix factorization [99] learn the parameters of a model. Hence they do not need to explore the rating matrix and only store the model parameters. Model-based approaches are very fast after the parameters of the model are learned. The bottleneck for model-based approaches is the training phase, while in memory-based approaches there is no training, but the prediction (test) phase is slower.

It should be noted that although recommender systems and search engines seem to be similar, but they are fundamentally different. Both search engines and recommender systems are information filtering tools that retrieve the relevant piece of information for their users. In search engines the input is the query keywords and the output is the ranked list of results. In other words, users need

[4] www.epinions.com

[5] www.flixster.com

[6] www.facebook.com

[7] www.livejournal.com

[8] www.flickr.com

to know what they are looking for before using the search engine. Every user gets the same result (although modern search engines such as Google try to personalize the search results as much as possible). Users want to have personalized results, but are not willing to spend a lot of time to specify their personal information needs. Recommender systems, on the other hand, automatically identify information relevant for a given user, learning from available data.

Online social networks are playing an important role in shaping the behavior of users on the Web. Popular social networking services such as Facebook, Twitter[9], Flickr, and del.icio.us, are increasingly accepting more traffic and are turning into community spaces where users interact with their friends and acquaintances [4]. The availability of such rich data at large scales makes it possible to analyze user behavior at an individual level in order to understand user behavior at large. In particular, questions interpreting a user's action in the context of his online friends and correlating the actions of socially connected users, become highly interesting [4].

With the advent of online social networks, the social network-based approach to recommendation has emerged. This approach assumes a social network among users and makes recommendations for a user based on the ratings of the users that have direct or indirect social relations with the given user. The input data for recommendation in social networks is a social rating network. A Social Rating Network (SRN) is a social network in which each user expresses ratings on some items besides creating social relations to other users. Figure 1.1 illustrates a sample social rating network . In addition to the notion of a social rating network, we use the notion of *trust network* in order to emphasize the trust aspect of a social network which is most important in the context of recommendation. This notion has also been used in the literature [74, 36, 120], sometimes in the combined notion of social trust [69].

Collaborative filtering is most effective when users have expressed enough ratings to have common ratings with other users, but it performs poorly for so-called cold start user. Cold start users are new users who have expressed only a few ratings. Using similarity-based approaches, it is unlikely to find similar users since the cold start users only have a few ratings. Social network-based recommenders, however, can make recommendations as long as a new user is connected to a large enough component of the social network. More than 50% of users in existing real life data sets are cold start users (have less than 5 ratings[10]) [47, 74]. Therefore, addressing cold start users in a recommender system is very critical.

[9] www.twitter.com

[10] As stated in the literature [74, 36], users with less than 5 ratings are considered as cold start users.

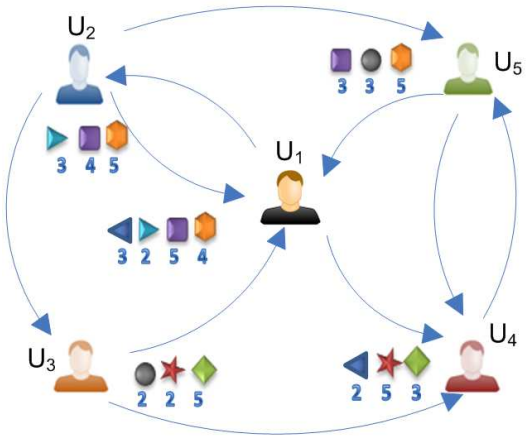


Figure 1.1: A sample social rating network.

Exploiting social networks in recommendation works because of the effects of selection and social influence that have been postulated by sociologists for a long time. Social selection [78] means that people tend to create social relations to people with similar attributes. Another fundamental property of social networks is that people tend to behave similarly to their friends. The process of social influence [33] in SRNs leads to people adopting the rating behavior of their friends. Social influence and selection together lead to social correlation, i.e., there is correlation between the behavior of a user and his neighbors. The increasing availability of online social network data has finally allowed a verification of these sociological models. The results of experiments in [26] and of similar work confirm that a social network provides an independent source of information which can be exploited to improve the quality of recommendations.

In this dissertation, we propose novel probabilistic models for recommendation in social networks. In Chapter 2 we formally introduce the problem definition for recommendation in social networks. Moreover, we review state-of-the-art methods on recommendation, in particular recommendation method in social networks.

Many effects influence the behavior of users in the evolution of a social rating network. Before digging deep into recommendation models, we first model the temporal dynamics of user behavior in social rating network to capture these effects and to better understand the underlying mechanisms of user behavior in a social network. The comprehensive set of effects influencing the user behavior and how we model them in a generative model is presented in Chapter 3. We perform experimental

studies on real life data sets from Epinions and Flickr to evaluate how well the proposed model captures the temporal dynamics of user behaviors.

We introduce TrustWalker in Chapter 4, random walk based method for recommendation ins SRNs. TrustWalker is a memory based approach and explores the network to find an estimated rating for a specific user on a specific item. We also extend TrustWalker to perform tasks such as top-N recommendation and link prediction. Link prediction is a recommendation problem specific to social networks in which the goal is to recommend a list of top users to a given user in order to create social relations to these recommended users. Experimental results on two real life data sets from Epinions and Flixster show that TrustWalker outperforms existing memory-based approaches, in particular for cold start users. The Flixster data set has been crawled and published as part of this research.

Memory-based approaches such as TrustWalker are slow in the test (prediction) phase, since they have to explore the social rating network for every single predictions. Model-based approaches, on the other hand, are very fast in prediction (although they need an additional time for the learning phase). Matrix Factorization is the most well know approach in model-based recommendation. In Chapter 5 we extend matrix factorization to take into account the social network in the recommendation process. Basically, we regularize the latent factors of a user by the latent factor of his direct neighbors in the social network. Regularization works particularly well for cold start users who do not have expressed many ratings and their latent factors can not be learned effectively based on their ratings only. Cold start items are also very important in recommender systems, in particular for industries since they want to be able to promote their new items. To address cold start items, we further extend our model and regularize the latent factor of items by the latent factors of their direct neighbors in an item graph. The item graph is constructed using simple properties of items such as product names in Epinions and movie genres and IMDB[11] ratings in Flixster. Again, we perform experiments on Epinions and Flixster to demonstrate the substantial improvement of recommendation quality in the proposed models, particularly for cold start users and items.

Social influence and selection together lead to the formation of communities of like-minded and well-connected users. Exploiting the clustering of users and items is one of the most important approaches for model-based recommendation. Users may belong to multiple communities or groups. In Chapter 6, we extend mixed membership stochastic block models [3] and introduce a generalized stochastic blockmodel (GSBM) that models not only the rating behavior of users but also models the

[11] www.imdb.com

creation of social relations. In other words, GSBM is a community based recommender that is capable of performing both rating prediction and link prediction. Experiments on Epinions and Flixster demonstrate the accuracy of the proposed GSBM for rating prediction as well as link prediction.

In Chapter 7, to present more insights on this thesis and the proposed models, we discuss some general questions such as: Given an SRN, how do we say whether social network-based recommendation works for this SRN or not? How do memory-based approaches for recommendation in social networks compare against model-based approaches? Finally we conclude the thesis in Chapter 8 and present some directions for future works.

Chapter 2

Introduction to Recommendation in Social Networks

In this chapter we formally define the recommendation problem and different classes of methods existing for recommendation. Then we discuss the approaches for evaluating recommender systems. Since the focus of this thesis is on collaborative recommendation, we review the state-of-the-art methods for collaborative filtering based recommendation, both memory-based and model-based approaches. Next, we formally introduce the problem definition for recommendation in social networks and review the state-of-the-art methods for recommendation in social networks.

2.1 Problem Definition in Recommender Systems

In recommender systems we have a set of users $\mathbb{U} = \{u_1, ... u_N\}$ and a set of items $\mathbb{I} = \{i_1, ... i_M\}$. The ratings expressed by users on items are given in a rating matrix $R = [r_{u,i}]_{N \times M}$. In this matrix $r_{u,i}$ denotes the rating of user u on item i. $r_{u,i}$ can be any real number, but often ratings are integers in the range $[1, 5]$[1]. In a social rating network, each user u has a set N_u of direct neighbors and $t_{u,v}$ denotes the value of social trust u has on v as a real number in $[0, 1]$. Zero means no trust and one means full trust. Binary trust networks are the most common trust networks (Facebook, Google+[2] Amazon[3], eBay[4], ...). The trust values are given in a matrix $T = [t_{u,v}]_{N \times N}$. Non-zero cells $t_{u,v}$ in

[1] Note that bookmarks and page hits can be formulated as binary rating values.

[2] plus.google.com

[3] www.amazon.com

[4] www.ebay.com

T denote the existence of a social relation from u to v. Note that T is asymmetric in general.

As discussed before, two general tasks can be defined for a recommender system: Rating prediction and top-N recommendation. In the following we formally define these two problems:

Rating Prediction:
Given a user $u \in \mathbb{U}$ and an item $i \in \mathbb{I}$ for which $r_{u,i}$ is unknown, compute the predicted rating of u on item i, $\hat{r}_{u,i}$, using the rating matrix R and the social network T.

Top-N Recommendation:
Given a user $u \in \mathbb{U}$ and the rating matrix R, recommend the top N most desired items that u has not rated yet.

The main focus of this thesis is on rating prediction. Using a rating prediction system, one can infer a top-N recommender by simply ranking all items according to their predicted rating. This approach is obviously not efficient but effective. There are more sophisticated ways to perform top-N recommendation that we discuss later in this thesis.

2.2 Classification of Recommender Systems

The predicted ratings of the not-yet-rated items can be estimated in many different ways using methods from machine learning, approximation theory, and various heuristics [1]. The commonly accepted formulation of the recommendation problem was first stated in [42, 96, 103] and this problem has been studied extensively since then. Recommender systems are usually classified according to their approach to rating estimation. In the next section, we will present such a classification that was proposed in the literature and will provide a survey of different types of recommender systems. Recommender systems are usually classified into the following categories, based on how recommendations are made [11]:

- Content-based recommendations: The user will be recommended items similar to the ones the user preferred in the past.
- Collaborative recommendations: The recommendation is performed based on the ratings expressed by people with similar tastes and preferences in the past. Collaborative filtering approaches exploit only the rating history of users and they do not consider the content features of items or profile attributes of users.
- Hybrid Methods: These methods combine collaborative and content-based methods.

2.2.1 Content-based Methods

In content-based recommendation methods, the predicted rating $\hat{r}_{u,i}$ of item i for user u is estimated based on the ratings $r_{u,j}$ assigned by user u to items $j \in I$ that are similar to item i. For example, in a movie recommendation application, in order to recommend movies to user c, the content-based recommender system tries to understand the commonalities among the movies user u has rated highly in the past (specific actors, directors, genres, subject matter, etc.) [1].

Content-based recommendation systems analyze item descriptions to identify items that are of particular interest to the user. Also the similarity of users is computed using user profiles.

The content-based approach to recommendation has its roots in information retrieval [10] and information filtering [13] research. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of several text-based applications, many current content-based systems focus on recommending items containing textual information, such as documents, Web sites (URLs), and Usenet news messages. The improvement over the traditional information retrieval approaches comes from the use of user profiles that contain information about users tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly-learned from their transactional behavior over time [1].

As stated earlier, content-based systems recommend items similar to those that a user liked in the past [63, 83, 91]. In particular, various candidate items are compared with items previously rated by the user and the best matching item(s) are recommended.

Too many information is required for the content-based approach to perform high quality recommendations. Descriptive features of users and profiles of users should be available to compute the similarities. Having to deal with the content features of users and items makes content-based approaches less efficient than collaborative recommenders where only the rating history is exploited. Besides, many rating systems do not grant permission to access the private contents of users and items by recommenders.

2.2.2 Collaborative Methods

Unlike content-based recommendation methods, collaborative recommender systems (or collaborative filtering systems) try to predict the rating of items for a particular user based on the ratings already expressed by this users and other users. Note that the item description or users profiles are not exploited in collaborative recommenders.

For example, in a movie recommendation application, in order to predict the rating of user u on movie i, the collaborative recommender system tries to find the users similar to u, i.e., other users that have similar tastes in movies (rate the same movies similarly). Then, the aggregate of ratings expressed by the users with similar rating pattern is computed as the predicted rating.

According to [19], algorithms for collaborative recommendations can be grouped into two general classes: memory-based (or heuristic-based) and model-based.

Memory-based algorithms [19, 28, 85, 96, 103] essentially are heuristics that make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating $r_{u,i}$ for user u and item i is usually computed as an aggregate of the ratings of some other (usually, the N most similar) users for the same item i.

Model-based algorithms [16, 43, 73, 90, 109, 115, 100, 14, 60, 58] use the collection of ratings to learn a model, which is then used to make rating predictions. After learning the model, we do not need to have access to the entire ratings anymore. We can store the model parameters and use them to compute the predictions.

In contrast with memory-based approaches, model-based approaches have a learning phase which could be time consuming. On the other hand, memory-based approaches are slower in the prediction part since they have to explore the whole ratings heuristically, while model-based approaches are pretty fast since they only use the model parameters to compute the predicted rating.

2.2.3 Hybrid Methods

Several recommendation systems use a hybrid approach by combining collaborative and content-based methods, which helps to avoid certain limitations of content-based and collaborative systems [11, 12, 24, 92, 102, 105, 109]. Different ways to combine collaborative and content-based methods into a hybrid recommender system can be classified as follows [1]:

- implementing collaborative and content-based methods separately and combining their predictions,
- incorporating some content-based characteristics into a collaborative approach,
- incorporating some collaborative characteristics into a content-based approach, and
- constructing a general unifying model that incorporates both content-based and collaborative characteristics.

2.3 Evaluation of Recommendation systems

Typically, recommenders are evaluated based on the quality of their predictions in terms of prediction errors. Also, leave-one-out method is used to evaluate recommendation systems [36, 74, 101]. In the leave-one-out method, we withhold a rating and try to predict it using the trust network and the remaining ratings.

Two most common metrics for computing the error of predictions are Root Mean Squared Error (RMSE) [115, 58], and Mean Absolute Error (MAE) [41]. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{(u,i)|R_{u,i}} (r_{u,i} - \widehat{r}_{u,i})^2}{|\{(u,i)|R_{u,i}\}|}} \tag{2.1}$$

In the above Equation, $R_{u,i}$ is a boolean showing whether u has a rating on i in our data set, and $r_{u,i}$ and $\widehat{r}_{u,i}$ denote the actual and recommended rating, respectively. The smaller the value of *RMSE*, the more precise a recommendation. MAE is defined as follows:

$$MAE = \frac{\sum_{(u,i)|R_{u,i}} |\widehat{r}_{u,i} - r_{u,i}|}{|\{(u,i)|R_{u,i}\}|} \tag{2.2}$$

Besides the evaluation of recommenders in terms of measuring their prediction errors, some other aspects of recommendation quality have also attracted attentions in the literature. Examples of these aspects are explainability, serendipity, and diversity.

Explainability means that the recommender system is able to explain how it predicted the rating. There is now a growing recognition that recommender systems must be able to explain and justify the recommendations in order to help users to understand why particular items have been suggested. Some works in the literature have addressed the explainability issue and tried to propose recommendation models that can explain their predictions [110, 107, 40, 117]. To explore how explanation interfaces should be implemented, authors of [40] present a model for explanations based on the user's conceptual model of the recommendation process. Vig et al. [110] introduced tagsplanations, which are explanations based on community tags. This paper investigates the properties of a good explanation in a movie recommender system. Tintarev [107], in her PhD thesis, suggests seven criteria for evaluation of explanations in recommender systems, followed by an attempt to define the properties of a useful explanation using a movie review corpus and focus groups.

Another aspect of recommendation quality is diversity of results in recommendation. This aspect is most related to the top-N recommendation. Diversity means that the list of top-N recommendation items are from different contexts and topics and they are not all similar to each other. In other words,

there is topic diversification among the recommended items [122]. There has been some works in the literature addressing serendipity and diversification [122, 117, 84]. Serendipity in recommendation means the extent of unexpectedness in the list of recommended items. Serendipity is in close relation with diversity. Authors of [84] propose metrics, unexpectedness and unexpectedness$_r$, for measuring the serendipity of recommendation lists produced by recommender systems. The basic idea of their metrics is that unexpectedness is the distance between the results produced by the method to be evaluated and those produced by a primitive prediction method [84]. Ziegler et al. [122] introduced an intra-list similarity metric to assess the topical diversity of recommendation lists and a topic diversification approach for decreasing the intra-list similarity. Yu et al. [117] introduced the notion of explanation-based diversification to address the problem of over-specialization in item recommendations. This paper leverages the reason for which a particular item is being recommended, i.e., explanation, for diversifying the results, without the need to access the attributes of the items.

2.4 Related Work on Collaborative Recommendation

Since the main focus of this thesis is on collaborative approaches, we devote this section to discuss the state-of-the-art memory-based and model-based approaches for collaborative recommendations.

2.4.1 Memory-based Approaches for Collaborative Recommendation

Memory-based approaches are essentially heuristics that make rating predictions based on the entire collection of previously rated items by the users. In this chapter we introduce two general approaches for memory-based collaborative filtering (CF): user-based collaborative filtering, and item-based collaborative filtering. Also, at the end of this chapter, we introduce a memory-based approach that is random walk based method for alleviating the sparsity problem in collaborative filtering.

User-based Collaborative Filtering

User-based collaborative filtering is the method of making predictions about the interests of a user by collecting taste information from similar users. The underlying assumption of user-based CF approach is that those who agreed in the past tend to agree again in the future. For example, a collaborative filtering recommendation system for television tastes could make predictions about whether a user likes a television show given a partial list of that user's tastes and also the opinion of

other users with similar taste.

In a rating system, to predict the rating of user u on target item i, collaborative filtering methods make recommendations based on the ratings of item i by a set of users whose rating profiles are most similar to that of user u. The predicted rating is computed as follows [37]:

$$\hat{r}_{u,i} = \frac{\sum sim_{u,v} r_{v,i}}{\sum sim_{u,v}} \tag{2.3}$$

In the above equation, the summation is computed over top N users that are most similar to the user u. $\hat{r}_{u,i}$ denotes the predicted rating. Note that some users are optimistic and some users are pessimistic in their ratings. This leads to a bias in users' ratings. To avoid this the prediction is revised as follows:

$$\hat{r}_{u,i} = \overline{r}_u + \frac{\sum sim_{u,v}(r_{v,i} - \overline{v})}{\sum sim_{u,v}} \tag{2.4}$$

where $\overline{u}$ is the average of ratings expressed by u. The above equation subtracts each user rating by the average of ratings expressed by that user and computes an estimated deviation from the average rating for user u.

There are a number of different ways to compute the similarity between users. In the following we present two most common approaches: cosine-based similarity and correlation-based similarity.

- **Cosine-based Similarity**: In this case, two users are considered as two vectors in the N dimensional item-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the $N \times M$ ratings matrix, similarity between users u and v denoted by $sim_{u,v}$ is given by:

$$sim_{u,v} = cos(\overrightarrow{u}, \overrightarrow{v}) = \frac{\overrightarrow{u} \cdot \overrightarrow{v}}{\| \overrightarrow{u} \|_2 \times \| \overrightarrow{v} \|_2} \tag{2.5}$$

  where "$\cdot$" denotes the dot product of the two vectors and "$\|\|_2$" represent the L_2 norm of a vector.

- **Correlation-based Similarity**: In this case, similarity between two users u and v is measured by computing the Pearson-r correlation $corr_{u,v}$. Let the set of items rated by both user u and v denoted by $I_{u,v}$ then the correlation similarity is given by

$$corr_{u,v} = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \overline{r}_v)^2}} \tag{2.6}$$

Here, $\overline{r}_u$ denotes the average of ratings expressed by u.

Challenges of User-based CF

User-based collaborative filtering systems have been very successful in the past, but their widespread use has revealed some potential challenges [101] such as:

- **Sparsity**. In practice, many commercial recommender systems are used to evaluate large item sets (e.g., Amazon.com recommends books and CDnow.com recommends music albums). In these systems, even active users may have purchased well under 1% of the items (1% of 2 million books is 20000 books). Accordingly, a user-based collaborative filtering recommender system may be unable to make any prediction for a particular user [101]. Enough rating should have been expressed by users for the algorithm to be able to compute the similarity and find similar users. Therefore, user-based collaborative filtering methods are not efficient for cold start users who have rated only a few items.

- **Scalability**. User-based CF algorithms require computation that grows with both the number of users and the number of items. With millions of users and items, a typical web-based recommender system running existing algorithms will suffer serious scalability problems.

Item-Based Collaborative Filtering

In this section we study a class of item-based recommendation algorithms for producing predictions to users. Unlike the user-based collaborative filtering algorithm discussed in previous section, the item-based approach looks into the set of items the target user has rated and computes how similar they are to the target item i and then selects k most similar items $\{i_1, i_2, ..., i_k\}$. At the same time their corresponding similarities $\{sim_{i,i_1}, ..., sim_{i,i_k}\}$ are also computed. Once the most similar items are found, the prediction is then computed by taking a weighted average of the target user's ratings on these similar items. Computation of the similarity of two items is similar to the computation of user similarities in user-based CF.

The cosine-based similarity of items i and j is computed as follows [101][68]:

$$sim_{i,j} = cos(\overrightarrow{i}, \overrightarrow{j}) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{\| \overrightarrow{i} \|_2 \times \| \overrightarrow{j} \|_2} \tag{2.7}$$

where $\overrightarrow{i}$ is the vector in the N dimensional user-space representing ratings expressed on i.

The Pearson correlation is computed as follows:

$$corr_{i,j} = \frac{\sum_{u \in U_{i,j}} (r_{u,i} - \overline{r}_i)(r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U_{i,j}} (r_{u,i} - \overline{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \overline{r}_j)^2}} \tag{2.8}$$

where $U_{i,j}$ denotes the set of user who have rated both items i and j. There is also another way to compute the similarity and it has been shown to result in higher quality experimental results [30]. This similarity measure is called *Adjusted Cosine Similarity*:

$$corr_{i,j} = \frac{\sum_{u \in U_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in U_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \overline{r}_u)^2}} \tag{2.9}$$

Item-based collaborative filtering also has challenges with handling the sparsity of the rating data, specially for cold start users and items. It should be noted that since the number of users is much more than the number of items in most rating data sets, the item-based CF is more scalable than the user-based CF. The reason is that in user-based CF we have to compute the similarity for user pairs while in item-based CF we compute the similarity for item pairs.

2.4.2 Model-based Approaches for Collaborative Recommendation

Model-based algorithms [16, 43, 73, 90, 109, 115, 100, 14, 60, 59] use the collection of ratings to learn a model, which is then used to make rating predictions. After learning the model, we do not need the entire ratings anymore. We can store the model parameters to compute the predictions.

In contrast with memory-based approaches, model-based approaches have a learning phase which could be time consuming. On the other hand, memory-based approaches are slower in the prediction part since they have to explore the whole ratings heuristically, while model-based approaches are pretty fast since they only use the model parameters to compute the predicted rating.

In this section we review some of the state-of-the-art model-based approaches.

Clustering-based Recommendation

A standard model-based collaborative filtering algorithm uses k-means to cluster similar users. Given a set of user profiles, the space can be partitioned into k groups of users that are close to each other based on a measure of similarity. The discovered user clusters are then applied to the user-based neighborhood formation task, rather than individual profiles used in the user-based collaborative filtering. To make a recommendation for a target user u and target item i, we select a

neighborhood of user clusters that have a rating for i and whose aggregate profile v_k is most similar to u. This neighborhood represents the set of user segments that the target user is most likely to be a member, based on a measure of similarity. For this task, we use Pearsons correlation coefficient. We can now make a prediction for item i as described in user-based collaborative filtering, where the neighborhood is the set of user cluster aggregate profiles most similar to the target user.

To contrast the user-based collaborative filtering and the clustering-based collaborative filtering, we should note that user-based CF is like a k-nearest neighbors approach, while clustering-based CF is like k-means approach. The clustering-based approach improves the quality of recommendation for cold start users as long as we can assign them to one of the clusters.

Association Rule-based Recommendation

Association rule mining is a common technique for performing market basket analysis. The intent is to gain insight into customers buying habits and discover groups of products that are commonly purchased together. As an example, an association rule may show that 98% of all customers that purchase frozen pizza also purchase soda. Association rules capture relationships among items based on patterns of co-occurrence across transactions. In [100], this method has been adapted to the context of collaborative filtering. Considering each user profile as a transaction, it is possible to use the Apriori algorithm [2] and generate association rules for groups of commonly liked items.

To make a recommendation for a target user profile u, we create a set of candidate items C such that an association rule r exists of the form $X \subseteq u \Rightarrow i \in C$ where i is an unrated item in the profile u [100]. In practice, it is not necessary to search every possible association rule given u. It is sufficient to find all frequent item sets $X \subseteq u$ and base recommendations on the next larger frequent itemsets $Y \supset X$ where Y contains some item i that is unrated in u.

Rule-based recommendation is typically used in the domain where the input data is binary ratings (e.g., user purchase history) and these models are not suitable to the cases where actual ratings exist. Also, rule-based recommendation is used for top-N recommendation rather than for the prediction task in recommenders.

Random Walk based Method for Alleviating the Sparsity Problem

Authors of [115] propose a novel item-based algorithm, RandomWalk recommender, that first infers transition probabilities between items based on their similarities and models finite length random walks on the item space to compute predictions. This method is especially useful when training data

is less than plentiful, namely when typical similarity measures fail to capture actual relationships between items. Aside from the proposed prediction algorithm, the final transition probability matrix computed in one of the intermediate steps can be used as an item similarity matrix in typical item-oriented approaches.

The experimental results in [115] show that the quality of results of RandomWalk recommender is higher than the regular item-based collaborative filtering.

In [115], they have an item graph on which they perform the random walk. This graph is independent of users, and hence they do not perform random walks on users. Moreover, the random walks are performed only starting from the items already rated by a user. This will lead to recommending items similar to items already rated by the user. Hence the diversification of the results of this approach is limited and the results have low serendipity.

Matrix Factorization based Recommendation

Matrix Factorization (MF) is one of the common techniques for model based recommendation. In MF, each user and each item has a K dimensional latent factor vector [58]: the latent factor of user u is denoted by U_u and stored in the uth row of user factor matrix U. The latent factor of item i is denoted by V_i and stored in the ith row of item factor matrix V.

In order to learn the latent factors of users and items, [99] employs matrix factorization to factorize the user-item matrix into product of user and item latent factors. The conditional probability of the observed ratings is defined as:

$$p(R|U,V,\sigma_R^2) = \prod_{u=1}^{N}\prod_{i=1}^{M}\Big[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_r^2\Big)\Big]^{I_{u,i}^R} \tag{2.10}$$

where $\mathcal{N}(x|\mu,\sigma^2)$ is the normal distribution with mean μ and variance σ^2, and $I_{u,i}^R$ is the indicator function that is equal to 1 if u has rated i and equal to 0 otherwise. The function $g(x)$ is the logistic function $g(x) = 1/(1+e^{-x})$, which bounds the range of $U_u^T V_i$ within [0,1]. Also, zero mean Gaussian priors are assumed for user and item factors:

$$p(U|\sigma_U^2) = \prod_{u=1}^{N}\mathcal{N}(U_u|0,\sigma_U^2\mathbf{I}),\ p(V|\sigma_V^2) = \prod_{i=1}^{M}\mathcal{N}(V_i|0,\sigma_V^2\mathbf{I}) \tag{2.11}$$

Now, through a Bayesian inference, the posterior probability of the latent variables U and V can be obtained as follows:

$$p(U,V|R,T,\sigma_R^2,\sigma_U^2,\sigma_V^2) \propto p(R|U,V,\sigma_R^2)p(U|\sigma_U^2)p(V|\sigma_V^2)$$
$$= \prod_{u=1}^{N}\prod_{i=1}^{M}\left[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_r^2\Big)\right]^{I_{u,i}^R} \times \prod_{u=1}^{N}\mathcal{N}\Big(U_u|0,\sigma_U^2\mathbf{I}\Big) \times \prod_{i=1}^{M}\mathcal{N}\Big(V_i|0,\sigma_V^2\mathbf{I}\Big) \quad (2.12)$$

The corresponding graphical model is presented in Figure 2.1. Using the above equation, we can learn the latent factors of users and items purely based on the user-item rating matrix.

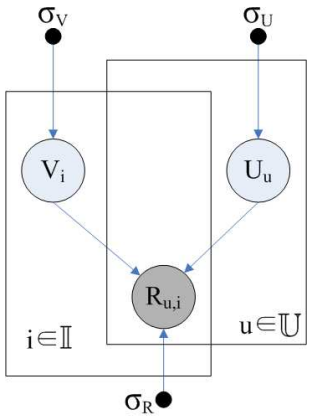


Figure 2.1: Graphical Model representing matrix factorization technique.

The matrix factorization based method has been recently extended to incorporate other factors in the factorization of the rating matrix: such as modeling at multiple scales [14], incorporating user with similar rating patterns [58], and taking the timestamps into account [59]. The last approach [59] won the Netflix grand $1M prize[5].

2.5 Social Networks Used in Recommender Systems

In this section, we elaborate on the characteristics of the social networks that can be employed for prediction quality improvement in recommender systems.

Throughout this dissertation, we use the terms social network and trust network. A social network is a network among users in which users are interconnected through social relations such as

[5]http://www.netflixprize.com/

friendship [27]. On the other hand, a trust network is a network of users in which users connect to others based on the trust they have on other user's opinions [21]. Members of a trust network may not actually know each other in real life, but they know of each other's opinions and expertise. Trust can be considered as one aspect of social relations. In fact, in social network based recommendation, we are emphasizing on the trust aspects of social relations, and that is why we use the terms social network and trust network interchangeably.

Social relations can be explicit or implicit. For example, the social relations in online social networking services such Facebook and Twitter are explicitly expressed by the users. On the other hand, users in content generation Websites such as Wikipedia are allowed to interact with other users by commenting on each other's walls. These types of interactions form an implicit social network [26]. All the datasets used in this thesis include explicit social networks that are either trust relations (Epinions), or social relations (Flixster).

The social relations in a social network can be directed or undirected. Friendship based networks (e.g. Flixster) are mainly undirected, since friendship is a bidirectional relation. On the other hand, networks based on trust or reputation (e.g. Epinions, Twitter) are directed. Note that even in directed networks, a relation can be bidirectional, meaning that it exists in both directions.

Social relations can be associated with weights that indicate the strength of a relation. Typically, online social networks do not explicitly ask user to put a with on their relation with other users. However, social networks such as Orkut have some ways to ask for the strength of a relation.

All the models we propose in this dissertation are capable of handling directed, undirected, weighted and un-weighted social networks. We do not focus on how the social networks are extracted (implicit or explicit), and we assume that the social network is given as an input in this thesis. In the next section, we review some related work on recommendation in social networks.

2.6 Related Work on Recommendation in Social Networks

With the advent of online social networks, the social network based approach to recommendation has emerged. This approach assumes a social network among users and makes recommendations for a user based on the ratings of the users that have direct or indirect social relations with the given user. Social network based recommendation methods assume as input a so-called Social Rating Network (SRN), i.e., a social network in which each user expresses ratings on some items besides creating social relations to other users. Figure 2.2 illustrates a social rating network.

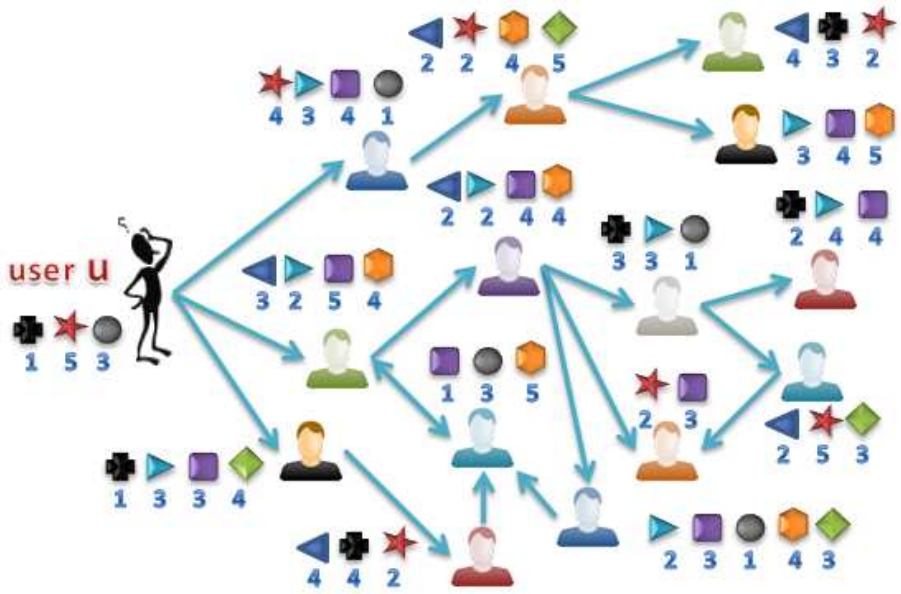


Figure 2.2: Illustration of a social rating network used in recommender systems. The ratings are shown below the item icon beside the user.

In this section we introduce existing related work on social network-based (trust-based) recommendation. We can distinguish two types of trust: Explicit trust and Implicit trust. Explicit trust denotes the trust values explicitly indicated by users, while implicit trust is the trust value inferred from some evidence such as feature similarity of users or email exchange among two persons. In this dissertation, we only consider the explicit trust indicated by users. In the case of explicit trust, we have direct trust and indirect trust. Direct trust is the trust value explicitly indicated by users, but indirect trust is the trust inferred from direct trust using transitivity of trust. How to compute indirect trust is one of the main issues of trust models. Two different approaches can be distinguished for trust computation: Model-based [97, 71] and Memory-based [74, 36, 120, 67]. In Model-based approaches, a model with its parameters will be learned to compute indirect trust. However, in Memory-based approaches, no model is being learned and normally exploration and heuristics are being used.

Most existing approaches for social network-based recommendation are memory-based approaches. They compute a neighborhood of trusted users in the social network who have rated the target item. Then they aggregate the ratings expressed by these trusted users weighted by trust

values to compute a rating prediction. The main challenge in these approaches is computing the indirect trust between pairs of users and how the trust propagates in the trust network. In this section we review some of the methods proposed in the literature for computing the indirect trust. Recently some model-based approaches have also been proposed for recommendation in social network. We discuss these models at the end of this section.

2.6.1 TidalTrust

TidalTrust [36] performs a modified breadth first search in the trust network to compute a prediction. Basically, it finds all raters with the shortest path distance from the source user and aggregates their ratings weighted by the trust between the source user and these raters. To compute the trust value between user u and v who are not directly connected, TidalTrust aggregates the trust value between u's direct neighbors and v weighted by the direct trust values of u and its direct neighbors as follows:

$$t_{u,v} = \frac{\sum_{w \in N_u} t_{u,w} t_{w,v}}{\sum_{w \in N_u} t_{u,w}} \tag{2.13}$$

In the above equation, $t_{u,v}$ denotes the trust value from u to v. N_u denotes the set of neighbors of u. Note that if any of the trust values $t_{w,v}$ is an indirect trust value, they will be computed recursively. After computing all the trust values between u and the raters at the shortest path, the predicted rating is computed as follows:

$$\hat{r}_{u,i} = \frac{\sum_{v \in raters} t_{u,v} r_{v,i}}{\sum_{v \in raters} t_{u,v}} \tag{2.14}$$

Since TidalTrust only uses information from raters at the nearest distance, it may loose a lot of valuable ratings from users a little further apart in the network.

2.6.2 MoleTrust

Authors of [74] introduce MoleTrust. The ideas used in MoleTrust and TidalTrust are similar. But MoleTrust considers all raters up to a *maximum-depth* given as an input. Note that *maximum-depth* is independent of any specific user and item. Also, to compute the trust value between u and v, MoleTrust performs a backward exploration. It means that the trust value from u to v is the aggregation of trust values between u and users directly trusting v weighted by the direct trust values [75]:

$$t_{u,v} = \frac{\sum_{x \in N_u^-} t_{u,x} t_{x,v}}{\sum_{x \in N_u^-} t_{u,x}} \tag{2.15}$$

where N_u^- is the set of in-links of v. In other words, N_u^- is the set of users for whom v is a direct neighbor. The predicted rating is computed in a similar way to that of TidalTrust.

In MoleTrust a fixed maximum-depth is used. If this maximum depth is set to a very small value, the likelihood of finding a rater will be very low. On the other hand, having a large maximum depth lead to incorporate the noisy information from users far way from the user u. It should be noted that both TidalTrust and Model trust do not distinguish different lengths of shortest path between the user u and the raters. Raters farther away from the user u should have smaller influence on the prediction.

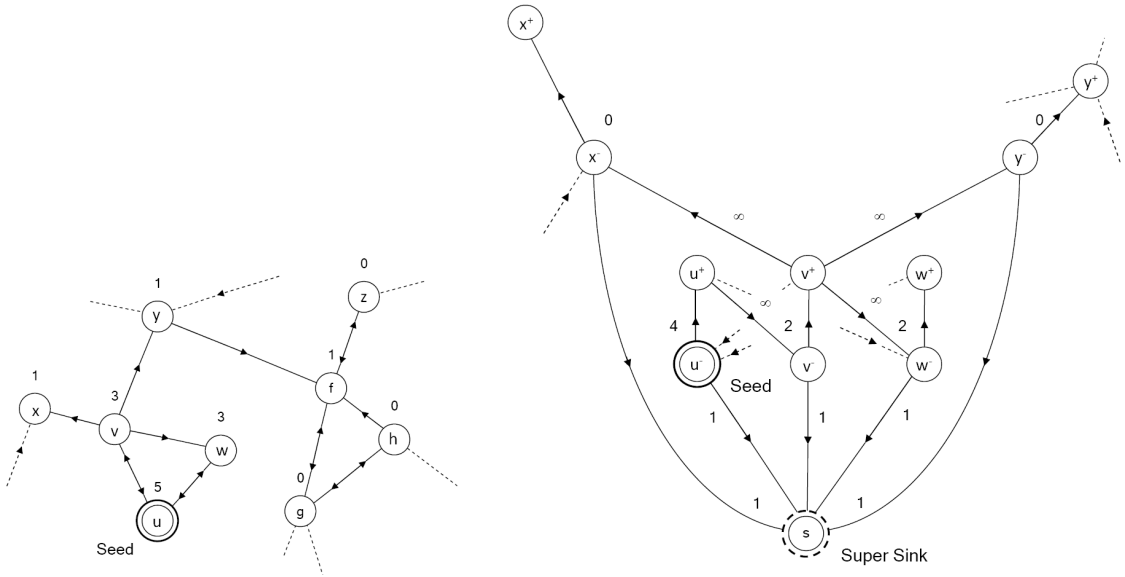
2.6.3 Advogato

The Advogato [67] maximum flow trust metric has been proposed in order to discover which users are trusted by members of an online community. The input for Advogato is given by an integer number n, the number of members to trust and the source (seed) user x for whom we like to find trusted users.

Capacities are assigned to every node in the network, based upon the shortest path from u to them. Hereby, the capacity of the seed itself is given by the input parameter n mentioned before, whereas the capacity of each successive distance level is equal to the capacity of the previous level l divided by the average out-degree of trust edges extending from l. The trust graph obtained hence contains one single source and multiple sinks, i.e., all nodes other than the seed. Capacities constrain nodes. In order to apply Ford-Fulkerson maximum integer network flow [31], the underlying problem has to be formulated as single-source/single-sink, having capacities on the edges rather than capacities on nodes.

Through a conversion algorithm, the graph is converted to a new graph where capacities are assigned to edges. A new node, the "super-sink" is added to serve as a single sink for the network flow algorithm. Each node x is split into two nodes, $x-$ and $x+$. For a node x with capacity c, an edge is added from $x-$ to $x+$ with capacity $c-1$. For each edge from s to t in the original graph, an infinite capacity edge from $s+$ to $t-$ is added to the new graph. Finally, from each node x, we add a unit capacity edge from $x-$ to the super-sink node [67]. Figure 2.3 depicts a sample network, and the network constructed for input $n = 5$. Figure 2.3(a) shows the original graph with capacities assigned to the nodes and Figure 2.3(b) demonstrates the converted graph to be used as the input of

Ford-Fulkerson algorithm.



(a) The original graph with capacities on nodes.

(b) The converted graph with capacities on edges.

Figure 2.3: A sample input graph for the Advogato trust metric [67].

After running the maximum flow algorithm, the nodes who have flow to the super-sink are considered to be the top-trusted users. It is easy to see that the number of such nodes is always n.

To assign capacities to the edges of the network, they need to transform the network, so it needs to know the whole structure of the network. Moreover, it only computes the nodes to trust and does not compute different degrees of trust. Since the number of users to trust is independent of users and items and there is no distinction between the trusted users, this approach is not appropriate for trust-based recommendation. However, this trust metric can be used to find top trusted users in a social network.

2.6.4 AppleSeed

AppleSeed has been proposed in [120] as part of a PhD thesis. In contrast to Advogato, being inspired by maximum network flow computation, the basic intuition of Appleseed is motivated by spreading activation models. Source node u is activated through an injection of energy e, which is then fully propagated to other nodes along edges proportional to the weight of the edge [121]. After the energy is propagated, nodes which receive more energy are higher ranked in terms of trust.

AppleSeed considers the trust to be additive. If there are many weakly trusted paths between two users, this pair of users will obtain a high trust value, which is not intuitive.

2.6.5 Model-based approaches for recommendation in SNs

Recently, the model-based approach for recommendation in social rating networks has been investigated [69][71]. These methods exploit the matrix factorization technique to learn latent factors for users and items from the observed ratings.

Ma et al. [71] developed a factor analysis method based on the probabilistic matrix factorization. In their model, SoRec, they consider three sets of latent features: U for users, V for items, and F for factors. They factorize the rating matrix R using latent item features V and latent user features U. On the other hand, they factorize the trust matrix T using latent user features U and latent factor features F. They assume a factor latent vector for each user. The graphical model for SoRec is presented in Figure 2.4.

As stated in in their next paper [69], although the users' social network is integrated into the recommender systems by factorizing the social trust graph, the real world recommendation processes are not reflected in the model. Two sets of different feature vectors are assumed for users which makes the interpretability of the model very hard. Moreover, experiments in [69] show that the more recent STE model outperforms the SoRec model in terms of the RMSE.

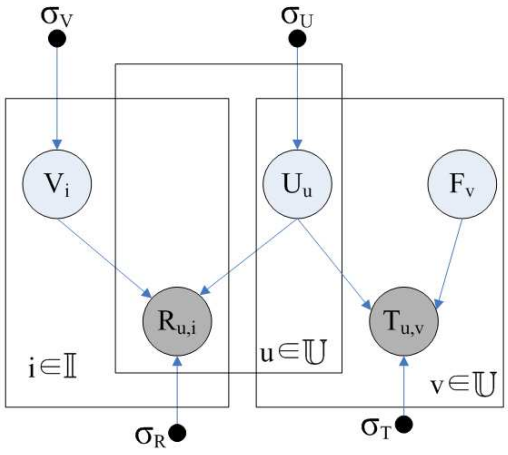


Figure 2.4: The graphical model representing SoRec [71].

As mentioned above, the same authors proposed a matrix factorization approach for social

network-based recommendation, called Social Trust Ensemble (STE) [69]. Their method is a linear combination of basic matrix factorization approach [99] and a social network-based approach. The graphical model for their proposed model is illustrated in Figure 2.5[6]. The predicted rating of user u on item i is as follows:

$$\hat{R}_{u,i} = g(\alpha U_u^T V_i \ + \ (1-\alpha) \sum_{v \in N_u} T_{u,v} U_v^T V_i) \tag{2.16}$$

where parameter α controls the effects of neighbors on the estimated rating.

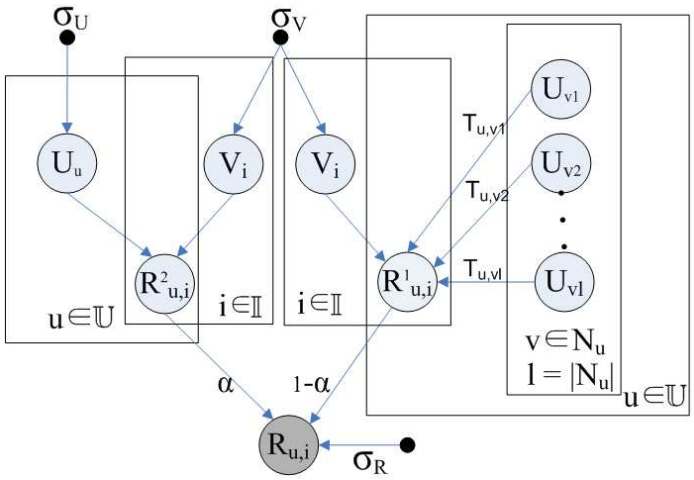


Figure 2.5: The Social Trust Ensemble (STE) model [69].

Experiments show that their model outperforms the basic matrix factorization based approach and existing trust-based based approaches. However, in their model, the latent factors of direct neighbors of u affect the ratings of u instead of affecting the latent factor of u. This model does not handle trust propagation. Also, since the procedure of learning latent factors of users is based on observed ratings, enough ratings should have been expressed by users so that the model can learn the latent factors of the user effectively. This issue make the model less effective for cold start users.

The authors of the STE model also proposed another model in [70]. They define a loss function that regularizes the latent factors of users by the sum of the latent factors of their neighbors. The loss function in [70] is similar to the objective function in SocialMF.However, unlike our SocialMF,

[6]It should be noted that model in Figure 2.5 is different from the graphical model presented in [69], but it correctly represents the joint probability distribution actually computed for the STE model.

this model is not a probabilistic generative model. More importantly, since this model uses a sum function for regularization (instead of the mean function used in SocialMF), it penalizes users with lots of social relations more than other users. Note that this paper also incorporates not only trust relations but also distrust relations in recommendation. In a very recent paper [72], the same authors extend their model from [70] and our SocialMF model [49] by taking into account the rating pattern similarity of users to put weights on the regularization terms in their model.

Yang et al. [114] propose a matrix factorization based model that factorizes both the rating matrix and the social network. Basically, the rating matrix is decomposed into the product of user latent factors and item latent factors. Also, the social network is decomposed into the product of the user latent factors. Note that the social network in this paper is considered to be undirected and the user latent factors is shared between decomposition of ratings and links.

2.6.6 Other methods

There has been some other methods addressing the trust-based recommendation. Andersen et al. [5] present a set of axioms for trust-based recommender systems and analyzes which combination of these axioms can be satisfied simultaneously. In the context of this discussion, a simple random walk method for binary (+1,-1) ratings is presented. Authors of [5] do not perform any experimental evaluation or comparison to other methods.

Trust has been defined and used in a different way by [88]. They extract the social network from the similarity of users' profiles, which is not providing additional information as is provided by trust network. They have two definitions for trust: profile level trust, which is a global reputation metric; and item level trust, which measures the trustworthiness of a user according to his recommendations for an item rather than a user-user local trust metric. They use the trust values to filter raters, and they aggregate the ratings weighted by a combination of trust and similarity values. Moreover, they do not use the transitivity of trust, but only directly trusted users. This method is not a trust-based recommendation method in the sense in which we use this term in this thesis.

Authors of [112] proposed a model for trust-based recommendation. Their approach is a simplified approach of the one proposed in TidalTrust[36] and MoleTrust [7]. To compute the trust value between users u and v, this model only considers one shortest path between u and v in the trust network and multiplies the trust values along the path to compute the indirect trust value.

Sometimes, people express distrust to some users besides expressing trust to some other users. Recently, propagation of distrust has been addressed in the literature [38]. The intuition is that the

distrust propagates only one level. In other words, if a trusted friend distrust another user, it can be inferred that we may distrust that user as well. However, if u distrusts v and v distrusts w, we can not say anything about the relation of u and w [38].

2.7 Top-N Recommendation

As discussed in the introduction, two general tasks can be distinguished in a recommender system: predicting a rating on a target item, and recommending top-N items. Most state-of-the-art methods for recommendation address the problem of predicting a single rating for a user. However, there has also been some work on top-N recommendation, which is reviewed in this section. [30] and [56] deal with Boolean ratings (indicating whether an item was purchased), while [57], [62] and [77] deal with numeric ratings (integer-valued scores). All of these methods adopt the collaborative filtering approach.

User-based collaborative filtering has been extended for top-N recommendation [77]. To recommend top-N items to a user u, the top K similar users to the u are computed first. Then list of items rated by similar users is aggregated and the top-N highly ranked items in this aggregated list are returned as the top-N recommended items. Similarly, item-based collaborative filtering can be extended to perform top-N recommendation [30]. The general idea is to find the top-K items similar to the items rated by a user and aggregate those items to compute the top-N recommended items. In the following, we review some works addressing the top-N item recommendation problem.

The method proposed in [30] is one of the first works addressing the problem of top-N recommendation. They extend the item-based collaborative filtering method [101] and present two alternative item-to-item similarity measures. The first one models the items as vectors in the user space and uses the cosine function to measure the similarity, whereas the second one uses a technique based on the conditional probability between two items. The second measure can differentiate between users with varying amounts of historical information as well as between frequently and infrequently purchased items. To perform the actual recommendation, the top similar items are computed for each item purchased by the target user, items are ranked according to the frequency of appearing in the set of similar items for different items purchased, and the top-ranked N items are returned. Intuitively more similar items should have more impact on the recommended items, but this is not supported. In addition, this method cannot handle cold start users well, since these users have purchased only few items which do not provide enough information to compute the top-N recommended items.

Authors of [56] also employ item-based collaborative filtering for top-N recommendation and introduce various evaluation metrics for top-N recommendation. Since they are using item-based collaborative filtering, they have similar issues with cold start users and similar items as the previous method.

The model proposed in [57] is another work exploring the item-based collaborative filtering approach, taking user feedback into account. They compute an error matrix recording the difference between the actual ratings and the predicted ratings. For a source user u and a target item i for which u has not expressed a rating, the predicted rating is computed by summation of the average error on predictions for u and the average error on predictions for i. The error matrix is updated upon receiving a new actual rating. This model is also unable to handle users and items with few ratings.

Kwon et al. [62] build a top-N recommendation method on top of existing recommendation systems for single ratings. They observe that the prediction accuracy decreases with increasing rating variance. Therefore, various methods are presented that filter out items with high rating variance. Since this approach reduces the diversity of items among the recommendations over all users, another method is proposed which does not filter out, but only penalizes high variance items. Variance-based filtering and weighting can be applied to any top-N recommendation algorithm including memory-based or model-based collaborative filtering (which are considered in [62]).

The method proposed in [77] has exploited user-based collaborative filtering to perform top-N recommendation. They introduce the Belief Distribution Algorithm that computes the belief (distribution) of rating differences instead of point estimates of the rating as done by existing methods. They estimate the belief difference between each user's average rating and the estimated rating on the items. The predicted belief difference for the source user and a given item is computed and added to the source user's average rating to obtain the predicted rating. Finally, the items having one of the top-N predicted ratings are returned as the recommended items. This method is an extension of the collaborative filtering algorithm for top-N recommendation.

In this chapter, we reviewed some of the state-of-the-art related work on recommendation, in particular recommendation in social networks. In the following chapters, if necessary, we also discuss the related work specific to the topic of that chapter.

Before digging deep into recommendation models, in the next chapter we model the temporal dynamics of user behavior in social rating network to capture these effects and to better understand the underlying mechanisms of user behavior in a social network.

Chapter 3

Modeling the Temporal Dynamics of Social Rating Networks using Bidirectional Effects of Social Relations and Rating Patterns

Before drilling down into discussing different models for recommendation, in this chapter we investigate the temporal dynamics of a social network and how different effects influence the behavior of users in a social rating network. These effects are fundamental to better understanding of user behaviors in SRNs. In this chapter, we analyze these effects and propose a generative model [53] to capture these effect in social rating networks.

3.1 Introduction

Users in a social rating network can perform two types of actions: creating a link to another user (*social action*) and creating a rating for an item (*rating action*). Note that the "rating action" is a general term and includes the real valued item ratings in Epinions and Flixster and binary rating values such as joining a community in Facebook and LiveJournal or adding a photo to your favorite list in Flickr. In other words, different types of user behaviors can be formulated as a rating action. In this chapter, we analyze and model the behavior of users while performing different actions. We also take into account the timestamp to analyze the temporal dynamics of user behavior. Our

model assumes real valued ratings. Binary ratings such as joining a community can be considered as special cases of real valued ratings.

Many effects influence the behavior of users in the evolution of a social rating network. A fundamental property of social networks is that people tend to behave similarly to their friends. The process of *social influence* [33] in SRNs leads to people adopting the rating behavior of their friends. Also people may adopt the rating behavior from users having similar rating patterns. We call this *correlational influence*. While this effect is the implicit foundation of the successful collaborative filtering recommenders [37], it has not yet been explicitly considered in social network models. The social behavior of users is also investigated. As discussed in [26], people tend to form relationships with others who are already similar to them, so-called *selection* [78]. However, we argue that social selection is not the only mechanism influencing the social relation creation process in SRNs. Implied by the well-known trust transitivity [44], people tend to create social relations to friends of their friends. We call this effect *transitivity*.

There are also some other effects that influence the behavior of users. For example, users may create social relations to other users who live in the same location, or they may befriend their co-workers. Users may rate items they face in their daily life independent of their friends. These effects are called environmental or external effects [4]. Since we do not have access to these kinds of information, we make a simplifying assumption about these kinds of behaviors: when creating a social relation, we assume that there is a chance that users create a link to a random existing user (representing unknown effects) or to a new user. Similarly, there is a chance that users rate a random existing item or a new item.

In our experiments, we use one data set with real valued ratings (Epinions) and another data set with binary ratings (Flickr). According to our observations from these real life data sets, different effects influencing the behavior of users are dynamic. In other words, the strength of each each effect is not constant throughout the evolution of an SRN. For example, when very few users have created social relations and the average number of social relations per user is very low, the strength of social influence and transitivity is low, but these effects become more important during the growth of a social rating network in which users rate more items, create more social relations and are exposed to more people and items. Effects influencing the behavior of users are not the only dynamic components governing the evolution of an SRN. At the beginning of the evolution when there are only few nodes in the system, the rate of new users joining the system is high. Also, the rate of creating a social relation to a new user decreases with the increase in the number of existing users in the system.

In this chapter, based on our observations and analysis, we propose a probabilistic generative model [53] to model the behavior of users and capture the temporal dynamics of different effects influencing the behavior of users. To capture the dynamism of effects, we model each of them as a function of some evolving features of the SRN such as the number of users, the number of neighbors for a users, etc.

This generative model can serve various purposes. First, there are very few SRN data sets publicly available. This is mainly due to privacy concerns in social networks, especially when some behaviors of the user should also get published. The lack of publicly available data sets is seriously restricting research on social network-based recommendation. A generative model for social rating networks can be used to generate synthetic data sets with properties similar to those of real-life data sets, which can be used for research purposes. Moreover, the study of the evolution of social rating networks and the growth patterns for social networks and rating profiles is an important application of a generative model, providing insights into models developed in the social sciences. Also such a model can be used for predicting future links, ratings or community structures.

The main contributions of this chapter are as follows:

- We analyze the temporal dynamics of SRNs and provide insights on bidirectional effects of social relations and rating patterns. We analyze and model the dynamism of effects influencing the behavior of users in an SRN. The dynamic nature of these effects has been neglected in the literature.

- We present a generative model for the evolution of SRNs, modeling different effects influencing the behavior of users. To the best of our knowledge, our model is the first one to comprehensively capture all the four effects between social relations and ratings: social-on-rating (social influence), social-on-social (transitivity), ratings-on-social (selection), ratings-on-rating (correlational influence).

- We perform an experimental study on two real life data sets (Epinions and Flickr), showing that the proposed model can indeed produce realistic SRNs. To compare different generative models we introduce several formal evaluation metrics.

- Our experimental study provides interesting insights into the factors that drive SRNs. For example, we find that transitivity is much more important than selection in the creation of social relations for both Epinions and Flickr.

The rest of the chapter is organized as follows. We present our observations and analysis of the temporal dynamics of SRNs in Section 3.2. Based on our observations and analysis, we present a model for the behavior of users and the evolution of SRNs in Section 3.3. Related work is discussed in Section 3.4. Evaluation metrics and experimental results on real life data sets are presented in Section 3.5.

3.2 Analyzing the Temporal Dynamics of SRNs

To observe the behavior of users in an SRN and to analyze the evolution of SRNs, we performed experimental studies on two real life data sets: Epinions and Flickr. Epinions.com is an online reviewing website. In Epinions, two types of actions are provided: users rating items with some rating values, and users creating social relations to other users. In Flickr, we consider two types of actions: users creating social relations to other users, and users adding photos to their favorite list. We consider adding a photo to favorites as a binary rating action (with rating value 1).

We represent the data set as a sequence of actions $\mathcal{A}$. Each action can be either a rating action creating a rating or a social action creating a social relation. Every action a in $\mathcal{A}$ is also associated with a timestamp. We order the actions with respect to their timestamps and observe the behavior of users in chronological order.

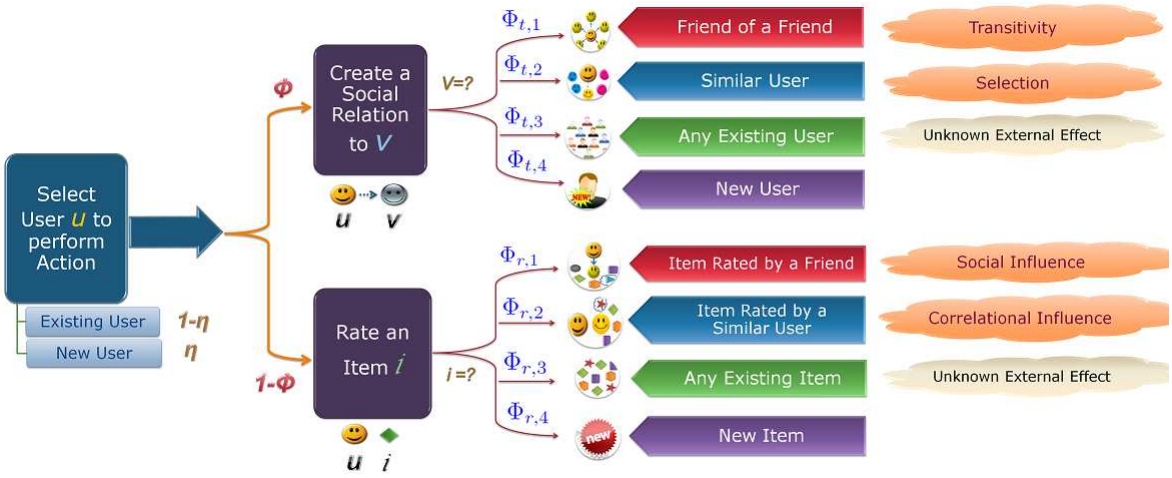


Figure 3.1: Illustration of different effects influencing the behavior of users in an action.

For every action a, we observe and analyze the behavior of users as follows (see Figure 3.1):

- Who is the user u performing the action? One of the existing users or a new user joining the SRN and performing his first action.
- What type of action is being performed? A rating action or a social action?
- If it is a social action, who is the target user v to whom the social relation is being created?
  - If transitivity is playing role in this action, then v should be one of the friends of friends of u.
  - Selection leads to select v among top similar users to u.
  - If unknown external effects are influencing this action, then v could be any existing user.
  - Finally, there is the possibility of v being a new user just joining the SRN.
- If it is a rating action, what item i is being rated and by what value?
  - If social influence is affecting the rating behavior of u in this action, then i should be one the items rated by friends of u. The value of the rating should also be affected by ratings expressed by direct neighbors of u on i.
  - If correlational influence is affecting the behavior of u, then i should be one of the items rated by top similar users to u. The rating of u on i should also be influenced by the ratings of similar users on i.
  - If some unknown external effect is influencing the behavior of u in this action, then i could be any existing item.
  - It could be also the case that i is a new item which has not been rated in the SRN before this action.

Note that the rating actions being affected by social influence, correlational influence and unknown external effects may overlap. Also the social actions influenced by transitivity, selection and unknown external effects may overlap. We compute the similarity among users using Pearson correlation [96] between their ratings. Similar to the idea used in [26] to construct a similarity network, the number of top similar users considered for each user u in rating actions is the same as the number of direct neighbors of u in the social network, N_u[1]. For social actions of user u, the number of similar users considered is the same as the number of users who are friends of friends of u.

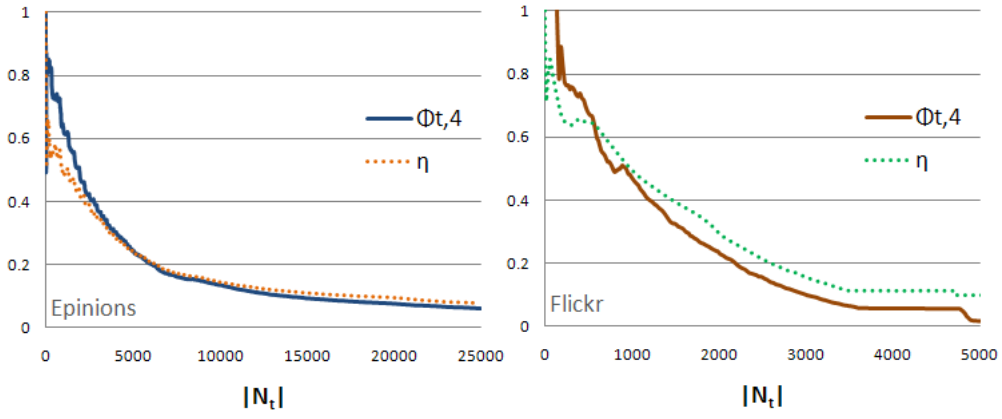
[1]Since there could be users with no direct neighbor but with some similar users, in our experiments, the number of similar users is set to be at least K=10.

First we analyze the role of new users and items in the evolution of SRNs. As shown in Figure 3.1, the probability that an action is performed by a new user is η. However this probability is not a fixed number. Figure 3.2(a) presents our observation on the percentage of actions being performed by new users versus the total number of users existing in Epinions and Flickr. As shown in this Figure, for both data sets η has a power law-like distribution over $|N_t|$, where N_t is the current set of existing users in the SRN at time t. At the very beginning of SRN's evolution, when there exists only a few users, the probability of an action being performed by a new user is very high, and this probability decreases very fast with the increase in the number of existing users. As depicted in Figure 3.2(a), a similar behavior exists for the percentage of social actions in which a social relation is created to a new user, denoted by $\Phi_{t,4}$. If the action is a rating action, the percentage of rating actions in which a new item is rated (denoted by $\Phi_{r,4}$) also exhibits a power law-like distribution over the number of existing items for both Epinions and Flickr, as shown in Figure 3.2(b).

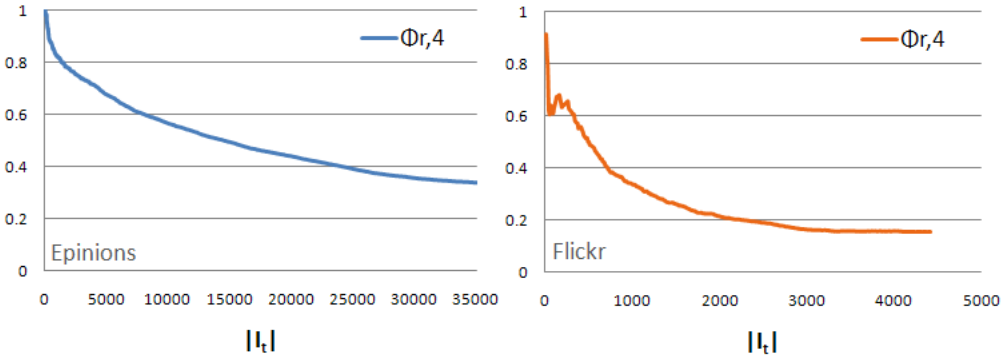
Our observations show that the percentage of social actions (ϕ) over the total number of actions (denoted by $|\mathcal{A}_t|$) remains fairly constant throughout the evolution of SRN for both Epinions and Flickr.

When a social relation is created by a user to some existing user, it could be either influenced by transitivity or selection, or by some unknown external effect. If unknown effects are playing a role, we assume that the social relation is being created to an arbitrary user. Our observations show that the percentage of social actions that are influenced by transitivity or selection increases almost linearly with growth of the number of social actions. We believe that in the course of time more social relations and more ratings are being expressed and hence users get more similar and also get to know friends of their friends. These events may lead to a stronger influence of similar users and direct neighbors on their behavior for creating social relations. Figure 3.3(a) depicts the percentage of social relations created to existing users that are not affected by transitivity or selection for different values of $|N_u^*| + |S_u^*|$ for both data sets, where N_u^* is the set of friends of friends of u and S_u^* is the set of top similar users for u. As shown in this figure, the more users become available in N_u^* and S_u^*, the less likely it becomes that a social action can not be explained by the effect of transitivity or selection. Again, the diagram resembles a power law distribution.

Similar behavior is observed for rating actions. The percentage of rating actions that are affected by social or correlational influence increases almost linearly with the growth of the number of rating actions. Figure 3.3(b) depicts the percentage of rating actions rating an existing item that are not affected by social or correlational influence for different values of $|IN_u| + |IS_u|$ for both data sets, where IN_u denotes the set of items rated by friends of u and IS_u denotes the set of items rated by



(a)

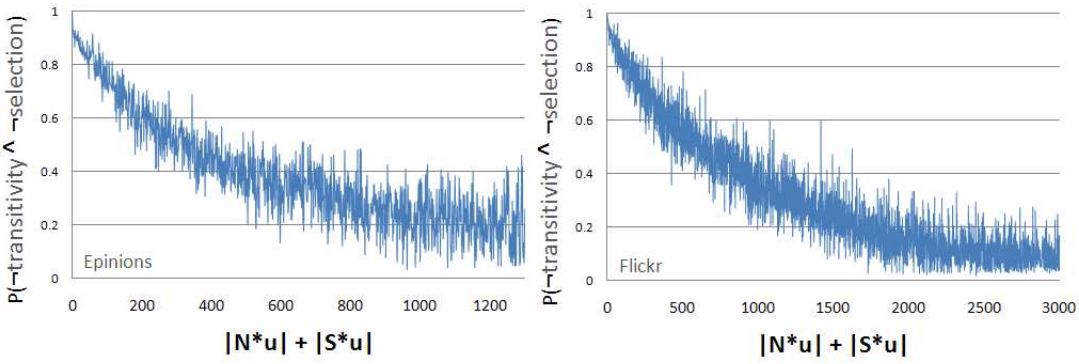


(b)

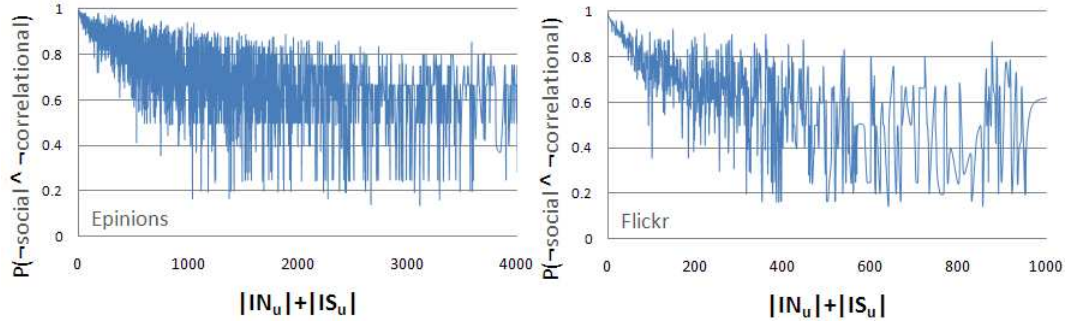
Figure 3.2: a) The distribution of η and $\Phi_{t,4}$ versus the number of existing users in Epinions and Flickr. η is the percentage of actions that are performed by a new user. $\Phi_{t,4}$ is the percentage of social actions that are created to a new user. b) Evolution of the percentage of ratings in which a new item is rated ($\Phi_{r,4}$).

top similar users for u. Figures 3.3(a) and 3.3(b) show that the percentage of actions that can be explained only by unknown effects decreases in a power law-like distribution in both Epinions and Flickr.

Furthermore we analyze the percentage of social actions in which a social relation is created to a friend of a friend versus the social actions in which the social relation is created to a top similar user. Surprisingly, this percentage remains fairly constant throughout the evolution of the Epinions data set. Figure 3.4(a) shows the percentage of social actions influenced by transitivity or selection that are solely influenced by transitivity (denoted by $\phi_{t,1}$) versus the total number of social actions for Epinions and Flickr. Note that social actions influenced by transitivity may overlap the social



(a)



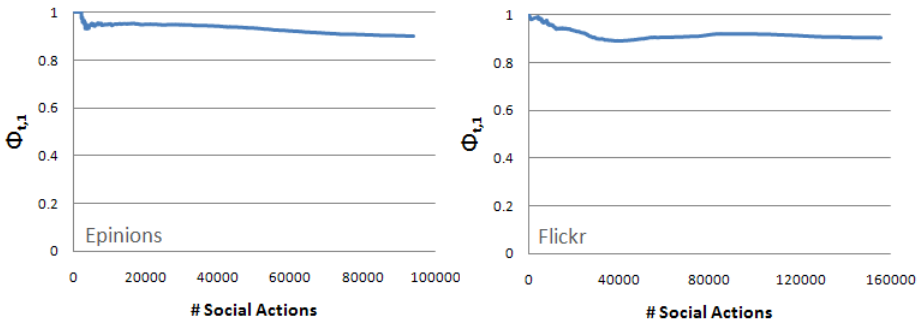
(b)

Figure 3.3: a) *P(¬(transitivity ∨ selection))*: Percentage of social actions that are not affected by transitivity or selection for different values of $|N_u^*| + |S_u^*|$ in Epinions and Flickr. b) *P(¬(social ∨ correlational))*: Percentage of rating actions that are not affected by social or correlational influence for different values of $|IN_u| + |IS_u|$ in Epinions and Flickr.

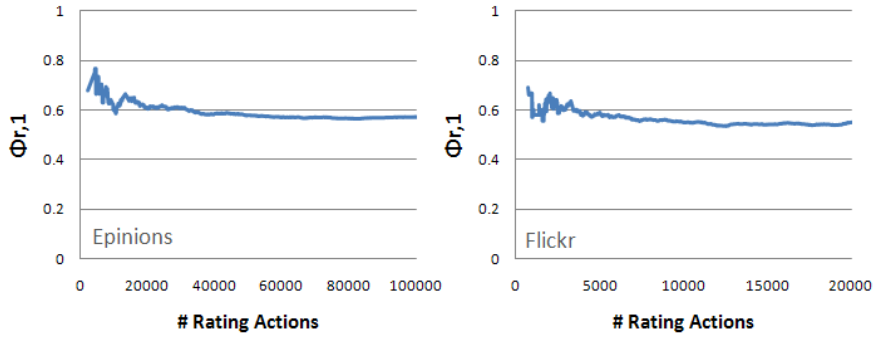
actions influenced by selection. As shown in this figure, transitivity plays a more important role in creating social relations than selection in both data sets.

The next observation is the percentage of ratings being affected by social influence versus the actions affected by correlational influence. Figure 3.4(b) depicts the percentage of rating actions affected by social or correlational influence that are solely influenced by social influence (denoted by $\phi_{r,1}$) versus total number of rating actions for Epinions and Flickr. Again, similar to the case for social actions, this percentage is fairly constant throughout the evolution of the SRN in both Epinions and Flickr.

Our final observation is the distribution of rating values throughout the evolution of an SRN.



(a) $\phi_{t,1}$



(b) $\phi_{r,1}$

Figure 3.4: a) Relative strength of transitivity and selection for Epinions and Flickr. Each diagram shows the percentage of social actions influenced by transitivity or selection that are solely influenced by transitivity (denoted by $\phi_{t,1}$) versus total number of social actions in a data set. b) Relative strength of social influence and correlational influence for Epinions and Flickr. The diagrams show the percentage of rating actions affected by social or correlational influence that are solely influenced by social influence (denoted by $\phi_{r,1}$) versus total number of rating actions.

Figure 3.5 depicts the evolution of the percentage of each rating value in Epinions[2]. Interestingly, we observe that the relative frequency of the rating values are fairly constant during the evolution of the SRN. The percentage of higher rating values are larger than those of low rating values, demonstrating that users in Epinions tend to be very generous in their ratings.

[2]Note that since the ratings in Flickr are binary only, this observation does not apply to the Flickr data set.

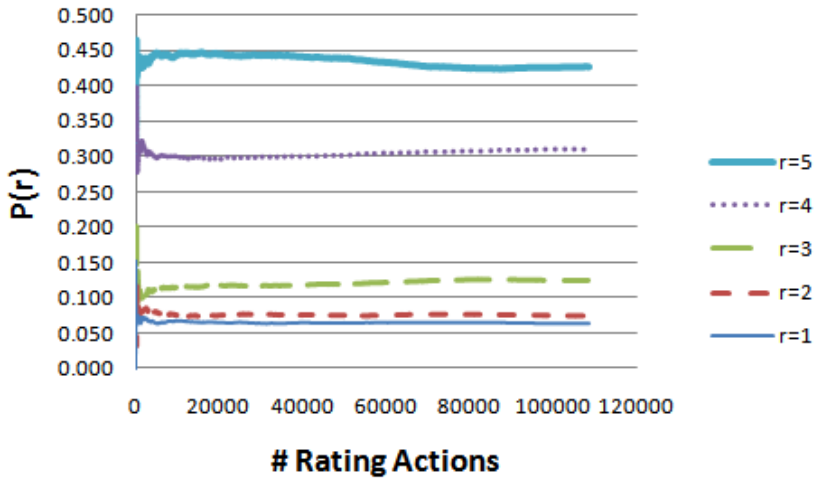


Figure 3.5: Distribution of rating values throughout the evolution of social network in Epinions. Possible rating values are integers in the range [1,5].

3.3 Modeling the Temporal Dynamics of SRNs

In this section we exploit our observations and analysis to model the temporal dynamics of social rating networks. To do so, we represent different effects influencing the behavior of users in a probabilistic generative model.

As discussed before, a social rating network consists of a sequence of dated actions. In the following, we explain the details of the proposed model (see Figure 3.1) for the generative process of performing an action.

In our model, first the user to perform the action is selected. With probability η this user is a new user. With probability $1-\eta$ one of the existing users performs the action. η is not a fixed value. Inspired by our observation in Figure 3.2(a), we assume a power law function for η over the number of existing users as follows:

$$\eta = power(|N_t|) = a_1 \times (|N_t| - b_1)^{c_1} \tag{3.1}$$

where N_t is the current number of existing users. Now we introduce $P(u|\mathcal{S}_t)$, the probability of user u performing the action. $\mathcal{S}_t$ denotes the current state of the SRN which summarizes the effects of all actions performed up to now. If the model decides that the next action is to be performed by a new user, then "selecting" a user u is deterministic: the model simply generates a new user u to

perform the action. If it is decided that the next action is to be performed by an existing user, then the probability of a particular user u performing the action is proportional to the number of actions already performed by u taking into account a smoothing parameter ϵ_0. This is similar to the "rich get richer" idea which is also the basis of preferential attachment [86][87].

Next, the type of the action being performed is decided: social action or rating action. Based on the observations the probability of creating a social relation is considered to be a constant parameter ϕ.

If the action is a social action, then there are four alternatives on how to create the social relation: creating a social relation to a new user, to a friend of a friend (transitivity), to a similar user (selection), or to a random existing user (unknown effect). With probability $\Phi_{t,4}$ the action will create a social relation to a new user. According to our observations shown in Figure 6.4(a), $\Phi_{t,4}$ decreases with the increase of the number of existing users by a power function. So, we model this probability as follows:

$$\Phi_{t,4} = power(|N_t|) = a_2 \times (|N_t| - b_2)^{c_2} \tag{3.2}$$

With probability $\Phi_{t,3} = (1 - \Phi_{t,4})\phi_{t,3}$, the action creates a social relation to some random existing user influenced by unknown external effects. These are the social actions that can not be explained by transitivity or selection. To model $\phi_{t,3}$ we exploit the observation shown in Figure 3.3(a) and assign social actions that can not be explained by transitivity or selection to unknown external effects as follows:

$$\phi_{t,3} = power(|N_u^*| + |S_u^*|) = a_3 \times (|N_u^*| + |S_u^*| - b_3)^{c_3} \tag{3.3}$$

where N_u^* is the set of friends of friends of u and S_u^* is the set of top similar users to u. Note that as mentioned before, S_u^* is defined in a way such that $|S_u^*|$ is the same as $|N_u^*|$. If we decide to create a social relation to an arbitrary existing user, we do so by assigning higher probabilities to users to whom more social relations have been created already. This is the same as in preferential attachment models [86].

$$P_e(v|u, \mathcal{S}_t) = \frac{d_{in}(v) + \epsilon_3}{\sum_w (d_{in}(w) + \epsilon_3)} \tag{3.4}$$

In the above equation, $d_{in}(v)$ is the in-degree of user v in the social network and ϵ_3 is a smoothing parameter. $P_e(v|u, \mathcal{S}_t)$ is the probability that u creates the social relation to v who is among existing users.

With probability $(1 - \Phi_{t,4})(1 - \phi_{t,3})$, the action creates a social relation to either a friend of a friend or a top similar user. Based on our observations in Figure 3.4(a), $\phi_{t,1}$ is constant throughout the evolution of the social rating network and can be learned as model parameters. Hence, in our model a social action creates a social relation to a friend of a friend with probability $\Phi_{t,1} = (1 - \phi_{t,4})(1 - \phi_{t,3})\phi_{t,1}$ and to a top similar user with probability $\Phi_{t,2} = (1 - \phi_{t,4})(1 - \phi_{t,3})(1 - \phi_{t,1})$.

If the social action is creating a social relation to a friend of a friend, then the probability of a friend of friend v being selected is proportional to the frequency of appearance of v in the set of friends of friends of u:

$$P_t(v|u, \mathcal{S}_t) = \frac{f_t(v|u) + \epsilon_1}{\sum_{w \in N_u^*} (f_t(w|u) + \epsilon_1)} \tag{3.5}$$

where $P_t(v|u, \mathcal{S}_t)$ is the probability of creating a social relation from u to a friend of friend v when deciding to create the social relation to a friend of friend. $f_t(v|u)$ is the frequency of appearances of v in the set of friends of friends of u and ϵ_1 is a smoothing parameter to be learned.

Similarly, the probability of creating a social relation to a top similar user v is modeled as follows:

$$P_s(v|u, \mathcal{S}_t) = \frac{sim_{u,v} + \epsilon_2}{\sum_{w \in S_u} (sim_{u,w} + \epsilon_2)} \tag{3.6}$$

where ϵ_2 is the smoothing parameter to be learned for our model. $P_s(v|u, \mathcal{S}_t)$ is the probability that u creates a social relation to v who is among his top similar users.

Next, we model rating actions. There are four alternative ways to rate an item: rating a new item, rating an item rated by friends (social influence), rating an item rated by similar users (correlational influence), and rating a random existing item (unknown effect). With probability $\Phi_{r,4}$ a new item is rated by action performer u. As we observed in Figure 3.2(b), this probability exhibits power law behavior. So, we model it as follows:

$$\Phi_{r,4} = power(|I_t|) = a_4 \times (|I_t| - b_4)^{c_4} \tag{3.7}$$

where I_t is the current set of existing items. To select the rating value for this rating action, we apply the observation in Figure 3.5 and model the probability of each rating value r by μ_r, where μ_r is a model parameter indicating the prior probability of rating value r.

With probability $\Phi_{r,3} = (1 - \Phi_{r,4})\phi_{r,3}$, user u rates an arbitrary existing item influenced by unknown external effects. These are the rating actions that can not be explained by social or correlational influence. Similar to our model for social relations, we exploit the observation shown

in Figure 3.3(b) to model $\phi_{r,3}$ and assign all rating actions that can not be explained by social or correlational influence to unknown external effects as follows:

$$\phi_{r,3} = power(|IN_u| + |IS_u|) = a_5 \times (|IN_u| + |IS_u| - b_5)^{c_5} \tag{3.8}$$

where IN_u is the set of items rated by direct neighbors of u and IS_u is the set of items rated by top similar users of u. In this case, the probability of selecting an existing item i, $P_e(i|u, \mathcal{S}_t)$, is modeled to be proportional to the number of ratings already expressed for i:

$$P_e(i|u, \mathcal{S}_t) = \frac{|I_i| + \epsilon_8}{\sum_j (|I_j| + \epsilon_8)} \tag{3.9}$$

where I_i is the set of users who have rated i. The next step is to decide the rating value for this action. For each rating value r, we combine the prior probability μ_r with the percentage of existing ratings for item i with value r:

$$P_e(r|i, \mathcal{S}_t) \propto \frac{f_i(r) + \epsilon_9}{\sum_{r'}(f_i(r') + \epsilon_9)} + \mu_r \tag{3.10}$$

where $f_i(r)$ is the frequency of rating value r for item i and ϵ_9 is the smoothing parameter. $P_e(r|i, \mathcal{S}_t)$ is the probability of assigning rating value r to i if i is a random existing user.

With probability $\Phi_{r,1} = (1 - \Phi_{r,4})(1 - \phi_{r,3})\phi_{r,1}$, the rating action selects the item from one of the items rated by direct neighbors of u. Based on our observations in Figure 3.4(b), $\phi_{r,1}$ remains fairly constant throughout the evolution of the SRN, and can be learned as a model parameter. If an item already rated by direct neighbors is being rated, then the probability of item i being rated is as follows:

$$P_t(i|u, \mathcal{S}_t) = \frac{f_t(i|u) + \epsilon_4}{\sum_{j \in IN_u} (f_t(j|u) + \epsilon_4)} \tag{3.11}$$

where $f_t(i|u)$ is the frequency of item i appearing in the set of items rated by direct neighbors of u. To compute the probability of rating the item by value r (denoted by $P_t(r|u, i, \mathcal{S}_t)$), we use the prior knowledge of general probability of rating i by value r combined with the distribution of rating values on i among direct neighbors u as follows:

$$P_t(r|u, i, \mathcal{S}_t) \propto \frac{f_t(r|i, u) + \epsilon_5}{\sum_{r'} (f_t(r'|i, u) + \epsilon_5)} + P_e(r|i, \mathcal{S}_t) \tag{3.12}$$

In the above equation, $P_e(r|i, \mathcal{S}_t)$ is the prior probability of rating item i by value r as defined in Equation (3.10). $f_t(r|i, u)$ is the frequency of rating value r for item i among ratings expressed

by friends of u for item i. It should be noted that since the ratings are binary in Flickr, models for rating values (Equations (3.10) and (3.12)) apply to Epinions only. In Flickr, all rating values are 1.

With probability $\Phi_{r,1} = (1 - \Phi_{r,4})(1 - \phi_{r,3})(1 - \phi_{r,1})$, the rating action selects the item from one of the items rated by top similar users to u. The process for rating an item in this case is similar to the process of rating an item rated by direct neighbors of u in the social network. $P_s(i|u, \mathcal{S}_t)$ and $P_s(r|u, i, \mathcal{S}_t)$ are defined in a way similar to the definition of $P_t(i|u, \mathcal{S}_t)$ and $P_t(r|u, i, \mathcal{S}_t)$.

3.3.1 Learning the model parameters

In this section, we present our approach to learn the parameters of our model. As discussed above, parameters of our model include 10 smoothing parameters ($\epsilon_0 ... \epsilon_9$), 15 parameters for power law functions ($a_1, b_1, c_1 ... a_5, b_5, c_5$), ϕ, and $\phi_{t,1}$,$\phi_{r,1}$. We denote the set of all parameters by Θ.

We compute the likelihood of the ratings and social relations observed in the data set under our model, and resort to maximum likelihood (ML) estimation to learn the values of the model parameters. We consider the network generation from the very beginning, and maximize the likelihood of the observed sequence of social and rating actions.

From the start time to the end time T, we observe K actions which transform the SRN from the state $\mathcal{S}_0$ to the state $\mathcal{S}_T$. $\mathcal{S}_t$ denotes the state of the SRN at time t which summarizes the effects of all actions performed up to this time. Let $\mathcal{A}$ denote the sequence of all actions performed between times 0 and T, in which the actions are sorted according to their timestamps. We denote each action in $\mathcal{A}$ by a_k and its timestamp by t_k, where $k \in [1..K]$. Based on the chain rule, the likelihood of the model is the product of the probabilities of each individual action in $\mathcal{A}$ given the previous actions:

$$P(\mathcal{A}|S_0, \Theta) = \prod_{k=1}^{K} P(a_k|\mathcal{S}_{t_k}, \Theta) \tag{3.13}$$

For the two types of actions (social actions, rating actions), we compute the likelihood of the action in a different way. The probability of selecting a user u, and performing a social action $T_{u,v}$ to create a social relation between user u and user v is:

$$\begin{aligned} P(T_{u,v}|\mathcal{S}_t, \Theta) = P(u|\mathcal{S}_t) \times \phi \times \Big[& \Phi_{t,4} g_n(v|\mathcal{S}_t) + \\ & (1 - \Phi_{t,4})\phi_{t,3} P_e(v|u, \mathcal{S}_t) + (1 - \Phi_{t,4})(1 - \phi_{t,3})\phi_{t,1} P_t(v|u, \mathcal{S}_t) \\ & + (1 - \Phi_{t,4})(1 - \phi_{t,3})\phi_{t,2} P_s(v|u, \mathcal{S}_t) \Big] \end{aligned} \tag{3.14}$$

where $g_n(v|\mathcal{S}_t)$ is an indicator of whether v is a new user. Likewise, the likelihood of an action $R_{u,i,r}$ in which the user u gives the rating r for item i is:

$$\begin{aligned} P(R_{u,i,r}|\mathcal{S}_t,\Theta) = P(u|\mathcal{S}_t) \times (1-\phi) \times \\ \Big[\Phi_{r,4} g_n(i|\mathcal{S}_t)\mu_r + (1-\Phi_{r,4})\phi_{r,3} P_e(i|u,\mathcal{S}_t) P_e(r|i,u,\mathcal{S}_t) \\ + (1-\Phi_{r,4})(1-\phi_{r,3})\phi_{r,1} P_t(i|u,\mathcal{S}_t) P_t(r|i,u,\mathcal{S}_t) \\ + (1-\Phi_{r,4})(1-\phi_{r,3})\phi_{r,2} P_s(i|u,\mathcal{S}_t) P_s(r|i,u,\mathcal{S}_t)\Big] \end{aligned} \quad (3.15)$$

where $g_n(i|\mathcal{S}_t)$ is an indicator of whether i is a new item. We use expectation maximization (EM) to estimate the maximum likelihood model parameters.

3.4 Related Work

There has been a lot of research on modeling the evolution of social networks [45, 61, 66, 81]. However, these works only model the creation of social relations and do not consider attributes for nodes. In the following, we review some recent works which address the problem of modeling the evolution of social networks together with node attributes.

Authors of [26] introduce an evolution model for social networks with user activities. In the Wikipedia data set used in their experiments, activities are the editing of a certain Wikipedia article, and network edges represent interaction, i.e., participation in the discussions on another users's profile. The generative model of [26] considers only two factors: social influence and selection.

The co-evolution of social and affiliation networks has been explored by the authors of [118]. In this scenario, nodes are associated with group labels. The proposed model creates edges and group labels. For each user, groups to join are selected from those of his friends (social influence), and new friends are selected from his friends of friends (transitivity). However, the effects of selection and correlational influence are not modeled.

Like our work, [76] investigates social networks with user ratings. Similar to our work the bidirectional effects between trust and ratings are explored, but only selection and correlational influence are considered. The effect of social relations on the behavior of users is not discussed in this paper.

Evaluating generative models and the fit of synthetic networks with actual networks is an important issue in designing models for the evolution of social networks. In the following we review two

recent works which investigate evaluation metrics for generative models.

Reference [4] addresses the problem of distinguishing social influence (causation) from correlation. A shuffle test is proposed based on the intuition that if social influence is not a likely source of correlation in a system, timing of actions should not matter, and therefore reshuffling the time stamps of the actions should not significantly change the amount of correlation. They also define a metric for measuring social correlation. If the social correlation measure does not change after shuffling, then social influence is ruled out as a cause of social correlation.

The authors of [46] present a systematic examination of a real network data set using maximum likelihood estimation for exponential random graph models as well as new procedures to evaluate how well the models fit the observed networks. These procedures compare structural statistics of the observed network with the corresponding statistics on networks simulated from the learned model [46]. All metrics in this paper measure some structural property of social networks. Attributes of nodes are not considered.

The microscopic evolution of social networks is investigated in [66]. The authors argue that using the likelihood is a more objective measure for comparing alternative generative models than using some of the many potential relevant network statistics. Only simple social networks without attributes and only the influence of transitivity are considered. The timing of the creation of nodes and edges is also modeled, which is beyond the scope of our work.

3.5 Experiments

In this section, we present our experiments and discuss the results. To evaluate the performance of our model, we build a generative model for each data set, learning the parameters using maximum likelihood and expectation maximization.

We used the version of the Epinions data set[3] published by the authors of [98]. The data set contains 22K users, 30K items, 108K ratings, and 117K social relations between users[4]. The timestamps in the data set range from 2001/01/17 to 2002/02/01. Flickr.com is an online photo sharing website. We used the version of the Flickr data set[5] published by the authors of [23]. The Flickr

[3] http://alchemy.cs.washington.edu/data/epinions/

[4] Note that we had to prune the data set to remove all the links for which no timestamp was available. The timestamps are available for links after 2001/01/17 and therefore we had to remove all the ratings before that date as well as un-dated links.

[5] http://socialnetworks.mpi-sws.org/

data set used in our experiments contains 150K social relations and 30K rating actions expressed by 5.2K users on 5K items. The timestamops in the flickr data set are from 2206/11/02 to 2007/03/15.

In our experiments, we compare the model proposed in Section 3.4, called the *FullModel*, against the following models which capture only some of the effects:

- *CrossModel.* This model considers only the effect of ratings on social relations (selection), and the effect of social relations on ratings (social influence). This model is similar to [26], and is derived by removing the parameters $\Phi_{r,2}$ and $\Phi_{t,1}$ from the FullModel (or setting these parameters to zero in the FullModel). It should be noted that this model considers social influence and social selection to be constant.

- *SocialOnly.* It ignores the effects of similarities, i.e., it just models the effects of social influence and transitivity. This model is derived by removing parameters $\Phi_{r,2}$ and $\Phi_{t,2}$, from the FullModel. This setting simulates the model proposed by [118] in our framework. It should be noted that the model in [118] ignores the dynamic nature of effects.

- *SimilarityOnly.* It ignores the effects of social relations and just models selection and correlational influence. This model is derived by removing the parameters $\Phi_{r,1}$, $\Phi_{t,1}$ from the FullModel. Results of this model show whether considering only effects of similarity and correlation of rating patterns can capture the behavior of real data. The SimilarityOnly model is the same as the model proposed in [76]

- *Baseline.* In this setting, we ignore all four bidirectional effects of social relations and rating patterns. This model is derived by removing the parameters $\Phi_{r,1}$, $\Phi_{t,1}$, $\Phi_{r,2}$ and $\Phi_{t,2}$ from the FullModel and is designed to investigate the power of randomness in SRN generation. This model is close to the preferential attachment model [86].

3.5.1 Experimental Results

In this section, we present our experimental results with respect to different evaluation metrics. It should be noted that we generate 10 SRNs using each model and take the average of each evaluation metric over these 10 generated SRNs. Each generated SRN consists of the same number of actions as the real data.

After learning the parameters, some interesting insights can be gained on the behavior of users in Epinions and Flickr. The estimated value of ϕ is 0.52 for Epinions and 0.83 for Flickr. This indicates that in the Flickr data set, users tend to create more social relations compared to adding photos to

their favorites list. The estimated value of $\phi_{t,1}$ is 0.91 for Epinions and 0.9 for Flickr which means that the effect of transitivity is much more important than selection in both data sets. In other words, ignoring the effect of friends of friends on the creation of social relations (as done in [26]) is not realistic. For rating actions, we have $\phi_{r,1} = 0.59$ for Epinions and $\phi_{r,1} = 0.54$ for Flickr which shows that the strength of social influence is close to the strength of correlational influence with social influence being a little stronger in both data sets. In the following subsections we evaluate comparison partners against several evaluation metrics.

Comparing the likelihoods

As discussed in [66], using the likelihood of the real data is an objective measure for comparing alternative models. Table 3.1 lists the log-likelihood of Epinions and Flickr for all of the comparison partners. The FullModel achieves the clearly highest log-likelihood in both data sets, while the Baseline model, that ignores all four bidirectional effects, achieves the lowest likelihood.

Table 3.1: Comparison of the log-likelihood of the Epinions data set and the Flickr data set using alternative models.

Model	Log-Likelihood	
	Epinions	Flickr
FullModel	-2.914E6	-2.732E6
CrossModel	-2.994E6	-2.791E6
SocialOnly	-2.977E6	-2.764E6
SimilarOnly	-3.031E6	-2.807E6
Baseline	-3.122E6	-2.829E6

Evaluation of structural properties

As proposed in [46], we evaluate a generative model for social rating networks based on some structural properties of the generated graph. We compare the following measures of the generated SRN with those of the real SRN data sets in both Epinions and Flickr.

- *Link Degree Distribution*: The distribution of the out-degree of nodes.
- *User Rating Degree Distribution*: The rating degree of a user is the number of items rated by him.

- *Item Rating Degree distribution*: The rating degree of an item is the number of users who have rated the item.

Our experiments show that all the above distributions follow a power law. For each model, we compute the *scaling exponent*[6] of each distribution and compare it to the corresponding scaling exponent in the real data set. Table 3.2 compares the scaling exponent of different degree distributions for both Epinions and Flickr. Note that the scaling exponents for all models are very close. We believe this is mainly due to the fact that all models exploit the idea of "rich get richer". Thus all the comparison partners are generating SRNs close to the real data in terms of degree distributions.

Table 3.2: Comparison of the scaling exponent of each degree distribution of the SRN created by FullModel and the real data for both Epinions and Flickr.

Structural Measure	Epinions		Flickr	
	Model	Real	Model	Real
Link Degree	2.164	2.132	1.368	1.382
User Rating Degree	1.904	1.901	1.665	1.718
Item Rating Degree	2.68	2.76	1.998	1.969

Another metric we use is the effective diameter [86] of the social network. All comparison partners generate social networks with effective diameter of 6 in Epinions and 4.8 in Flickr, which are the same as the effective diameters of the real data set.

To conclude, evaluation on general structural properties showed that all models generate SRNs that are generally similar to the real data set.

Evaluation of social influence

We use an influence model proposed in [4] to compare the degree of influence in SRNs generated by the various models and the real data set.

At each timestamp t, each user u is exposed to some items. A user u is k-exposed to an item i at time t if that individual has not rated item i and has exactly $k \geq 1$ direct neighbors who have rated i by time t and the latest rating of i among the direct neighbors of u is in the interval $[t - \delta_0, t]$. Each item i is exposed to u at timestamp t by $a_{u,i,t}$ users. The probability $P(u, i, t)$ of user u rating item i in a time window of size δ after t is modeled as follows:

[6] http://en.wikipedia.org/wiki/Power_law

$$P(u,i,t) = \frac{e^{\alpha \ln (a_{u,i,t}+1)+\beta}}{1+e^{\alpha \ln (a_{u,i,t}+1)+\beta}} \tag{3.16}$$

Let $A_{u,t}$ denote the set of items user u has not yet rated but is exposed to at time t. Then we can compute the likelihood of the given data D as follows:

$$P(D) = \prod_{t} \prod_{u} \prod_{i \in A_{u,t}} P(u,i,t)^{Y_{u,i,t}} (1 - P(u,i,t))^{1-Y_{u,i,t}} \tag{3.17}$$

where $Y_{u,i,t}$ is a boolean random variable that takes the value 1 if user u rates item i in the time window [t,t+δ][7].

By maximizing the above likelihood we estimate the values of α and β. As discussed in [4], the coefficient α is considered as the measure of influence by the neighbors for selecting an item to rate:

$$\ln(\frac{P(u,i,t)}{1-P(u,i,t)}) = \alpha \ln (a_{u,i,t}) + \beta$$

We call this influence coefficient the *item adoption influence* since only the event of rating an item (but not the value of the rating) is taken into account. We can also model the probability $P(u,i,t)$ of user u rating item i within distance 0.5 of the average of ratings of his neighbors on i. We call this influence coefficient the *rating adoption influence*. The social influence measures for different models and the real data set are presented in Table 3.3. Note that since ratings in Flickr have binary values, rating adoption influence is not applicable to Flickr. As shown in this table, FullModel has the social influence measures that are closest among all comparison partners to those of the real data set in both Epinions and Flickr. Surprisingly, the SimilarityOnly model that uses only the effect of similar users comes fairly close to the real data set in terms of social influence coefficients in both data sets. All other models exhibit social influence coefficients that are much smaller than in the real data set. These results show that correlational influence play a very important role in rating behavior of users and neglecting it is not realistic.

Measuring the effect of social relations on the growth of similarity of rating patterns

When user u creates a social relation to user v, then according to social influence, u will consider ratings of v in his future actions. Thus, the similarity of ratings of users u and v should grow after

[7]For the sake of efficiency, we consider only 12 timestamps in our experiment to compute the exposures in Epinions (2001/02/17, 2001/03/17, ... 2002/01/17). δ and δ_0 are set to 30 days in our experiments with Epinions. In Flickr, we consider 8 timestamps (2006/02/15, 2006/03/01, ..., 2007/03/01). δ and δ_0 are set to 15 days

Table 3.3: Influence coefficients for item and rating adoption in the social rating network generated by different models and the real data set in Epinions and Flickr.

Model	Epinions		Flickr
	item adoption influence	rating adoption influence	item adoption influence
Real Data	2.1436	2.195	1.41
FullModel	2.186	2.252	1.32
CrossModel	1.170	1.314	1.14
SocialOnly	0.966	1.079	1.21
SimilarityOnly	1.951	2.008	1.56
Baseline	0.687	0.820	0.74

creation of the social relation. The more actions they do, the more similar they are supposed to get. The difference between the similarity of u and v at the end of the evolution of the SRN and at the moment of creating the social relation is called the *similarity growth* for users u and v. In our SRN, users may perform different numbers of actions after creating a social relation. We group pairs of users (u, v) based on the number of rating actions u performs after creating the social relation to v. Figure 3.6 shows the similarity growth versus the number of actions for the different models on the

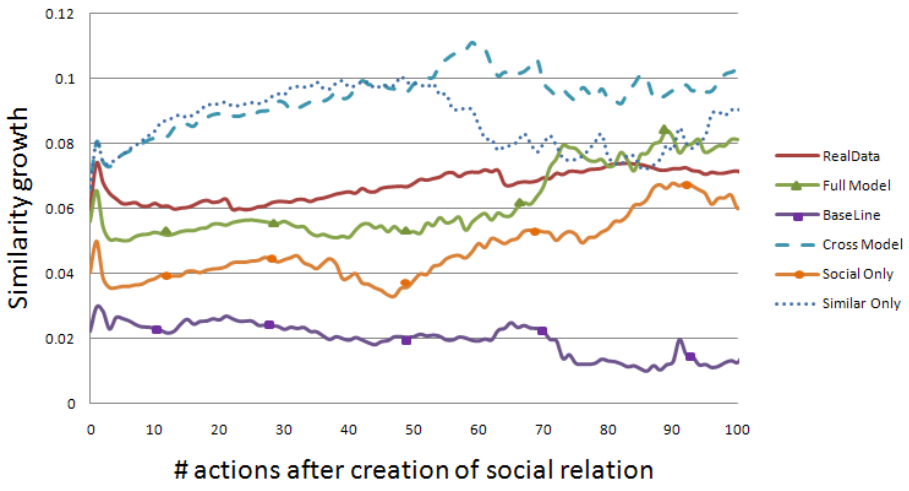


Figure 3.6: Comparison of Similarity Growth after creation of social relation for different models in Epinions.

Epinions data set. FullModel follows the pattern of similarity growth in the real data more closely than other models. There are some interesting and surprising results in Figure 3.6. The SimilarOnly and CrossModel tend to have higher growth of similarity compared to the real data, while SocialOnly tends to have a lower growth of similarity compared to both real data and FullModel. SimilarOnly and CrossModel use selection for link creation, but SocialOnly uses the transitivity for link creation. The similarity growth versus the number of actions for the different models in the Flickr data set is depicted in Figure 3.7. Again FullModel has the closest pattern of similarity growth to that of the real data. In a way, the results show that users who are already similar before creating a social relation, tend to get more similar compared to users who are not that similar when creating the social relation. FullModel is a tradeoff between these two cases and our experiments show that it behaves much more similar to the real data.

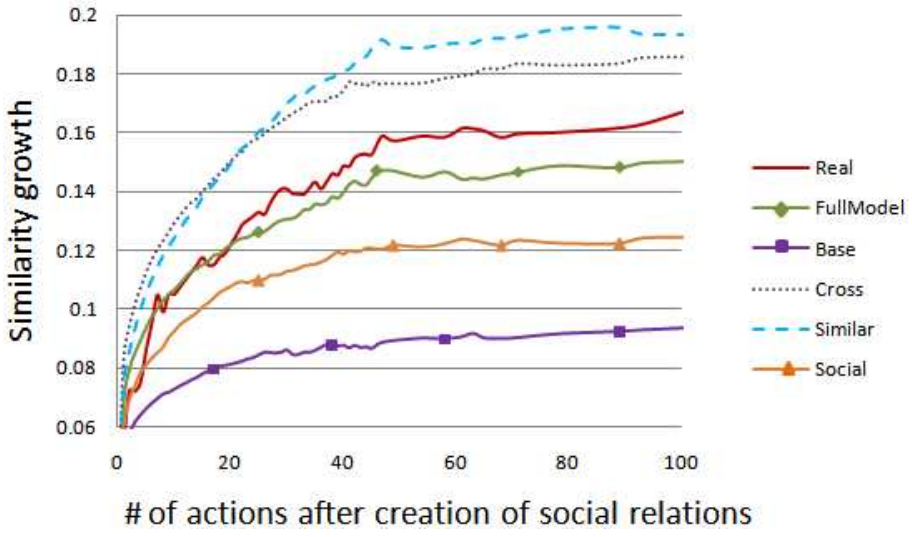


Figure 3.7: Comparison of Similarity Growth after creation of social relation for different models in Flickr.

Measuring the transitivity influence

Transitivity of social relations is an important property which affects the creation of social relations between users. The distribution of the shortest distance between users u and v, at the moment when u creates a social relation to v, can measure the transitivity effect. If the probability of short distances (i.e. 2) is high, then the transitivity has been an influential factor in trust creation. We compute the

average shortest distance of pairs of users at the moment of creating a social relation between these two users. Table 3.4 compares the average shortest distance of user pairs at the moment of creating

Table 3.4: Average shortest distance of user pair at the moment of creating social relations ($\overline{d_{min}}$) in different models and the real data set for Epinions and Flickr.

Model	$\overline{d_{min}}$	
	Epinions	Flickr
Real Data	3.93	2.73
FullModel	4.05	2.86
CrossModel	5.59	3.636
SocialOnly	3.24	2.68
SimilarOnly	5.59	3.64
Baseline	5.21	4.76

social relations ($\overline{d_{min}}$) in different models and the real data set for both Epinions and Flickr. Note that pairs of users who are not reachable at the time of the creation of a social relation are considered to have distance 7. As shown in this table, the average shortest distance for FullModel is the closest one to that of the real data set. SocialOnly achieves lower average distance since it only relies on neighbors and does not consider the selection effect. CrossModel, SimilarityOnly, and Baseline have a higher average shortest distance, likely because they are ignoring transitivity which according to our experiments is a very important factor in the creation of social relations.

Measuring the effect of selection

To evaluate the effect of selection, we compute the average similarity of users when creating a social relation. Table 3.5 presents the average similarity of user pairs at the moment of creating social relations for different models and for the real data in both Epinions and Flickr. In this table, we only consider the pairs of users for which we can compute the similarity. Table 3.5 also shows the percentage of user pairs for which we can compute the similarity in each comparison partner. Again, FullModel is clearly the closest to the real data set in both Flickr and Epinions. CrossModel and SimilarOnly have higher average similarity since they only rely on similar users for creating social relations and ignore the transitivity effect. The SocialOnly model, however, has an average similarity close to that of the real data set. This seems to be due to the fact that strength of selection in Epinions and Flickr is very low compared to transitivity. The FullModel is also closest to the real

data set in terms of the percentage of user pairs creating social relation for which we can compute the similarity.

Table 3.5: Average similarity of user pairs at the moment of creating social relations in different models and the real data set for both Epinions and Flickr.

Model	Epinions		Flickr	
	Average Similarity	% of non-zero similarity pairs	Average Similarity	% of non-zero similarity pairs
Real Data	0.381	3.07	0.618	6.2
FullModel	0.375	3.21	0.627	5.4
CrossModel	0.524	11.78	0.703	16.8
SocialOnly	0.365	2.01	0.716	4.3
SimilarOnly	0.558	12.03	0.681	15.7
Baseline	0.253	2.40	0.728	5.09

3.6 Conclusion

In this Chapter we analyzed and modeled the temporal behavior of users in an SRN using bidirectional effects of rating patterns and social relationships. Our model is based on our observations of the behavior of user while expressing ratings or creating social relations. While existing models for other types of social networks have captured some of the factors, our model is the first one to represent all four factors, i.e. social relations-on-ratings (social influence), social relations-on-social relations (transitivity), ratings-on-social relations (selection), and ratings-on-ratings (correlational influence).

Based on our observations, these effects are dynamic. In other words, the strength of each effect is not constant throughout the evolution of an SRN. We analyzed and modeled the temporal dynamism of these effects by defining them to be functions of the dynamic features of an evolving SRN. Given the sensitive nature of social network data, there are only very few public SRN data sets. This motivates the development of a model to capture the temporal dynamics of users' behaviors and to create such synthetic data sets for research purposes. To evaluate the accuracy of our model, we compared the synthetic SRN generated by our model with the real data and with SRN generated by other models that have been proposed in the literature.

Our experimental study on the Epinions data set and the Flickr data set demonstrated that the

proposed model produces social rating networks that agree with real world data on a comprehensive set of evaluation criteria much better than existing models. These criteria included the relevant degree distributions and measures of the four major influence factors that we identified for social rating networks. In addition, our experimental study provided interesting insights into the factors driving the evolution of SRNs. In particular, we found that transitivity plays a much bigger role in the creation of social relations than selection, although the most closely related generative model [26] does not even consider transitivity.

Inspired by our investigations in this chapter, in the next chapters we present models that exploit different effects in social rating networks to perform recommendation for a user.

Chapter 4

Random Walk Models for Combining Social Network-based and Similarity-based Recommendation

4.1 Introduction

In this chapter, we employ the effects influencing the behavior of users in an SRN and propose methods based on random walks on the SRN for rating prediction, link prediction and top-N recommendation. Random walk methods provide a principled approach to define the similarity between two users u and v in a network based only on the network topology. The similarity is defined by the steady state probability of a walk starting from u reaching v. Random walk models have been employed to address the link prediction problem, incorporating the transitivity effect into link prediction. In Chapter 3, we observed that transitivity plays a very important role in the formation of social relations in a social rating network [53].

We first present a novel random walk model for rating prediction called *TrustWalker* [47]. TrustWalker performs random walks on the social rating network to find predictions for the source user u on the target item i. When going far away from the source user u in the social network, the trust between these users and the source user will become fairly weak and their ratings will be noisy and unreliable. Therefore, the ratings expressed by users in the neighborhood close to the user u are preferred. However, in this case the probability of finding a rating expressed on the item will be very low and a prediction can not be computed. In order to consider enough ratings without suffering

from noisy data, TrustWalker combines social network-based and item-based recommendation and considers not only ratings of the target item, but also those of similar items. The probability of using the rating of a similar item instead of a rating for the target item increases with increasing length of the walk. Basically, TrustWalker performs random walks to compute the probability of using rating $r_{v,j}$ by user v on item j while trying to predict the rating $\hat{r}_{u,i}$ of user u on the target item i. The main intuition is to prefer users closer in the network and more similar to the source user u and also items more similar to the target item i. TrustWalker not only exploits the transitivity effect, but similar to other social network-based recommenders makes advantage of social influence [33] by using the ratings expressed by the users in the direct/indirect neighborhood. Moreover, TrustWalker takes correlational influence discussed in Chapter 3 into account by considering the user similarities in the transition probabilities in the random walks.

We employ the ideas introduced in TrustWalker to also address the link prediction problem: Given a user u_0, recommend top-N users whom u_0 is more likely to trust and create a social relation to. The proposed model, *LinkWalker* [51], performs random walks on the social network to find recommended users, rather than predicted ratings. LinkWalker is actually an extension of the random walk with restart method [89] for social rating networks, considering rating pattern similarities for the choice of transition and restart probabilities. Basically, by considering the rating pattern similarities for prediction of a link, LinkWalker takes into account the social selection effect [65][78] that have been discussed in Chapter 3. As stated before, since LinkWalker performs random walks to compute the recommended users, it also incorporates transitivity in the process of link prediction.

In this chapter, we also explore the social network-based approach to top-N item recommendation [48]. First, we extend TrustWalker for top-N recommendation. However, since we use leave-one-out method for evaluation and it is very unlikely that users in the direct or indirect neighborhood rate the exact withheld items, direct extension of TrustWalker does not achieve substantial gain over existing methods. Therefore, we propose a second method for top-N item recommendation which combines the social network-based and the collaborative filtering approach, performing a weighted merge of the results from both approaches. This method combines the strength of collaborative filtering, higher density of the neighborhood for normal users, with the strength of social network-based recommendation, good performance for cold start users. Again, similar to TrustWalker, the top-N recommender method proposed in this chapter also takes into account transitivity, social influence, and correlational influence.

Note that all the methods introduced in this chapter are memory-based approaches in which we explore the social rating network for recommendation of ratings, items, or users.

We perform experimental studies on two real life data sets: the public domain Epinions data set, and the Flixster data set we crawled and prepared for this research.

The rest of this chapter is organized as follows: Section 4.2 discusses some random walk models for recommendation and the foundations behind using random walk based approaches. We present the details of the TrustWalker model in Section 4.3. The LinkWalker model is introduced in Section 4.4. We present our proposed models for top-N recommendation in Section 4.5. In Section 4.6, we introduce the real life data sets used in our experiments. The experimental results and comparison with existing methods are discussed in Section 4.7. Finally we conclude the chapter in section 8.

4.2 Random Walk Methods for Recommendation

In this section, we motivate our random walk based approach to recommendation of items and links. Random walk methods have been proposed to address the link prediction problem [89, 116, 108, 32]. Random walks are used to compute the pairwise node similarities based on the network structure only [32]. More specifically, to predict links for a node u in a network, random walk with restart (*RWR*) works as follows: Consider a walker that starts from source node u. The walker iteratively walks to its neighborhood with the probability that is proportional to the edge weights. Also at each step, it has some probability c to return to the node u. The relevance score of node v with respect to node u is defined as the steady-state probability that the walker will finally stay at node v [89]. Nodes with highest relevance scores are returned as the top-N recommended users. The edge weights are computed by normalizing the adjacency matrix of the graph. The restart probability c is constant in RWR and does not distinguish different nodes in the network for restarting criteria. Note that in RWR, relevance score of node v with respect to node u is in fact a similarity measure between two nodes based on the network topology only [32].

The main intuition behind RWR is transitivity of social relations [94] [44]. In other words, neighbors of a neighbor are considered to be worth creating a social relation to. RWR considers only social networks as graphs with nodes and edges. In a social rating network, we have user ratings on top of the link structure of the social network. Sociologists believe that users tend to form social relations to users who have similar interests [65][78]. This phenomenon is called homophily or social selection [65]. Social selection has already been investigated as one of the effects influencing user while creating social relations [26][53]. To take user similarity (social selection) into account in RWR, we consider the rating pattern similarity of users as edge weights in the social network. Also we use the similarity of the source user u and other users to compute the restart probability in

a random walk. We discuss more details in the following sections.

The RWR method is traditionally employed for link prediction to estimate the likelihood of creation of a link. However, the main problem we address in this chapter is rating prediction. In the following section, we describe how we extend random walk methods for rating prediction. Basically, we perform random walks to compute the probability of using rating $r_{v,j}$ by user v on item j to predict the rating $\hat{r}_{u,i}$ of user u on target item i. The main intuition is that the closer the user v is to user u in the social network, the more likely that we use ratings expressed by v for prediction for u. Also, the more similar item j is to the target item i, the more likely we use a rating expressed on item j for prediction of a rating on item i. All the details and how we exploit social correlation, transitivity and correlational influence in random walk based methods are discussed in the next sections.

4.3 TrustWalker: Rating Prediction Model

The main challenge in trust-based rating prediction is to decide how far to go in exploring the network. There is a tradeoff between precision and coverage: the further you go, the more likely to find *raters*, but the less trust-worthy their ratings become. Our approach to find a good tradeoff is based on the following observation. Ratings expressed by strongly trusted friends on similar items are more reliable than ratings expressed by weakly trusted far neighbors on the exact target item. This motivates us to combine the trust-based and item-based approach.

We propose a random walk model, called TrustWalker, which considers not only ratings of the target item, but also those of similar items. The probability of using the rating of a similar item instead of a rating for the target item increases with increasing length of the walk. Basically, our model consists of two major components: the random walk on the trust network and the probabilistic item selection. The random walk performs the search in the trust network, and the item selection part considers ratings on similar items to avoid going too deep in the network. So our model improves the precision by preferring raters at a nearer distance and improves the coverage by considering similar items as well as the exact target item.

To predict a rating for a source user u_0 on target item i, we perform random walks on the trust network, each starting at u_0 to find a user having expressed rating for i or items similar to i. The details of the random walks will be discussed later in this section. Each random walk returns a rating. We perform several random walks, and the aggregation of all ratings returned by different random walks are considered as the predicted rating $\widehat{r}_{u_0,i}$.

In the following subsections, we will discuss the details of our random walk model. In our

notations, we use symbols $u, v, w, ...$ for users, $i, j, ...$ for items, and k for the step of a random walk. Table 4.1 lists all notations used in our model.

Table 4.1: Notations used in TrustWalker. RV stands for Random Variable. All the notations have index i denoting target item i.

Notation	Description
$\phi_{u,i,k}$	probability of stopping the random walk at node u in step k.
$X_{u,i,k}$	RV for being at node v in k steps starting from u
$X_{u,i}$	RV for being at node v at some steps starting from node u
S_u	RV for selecting a user v out of members of set N_u.
$Y_{u,i}$	RV for selecting item j amongst items rated by u
$XY_{u,i}$	RV for stopping at node v and selecting item j rated by v, while starting from u.
$r_{u,i}$	The rating expressed by u on item i
$\hat{r}_{u,i}$	The predicted rating of user u on i
$t_{u,v}$	The trust value among users u and v

4.3.1 A Single Random Walk

Every random walk in TrustWalker starts from source user u_0. At each step k of a random walk, we are at a certain node u. If u already has the rating on target item i, then the random walk stops and returns $r_{u,i}$ as the result of random walk. If u does not have a rating on i, then there are two options:

- With probability $\phi_{u,i,k}$, the random walk stops at node u and randomly selects one of the items (j) similar to i rated by u and return $r_{u,j}$.
- With probability $1 - \phi_{u,i,k}$, the random walk continues to another user v who is one of u's direct trusted neighbors ($v \in N_u$).

If the random walk decides to continue at node u, one of directly trusted neighbors of u has to be selected for the random walk to continue the walk to that user. We define S_u as the random variable for selecting a user v from N_u:

$$P(S_u = v) = \frac{sim^+(u, v)}{\sum_{w \in N_u} sim^+(u, w)} \tag{4.1}$$

where $sim^+_{u,v}$ denotes the similarity of u and v which is positive number in range [0,1]. The probability of continuing the random walk to a direct neighbor v is proportional to the similarity

of u and v. The similarity measure is defined later in this section. It should be noted that in our previous work [47], we did not take the similarity into account and considered a uniform distribution for S_u among neighbors of u. Note that considering the user similarities for transition probabilities represents the modeling of correlational influence discussed in Chapter 3.

Now, the probability of being at node v in step $k+1$ while looking for a prediction on target item i for source user u_0 and being at node u in previous step is defined as follows:

$$P(X_{u_0,i,k+1} = v | X_{u_0,i,k} = u, \widetilde{R_{u,i}}) = (1 - \phi_{u,i,k}) \times P(S_u = v)) = (1 - \phi_{u,i}) \times \frac{sim^+(u,v)}{\sum_{w \in N_u} sim^+(u,w)} \quad (4.2)$$

Here, $X_{u_0,i,k}$ denotes the random variable for being at node v in step k while looking for a prediction on target item i for source user u_0. Details of computing $P(X_{u,i,k} = v)$ are discussed later. Also we have a condition that the user u in step k-1 does not have the rating for item i (denoted by $\widetilde{R_{u,i}}$). The probability of walking from user u to v is independent of previous steps. However, since $\phi_{u,i,k}$ depends on the step k, it is not independent of the step of the random walk.

With probability $\phi_{u,i,k}$, if the random walk decides to stay at a user u, one of the items rated by u which is similar to the target item i is selected and the rating expressed on that item is returned as the result of this random walk. The idea is that we define a similarity measure between items, and for each item $j \in RI_u$, we assign a probability of selecting proportional to the similarity of i and$j$. We discuss the details of the similarity metric later.

$$P(Y_{u,i} = j) = \frac{sim(i,j)}{\sum_{l \in R_u} sim(i,l)} \quad (4.3)$$

In this equation $Y_{u,i}$ denotes the random variable for selecting item j amongst items rated by u while looking for an item similar to target item i. We return $r_{u,j}$ as the result of this random walk.

To define the whole probability distribution, we define the probabilities for conditions where $R_{u,i}$ is true, or for items not rated by u as follows:

$$\forall v \neq u \;\; P(X_{u_0,i,k+1} = v | X_{u_0,i,k} = u, R_{u,i}) = 0 \quad (4.4)$$

$$\forall j . j \notin R_u \;\; P(Y_{u,i} = j) = 0 \quad (4.5)$$

Item Similarities

In content-based recommendation, the similarity of items can be computed using their features. However in collaborative filtering, the only information available about items is their ratings. Hence, to compute the similarity of two items, we use the *Pearson Correlation* of ratings expressed for both items, as used in [101]. Values of the Pearson correlation are in the range [-1,1]. Negative correlations mean that the ratings expressed for two items are in opposite directions, so these items are not useful for our purpose. Therefore, we only consider items with positive correlation. As discussed in Section 2.4.1, the Pearson correlation is computed as follows:

$$corr(i,j) = \frac{\sum_{u \in C_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in C_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,i} - \overline{r}_u)^2}} \tag{4.6}$$

$C_{i,j}$ is the set of common users who have rated both items i and j ($R_i \cap R_j$), and $\overline{r}_u$ denotes the average of ratings expressed by u. $corr(i,j)$ denotes the correlation of items i,j.

The size of the set of common users is also important. For example, if $corr(i,j) = corr(i,l)$, but $|C_{i,j}| > |C_{i,l}|$, then, since i and j have been rated by more common users, so the correlation between them is stronger and $sim(i,j)$ should be greater than $sim(i,l)$. We consider $|C_{i,j}|$ in the similarity measure as follows:

$$sim(i,j) = \frac{1}{1 + e^{-\frac{|C_{i,j}|}{2}}} \times corr(i,j) \tag{4.7}$$

We used the sigmoid function to avoid favoring the size of $C_{i,j}$ too much and to keep the similarity value in the range [0,1]. If the size of the set of common users is big enough, then the first part of Equation (4.7) would converge to 1, but for small sets of common users, the factor would be 0.6. The number 2 in the denominator of the exponent is because we wanted to have a factor of greater than .9 if the size is greater than 5.

User Similarities

Similarity of two users is defines similar to the definition of similarity of items.

$$sim(u,v) = \frac{1}{1 + e^{-\frac{|C_{u,v}|}{2}}} \times corr(u,v) \tag{4.8}$$

where $C_{u,v}$ is the set of common items that have been rated by both users u and v ($R_u \cup R_v$). Also $corr(u,v)$ denotes the Pearson correlation of the ratings expressed by u and v.

The values of $sim(u, v)$ are in range [-1,1]. However, in our model we use similarity to compute probabilities. Hence, we scale the similarities to the range [0,1] as follows:

$$sim^{+}(u, v) = \frac{1 + sim(u, v)}{2} \tag{4.9}$$

Termination of a Single Random Walk

At each user u, the random walk has a probability $\phi_{u,i,k}$ of staying at u to select one of his items at step k of the random walk, while looking for a prediction on target item i. This probability should be related to the similarities of items rated by u and the target item i. Similarity values are real numbers in $[0, 1]$, so they can also be considered as probabilities. We consider the maximum similarity of items rated by u with target item i as the probability of staying at node u.

Furthermore, ratings on target item i from users far away from source user u_0 are noisy, but ratings expressed by trusted users nearby in the network are more reliable. So, the deeper we go into the network, the probability of continuing our random walk should decrease and so $\phi_{u,i,k}$ should increase.

To inject the factor k in $\phi_{u,i,k}$, we should use a function $f(k)$ which gives value 1 for big values of k, and a small value for small values of k. Since the sigmoid function satisfies these constraints, we consider a sigmoid function of the k as another factor affecting $\phi_{u,i,k}$:

$$\phi_{u,i,k} = \max_{j \in R_u} sim(i, j) \times \frac{1}{1 + e^{-\frac{k}{2}}} \tag{4.10}$$

Each random walk has three alternatives to stop:

1. Reaching a node which has expressed a rating on the target item i.

2. At some user node u, we decide to stay at the node and select one of the items rated by u and return the rating for that item as the result of random walk.

3. There is chance for a single random walk to continue for ever. To avoid such a case in our implementation of random walk, we terminate the random walk when we go very far from the source user ($k >$ *max-depth*). Based on the idea of "six-degrees of separation" [79], we set *max-depth* $= 6$.

4.3.2 Rating Prediction in TrustWalker

In TrustWalker, we have the probability of selecting items rated by different users and returning that rating as the result of a random walk. These items could be either the exact target item i, or another item. The estimated rating for source user u on target item i would be the expected value of ratings returned by different random walks.

$$\widehat{r}_{u,i} = \sum_{\{(v,j)|R_{v,j}\}} P(XY_{u,i} = (v,j))\, r_{v,j} \tag{4.11}$$

In the above equation, $XY_{u,i}$ is the random variable for stopping the random walk at node v and selecting item j rated by v, while we start the random walk from source user u looking for target item i. Notice that the value for XY are ordered pairs. As used before, $R_{v,j}$ is a boolean variable denoting whether v has a rating on item j. Basically, $P(XY_{u,i} = (v,j))$ is the steady state probability that a random walker starting from u looking for target item i stays at node v and item j. Now we have:

$$P(XY_{u,i} = (v,j)) = \begin{cases} P(X_{u,i} = v)\phi_{u,i}P(Y_{v,i} = j) & v \neq u; i \neq j \\ P(X_{u,i} = v) & v \neq u; i = j \\ \phi_{v,i,1}P(Y_{v,i} = j), & v = u; i \neq j \end{cases} \tag{4.12}$$

In this equation, $X_{u,i}$ is the random variable for being at node v at some step in a random walk starting from source user u looking for item i. Notice that in above formula, we used $\phi_{u,i}$ instead of $\phi_{u,i,k}$ for the first case. Since we do not know the number of steps needed to reach v, we do not consider the factor k (Actually $\phi_{u,i} = \phi_{u,i,\infty}$). It should be noted that if we actually perform random walks, we can consider the step k in the first case. But to have a closed form formula, we ignore the factor k at the last user v which gives us a pretty good approximation of the probability. Also, we should note that the case $v = u$ and $i = j$ is trivial since the user himself has the rating on the target item.

A random walk starting from u can reach v using different number of steps. As mentioned before, we use random variable $X_{u,i,k}$ for being at node v in k steps

$$P(X_{u,i,k} = v) = \sum_{w \in U} P(X_{u,i,k-1} = w)(1 - \phi_{w,i,k})P(S_w = v) \tag{4.13}$$

Also we have $P(X_{u,i,0} = u) = 1$ as the base for the above equation. Since the random walks have a probability of stopping at each step, $\sum_{v \in U} P(X_{u,i,k} = v) \neq 1$. To make $P(X_{u,i,k} = v)$ a probability distribution, we define a *dead state* $\perp$ to which all users go after deciding to terminate that random walk. So, we have

$$P(X_{u,i,k} = \perp) = 1 - \sum_{v \in U} P(X_{u,i,k} = v)$$

This state $\perp$ will be added to U for convenience in formalization of our method, but we do not consider this state in any actual random walk. Now, we can compute $P(X_{u,i} = v)$ as follows:

$$P(X_{u,i} = v) = \frac{\sum_{k=1}^{\infty} P(X_{u,i,k} = v)}{\sum_{w \in U} \sum_{k=1}^{\infty} P(X_{u,i,k} = w)} \tag{4.14}$$

4.3.3 Matrix Notation of TrustWalker

Similar to any random walk model, we can represent TrustWalker using matrix notations. We consider a probability matrix $\mathbf{P}$ in which $\mathbf{P}_{u,v} = P(S_u = v) = \frac{1}{|N_u|}$ for all users u and v for which $t_{u,v} = 1$. To formulate the values of $\phi_{u,i,k}$ in matrix notation, we define a diagonal matrix $\mathbf{\Phi}_{k,i}$ for each item i and step k. $\mathbf{\Phi}_{k,i}$ is a $|U| \times |U|$ matrix containing $1 - \phi_{u,i,k}$ in its diagonal elements. In other words, $\mathbf{\Phi_{k,i}}_{u,u} = 1 - \phi_{u,i,k}$.

It is easy to check that the elements of $\mathbf{\Phi_{1,i}P}$ are the probabilities of reaching from u to v in step 1. We can also define a closure on the $\mathbf{\Phi_{k,i}P}$ as follows:

$$\mathbf{P_i^*} = \sum_{K=1}^{\infty} \prod_{k=1}^{K} \mathbf{\Phi_{k,i}P} = \mathbf{\Phi_{1,i}P} + \mathbf{\Phi_{1,i}P\Phi_{2,i}P} + \mathbf{\Phi_{1,i}P\Phi_{2,i}P\Phi_{3,i}P} + ...$$
$$= \mathbf{\Phi_{1,i}P}(\mathbf{I} + \mathbf{\Phi_{2,i}P}(\mathbf{I} + \mathbf{\Phi_{3,i}P}(...))) \tag{4.15}$$

$\prod_{k=1}^{K} \mathbf{\Phi_{d,i}P}$ is the matrix containing $P(X_{u,i,K}) = v$ in its cells. In the above equation, k denotes the current step of each single random walk and K denotes then number of steps each random walk has. Now, we can compute $\widehat{\mathbf{P}}_\mathbf{i}$ (corresponding to Equation (4.14)) which is the probability matrix containing $P(X_{u,i} = v)$ in its elements as follows:

$$\widehat{\mathbf{P}}_\mathbf{i} = \mathbf{C_i P_i^*} \tag{4.16}$$

Here, $\mathbf{C_i}$ is an $N \times N$ diagonal matrix used for normalization. The values of the diagonal are $\mathbf{C_i}_{u,u} = \frac{1}{\sum_{w \in U} \mathbf{P_i^*}_{u,w}}$. Now we can use $\widehat{\mathbf{P}}_\mathbf{i}$ to compute $P(X_{u,i} = v)$ in Equation (4.14).

If we do not consider step k as a factor in $\phi_{u,i}$, it can be proved that $\mathbf{P_i^*}$ has a closed formula solution, and this closed formula is $(\mathbf{I} - \mathbf{\Phi_i P})^{-1}\mathbf{\Phi_i P}$. In this formula, $\mathbf{\Phi_i}$ is the same as $\mathbf{\Phi_{k,i}}$ ignoring the factor k. In Appendix A.1, we provide the proof of convergence of $\mathbf{P_i^*}$ if we ignore step k in computing $\phi_{u,i,k}$

Since matrices $\mathbf{\Phi_{k,i}}$ are different for different k values, we can not compute a closed formula for $\mathbf{P_i^*}$. We propose two alternatives to compute Equation (4.15):

- Performing the random walks. This way, we can see the results in action and the estimated value would be the aggregation of results of different random walks.
- Based on the idea of "six degrees of separation" [79], most nodes would be reachable with a walk of length at most 6. So, we can have a pretty good approximation of $\mathbf{P_i^*}$ by

$$\mathbf{P_i^*} = \sum_{K=1}^{6} \prod_{k=1}^{K} \mathbf{\Phi_{k,i}} \mathbf{P} \tag{4.17}$$

This formula can be easily computed, but there is a problem with the second approach. $\mathbf{P_i^*}$ associated with each item hast to be stored, which is expensive. For example, if there are 10K items and 40K users, and each cell of the matrix occupies just one byte, then each matrix $\mathbf{P_i^*}$ would occupy almost $1.6GB$ memory. To store all matrices we would need $16TB$ of memory, which is not feasible. This issue motivates performing actual random walk and aggregate the results of different random walks. Moreover, computing this matrix needs a global information on the whole network, but TrustWalker can be computed in in a local manner.

Notice that there are major differences between our random walk model and existing random walk approaches such as [20] and [115]. In PageRank [20], there is a random walk on the links among WebPages, which correspond to users. But there is no item in PageRank and walks do not depend on the step of the random walk. Moreover, PageRank is a global reputation metric. These differences make PageRank simpler than TrustWalker to find the closed form solution. Nevertheless, they do not compute the closed formula because of the scale of network. In [115], they have an item graph on which they perform the random walk. This graph is independent of users, and hence they do not perform random walks on users. Also, unlike TrustWalker, their walks are independent of step k of the random walk. Having a simpler model, they are able to compute the closed formula for walking on the item graph.

4.3.4 Termination of the Overall Method

The results of performing actual random walks approximate the results given by Equation (4.15). TrustWalker performs several random walks to be able to get a more reliable prediction. TrustWalker needs to be able to decide when we have done enough random walks to have a precise estimate of $\widehat{r}_{u,i}$.

TrustWalker computes the variance in the results of all the walks as follows:

$$\sigma^2 = \frac{\sum_{i=1}^{T} (r_i - \overline{r})^2}{T}$$

Here, r_i is the result of i^{th} random walk, and $\overline{r}$ denotes the average of the ratings returned by random walks. T is the number of random walks we perform to compute the prediction. We also define σ_i^2 as the variance in the results of the first i random walks. Since the values of ratings are in finite range of [1,5], it can be proved that σ^2 converges to a constant value (see the Appendix A.2). So TrustWalker terminates if $|\sigma_{i+1}^2 - \sigma_i^2| \leq \epsilon$.

It should be noted that we have a constant threshold of 10000 for the maximum number of unsuccessful random walks, and after that we consider the pair $< user, item >$ as non-covered.

4.3.5 Special Cases of TrustWalker

TrustWalker includes Item-based Collaborative Filtering and pure Trust-based Recommendation as its extreme special cases. If $\phi_{u,i} = 1$ for all $u \in U$, then our random walk will never start, and it will return the rating expressed by the source user u_0 on one of its rated items. Since the probability of selecting an item is proportional to its similarity to the target item i, the expected value of the recommended rating would be the weighted average of the ratings on items in RI_{u_0} with weights proportional to the similarities of these items to the target item i. This is the same as the result of Item-based collaborative filtering proposed in [101].

On the other hand, if we set $\phi_{u,i} = 0$ for all $u \in U$, then all random walks will continue until they have found a rating for the exact target item i. The recommended rating would be the aggregation of ratings expressed by users having the rating on i weighted by the probability of reaching these users from u_0. Existing methods [36][74] try to approximate these probabilities by simplifying the problem. So our TrustWalker, in one of its extreme cases, can be considered as an ideal trust-based recommender.

4.3.6 Notes on computational complexity of TrustWalker

Every single walk in TrustWalker has the length of D on average, where D is the diameter of the social network. In every step of the random walk, similarity of a user with all its neighbors is computed. Also, similarity of the items rated by a user and the target item is computed. To compute the similarity of two users, items rated by them are compared. Also to compute the similarity of two items, users rating these two items are compared. Overall, the complexity of a single random walk is as follows:

$$D \times (\frac{|E|}{|U|}\frac{|R|}{|U|} + \frac{|R|}{|U|}\frac{|R|}{|I|})$$

where $\frac{|E|}{|U|}$ is the average number of neighbors per user, $\frac{|R|}{|U|}$ is the average number of ratings per user, and $\frac{|R|}{|I|}$ is the average number of ratings per item. Complexity of rating prediction is T times the complexity of a random walk, where T is the number of walks TrustWalker performs to compute the prediction.

4.4 LinkWalker: Top-N Link Prediction Model

In this section we employ the ideas introduced in TrustWalker and propose the LinkWalker model to address the top-N link prediction problem. LinkWalker is in fact an extension of the RWR model where the transition probabilities and restart probabilities are modified to capture the similarity of user rating patterns. As discussed before, LinkWalker incorporates both transitivity and social selection discussed in Chapter 3 to compute the top-N recommended users.

TrustWalker performs a series of random walks to predict the rating for a target item. However, in the top-N link prediction problem, the goal is to recommend trustworthy users rather than predicting ratings for a given item. In TrustWalker, after each random walk stops, one of the items rated by the current user will be randomly selected, and the rating expressed for that item is returned as the result of the random walk. However, in LinkWalker, there is no target item and the user at whom the random walk stops is returned as the result of the random walk (see Section 4.4.1). Furthermore, LinkWalker returns a list of users instead of a single predicted rating in TrustWalker (see Section 4.4.2).

To recommend top-N links to a source user u_0, LinkWalker performs random walks on the trust network each starting at user u_0 to find users who are likely trustworthy for u_0. The details of the random walk will be discussed in SEction 4.4.1. Each random walk returns a recommended user

to create the trust link to. LinkWalker performs a series of random walks and uses the N most frequently returned users as top-N recommended users to create trust link to. Unlike TrustWalker, there is no target item, but only a source user. Therefore, the termination criteria in a random walk in LinkWalker is different from that of TrustWalker and needs to be independent of a specific item (see Section 4.4.1). Since LinkWalker is returning a list of users rather than a single rating prediction, the termination criteria for the overall method in LinkWalker is different from that of TrustWalker. TrustWalker uses the variance of the recommended ratings to decide on the termination, while in LinkWalker the stability of the list of top N recommended users is used for the termination criteria (see Section 4.4.3).

4.4.1 A single random walk in LinkWalker

Every random walk starts at user u_0. At each step k, the random walk is at a certain node u. Now there are two alternative options:

- With probability $\phi_{u,u_0,k}$, the random walk does not continue. It stops at user u and returns u.
- With probability $1 - \phi_{u,u_0,k}$, the random walk continues to another user v who is one of u's directly trusted neighbors $v \in N_u$.

If the random walk decides to continue walking at node u, one of the directly trusted neighbors of u has to be selected to continue the random walk to that node. The random variable S_u for selecting a user v from N_u is defined in the same way as for TrustWalker, as defined in Equation (4.1).

Now, the probability of being at node v in step $k+1$ while looking for a recommended trustworthy user for u_0 and being at node u in step k is defined as follows:

$$P(Z_{u_0.k+1} = v | Z_{u_0,k} = u) = (1 - \phi_{u,u_0,k}) \times P(S_u = v) = (1 - \phi_{u,u_0,k}) \times \frac{sim^+(u,v)}{\sum_{w \in N_u} sim^+(u,v)} \quad (4.18)$$

Here, $Z_{u_0,k}$ denotes the random variable for being at node v in step k while looking for a recommended user for u_0. Note that the termination probability in LinkWalker, $\phi_{u,u_0,k}$, is different from that in TrustWalker ($\phi_{u,i,k}$). In LinkWalker, the termination criteria depends on the similarity of the source user u_0 and the current user u. However, in TrustWalker, the termination criteria is

independent of the source user and depends on the similarity of items rated by u and the target item i.

Termination of a single random walk in LinkWalker

At each user u,there is a probability $\phi_{u_0,u,k}$ of terminating the random walk and returning u as the result of that random walk. Different from TrustWalker, the termination criteria in LinkWalker depends on the similarity of the source user u_0 and the current user u, rather than depending on the similarity of the target item and the items rated by u, as used in TrustWalker. Similar to TrustWalker, the current step k is taken into account in LinkWalker by defining $\phi_{u_0,u,k}$ as follows:

$$\phi_{u_0,u,k} = \begin{cases} 0 & u = u_0 \text{ or } u \in N_{u_0} \\ 1 & |N_u| = 0 \\ sim^+(u,u_0) \times \frac{1}{1+e^{-\frac{k}{2}}} & \text{otherwise} \end{cases} \tag{4.19}$$

If u has no neighbors and $u \in N_{u_0}$, then the random walk returns no user.

4.4.2 Top-N Link Prediction in LinkWalker

In LinkWalker, the results of all random walks are collected to create a list of recommended users. Each user u in this list is associated with a frequency denoting the number of times u has been returned as the result of a random walk. The list of users is sorted with respect to the frequency, and LinkWalker returns the top N users in this list as the top N recommended users to create trust link to.

Similar to TrustWalker, we can formulate the LinkWalker model with matrix notations. Authors of [89] have proposed a fast algorithm to approximate the probability of reaching user u in a random walk starting from u_0 for all pairs (u_0, u). However, in LinkWalker, we do not need to compute the probability for all pairs of users. We only need to compute the top-N high probable users reachable from u_0. The proposed recommendation model in this section approximates this by actually performing the random walks and considering the top-N users in the list of users returned by different random walks as the top-N probable users reachable from u_0.

4.4.3 Termination of the overall method in LinkWalker

LinkWalker performs several random walks to get a more reliable recommendation. LinkWalker needs to be able to decide when enough random walks have been done to produce a precise list of top-N recommended users.

After every K random walks, LinkWalker computes a weight vector for users in the list of top-N users and compares it to the current weight vector. To compute the weight for each user u, denoted by w_u, in the list of top-N users, the frequency associated with u in the list is divided by the sum of the frequencies of all top-N users.

Suppose that the list of top-N users from previous random walks is L'. Every user $u \in L'$ is associated with a weight w'_u. After every K random walks, this list is updated to a new list L with associated weights w_u for users $u \in L$. To compare two lists of top-N users L and L', we could compute the distance of the two lists as follows:

$$diff(L, L') = \sqrt{\sum_{u \in L \cap L'} (w_u - w'_u)^2}$$

The above equation computes the "Euclidian distance" of the weight "vectors" including users who appear in both lists L and L'. However, the lists may have only a few users in common and most of the users in the lists may be different. To penalize lists having many users not in common, LinkWalker assigns the maximum distance 1 to every user in L that does not appear in L' and defines the difference between lists L and L' as follows:

$$diff(L, L') = \sqrt{\sum_{u \in L \cap L'} (w_u - w'_u)^2 + \sum_{u \in L, u \notin L'} 1} \tag{4.20}$$

The weights are in the range [0,1]. Hence the terms $(w_u - w'_u)^2$ are also in the range [0,1], and each single penalty has the maximum value of one. LinkWalker continues performing further random walks until $diff(L, L') \leq \epsilon_1$.

4.4.4 LinkWalker as generalization of RWR

Random walk based methods have already been used to perform link prediction [116]. Actually, the RWR model [89] can be considered as a special case of LinkWalker. In RWR, there is a probability c for the random walk to restart. If we choose a fixed termination criteria in LinkWalker by setting $\phi_{u,u_0,k} = c$ and employ a uniform walking process by setting $P(S_u = v) = \frac{1}{|N_u|}$, then LinkWalker

will be identical to RWR. In other words, LinkWalker is a generalization of RWR which takes into account the similarity of users' rating patterns and the length of random walks.

4.5 Social Network-based Approaches to Top-N Recommendation

In this section, we propose novel memory-based approaches that exploit social networks to improve the quality of top-N recommendation. The first approach is an extension of the idea proposed in TrustWalker and the second method is a combined model aggregating a social network-based method and a collaborative filtering based approach.

4.5.1 Random Walk Approach

In this subsection, we extend TrustWalker to recommend top-N items for a source user u. Note that the problem definition is to compute a list of top-N desired items for a given user u. Starting from user u, we perform a random walk on the social network. Each random walk stops at a certain user. Then the items rated highly by that user will be considered as the recommended items, ordered according to the ratings expressed by that user. We perform several random walks to gather more information and compute a more confident result. The estimated rating of each item is the average of ratings for that item over all raters considered. At the end, we output items with the highest estimated rating as top-N recommended items. Similar to TrustWalker [47], we compute the variance of the estimated rating for items and continue performing further random walks until the variance converges.

Now, we discuss the details of a single random walk. We start each random walk from the source user u. At each node v, with probability $\phi_{u,v}$, we stop the random walk and return items rated by v. With probability $1 - \phi_{u,v}$, we continue the random walk to one of the neighbors of v. We select the neighbor of v uniformly from directly trusted neighbors of v. $\phi_{u,v}$ depends on the similarity of user v with the source user u. The more similar they are the more likely the random walk stops at v. Also, the further away from the source user we go in the network, the probability of stopping the random walk should get higher to avoid noisy data located far from the source user u. So we denote the probability of stopping at node v by $\phi_{u,v,k}$ where k is the current step of the random walk. We define this probability as follows:

$$\phi_{u,v,k} = (0.5 + \frac{sim_{u,v}}{2}) \times \frac{1}{1 + e^{-k/2}} \tag{4.21}$$

In the above equation, $sim_{u,v}$ is computed using Equation (4.8). Since the values of $sim_{u,v}$ are in the range [-1,1], we shift the values of similarity to get a value in the range [0,1] that can be interpreted as a probability. Basically, if v is not similar to u at all ($sim_{u,v}$ is a very small positive number), this offset allows the random walk to still have the possibility to stop at v. However, if $sim_{u,v}$ is close to -1, the probability of stopping at v would be very close to zero, and the random walk will most likely continue to find another user in the network.

Notice that the factor $\frac{1}{1+e^{-k/2}}$ is a sigmoid function which injects the effect of the current step of random walk into the stopping probability. $\phi_{u,v,k}$ depends on the similarity of users u and v, while the stopping criteria in [47] depends on the similarity of items rated by v and the target item. This is due to the different tasks being performed, in particular there is no target item in top-N recommendation. The intuition behind the new stopping probability in this section is that users with similar rating patterns are more likely to agree on their top-N items.

It should be noted that we also associate a parameter $maxDepth$ with the random walk approach which determines the maximum depth to which a random walk can be continued. Random walks are forced to terminate after $maxDepth$ steps.

4.5.2 Combined Approach

When a user u trusts another user v it does not necessarily mean that they rate the same items. Basically, when u trusts v, it means that if they both rate an item, it's more likely for them to rate this item in a similar way. So, if we use leave-one-out method and ask the trust-based approach to recommend top-N items, it may not be able to recommend the exact withheld item, although the recommended items are actually interesting for the source user. On the other hand, in the collaborative filtering approach, we consider users who have similar rating patterns. Two users who have already rated some common items, tend to rate more items in common. Hence, it's more likely for users with similar rating patterns to the source users to also rate the withheld item.

Our experiments confirm this effect, which causes collaborative filtering to slightly outperform the trust-based random walk. Therefore we propose a combined approach to benefit from properties of both trust-based and collaborative filtering approaches.

In this approach, we compute the top K trusted users in the network and rank the items rated by these trusted users to compute top-N recommended items. We use the collaborative filtering approach to compute another set of top-N recommended items. Finally we merge these two lists to produce a combined lit of top-N recommended items.

This approach uses similar users which are more likely to have the withheld items, and it also uses a trust network to deal with cold start users. Notice that half of the users are cold start users, and collaborative filtering is not successful in finding similar users for these users.

To compute top K trusted users, we can use two different alternatives:

- Breadth First Search.
- Random Walk in the social network.

Breadth First Search (BFS). In this approach, we perform a BFS to find k_2 closest users to the source user u in the social network. Then, we merge the items rated by these trusted users to find the top-N recommended items returned by social network-based approach (We denote these items as TR_u). We estimate the rating of items rated by trusted neighbors as follows:

$$\hat{r_{tu,i}} = \frac{\sum_{v \in Nt_u, i \in I_v} w_t(u,v) \times r_{v,i}}{\sum_{v \in Nt_u, i \in I_v} w_t(u,v)} \tag{4.22}$$

In the above equation, $\hat{r_{tu,i}}$ denotes the estimated rating using the trusted neighborhood. Also Nt_u denotes the top k_2 trusted users found by BFS. $w_t(u,v)$ is the influence coefficient of each user in the trusted neighborhood. We define $w_t(u,v) = 1/d_v$, where d_v is the depth at which we found v in the BFS starting from u.

Random Walk. In the random walk approach, we perform random walks similar to the ones introduced in the previous subsection. Each random walk stops at a certain user. This user will be considered as a trusted user. We continue performing random walks until we get k_2 users. The rest is the same as BFS approach, with $w_t(u,v)$ equal to the number of times user v has been returned as the result of a random walk.

After finding top k_2 trusted users (with either approach), we combine the results of the trust-based approach and the CF based approach. Suppose that in the CF based approach we use k_1 top similar users and merge their items. We denote the items returned by CF approach as CF_u. Now we define the merge method as follows:

$$\hat{r}_{u,i} = \begin{cases} \frac{\hat{r_{cu,i}} + \hat{r_{tu,i}}}{2} & i \in TR_u \; ; i \in CF_u \\ \hat{r_{tu,i}} & i \in TR_u \; ; i \notin CF_u \\ \hat{r_{cu,i}} & i \in CF_u \; ; i \notin TR_u \end{cases} \tag{4.23}$$

The top-N items with highest values of $\hat{r}_{u,i}$ will be returned as the top-N recommended items. It should be noted that we could do a weighted averaging instead of just taking the means for items which appear in both TR_u and CF_u. To consider the weights, we define $\hat{r}_{u,i}$ as follows:

$$\hat{r}_{u,i} = \frac{k_1 \times \hat{r_{cu,i}} + k_2 \times \hat{r_{tu,i}}}{k_1 + k_2}$$

We discuss the effect of weighted merging of results of CF approach and social network-based approach in the experiments section.

4.6 Data Sets

In this section we briefly introduce the data sets used in our experiments: the Flixster data set, which we crawled and prepared, and the public domain Epinions data set. The data sets introduced in this section are the basis for most of our experiments in the next chapters.

4.6.1 Epinions data set

To the best of our knowledge, before publication of the Flixster data set, the Epinions[1] data set was the only trust network data set publicly available that also includes ratings expressed by users. Epinions.com is a product reviewing websites in which users can express trust on other users besides writing reviews on different products.

We used the version of the Epinions data set[2] published by the authors of [98]. Table 4.2 shows the statistics of the Epinions data set. Each user has on average 8.1 expressed ratings and 7.2 direct neighbors. The social relations in Epinions are directed. The distribution of the number of ratings per user follows a power law. The items in Epinions are products from different categories such as cameras, dvd players, music, etc. Users are allowed to write reviews on products, rate items (products), or both. Also, users can rate the reviews written by other users on products. In this dissertation, we only use the ratings expressed by users on items and do not consider the content of reviews.

[1] www.epinions.com

[2] http://alchemy.cs.washington.edu/data/epinions/

Table 4.2: General statistics of the Flixster and Epinions

Statistics	**Flixster**	**Epinions**
Users	1M	71K
Social Relations	26.7M	508K
Ratings	8.2M	575K
Items	49K	104K
Users with Rating	150K	47K
Users with Neighbors	980K	60K

4.6.2 Flixster data set

We crawled a large scale data set from the Flixster website[3]. Flixster is a social networking service in which users can rate movies. Users can also add some users to their friend list and create a social network. Note that unlike Epinions, social relations in Flixster are undirected. General statistics of the Flixster data set is presented in Table 4.2.

Flixster has many ways to gather ratings from users. First, users can go online, log into Flixster.com and rate some movies or make friendship connections to other users. In addition, Flixster uses some applications to gather ratings from users. There are two Flixster applications which are very famous in Facebook[4] and Myspace[5]. When users of facebook or myspace install these applications on their profile, they will be asked to rate different movies. Initially, they are presented with a fixed set of 50 movies to rate. Many users of Flixster rate only these movies. As a results the rating counts for these 50 movies are extraordinarily high. Hence, we remove the ratings for these movies to reduce the bias of the data set. Although this information could be useful in a real recommender system, we ignore it to have a less biased data set in our research.

Possible rating values in Flixster are 10 discrete numbers in the range [0.5,5] with step size 0.5. Users are also allowed to rate the items with two other nominal values: “Want To See” and “Not Interested”. These two types of rating also include some information and may be useful for some research purposes. However, nominal ratings are not considered in the rest of this dissertation since we can not easily convert them to numerical rating values. Almost half of the ratings are numerical values. The statistics in Table 4.2 are only for the numerical ratings.

[3] www.flixster.com

[4] http://apps.facebook.com/flixster/

[5] http://www.myspace.com/flixstermovies

The Flixster data set contains ratings expressed by users during the period from November 2005 to November 2009. The data set is now publicly available at www.cs.sfu.ca/∼sja25/personal/datasets/ .

According to the statistics presented in Table 4.2, the Flixster data set is denser than the Epinions data set. The number of ratings and neighbors per user is larger in the Flixster data set. A large portion of users in Flixster data set have no expressed ratings, but most of them have social relations. Users without any ratings are also important. They may not be useful to compute the prediction for other users based on their own ratings, but they may allow us to connect indirectly to other users who have rated items.

We consider users with less than 5 ratings as cold start users (similar to [74]). 49% of users in Epinions are cold start users which is a very large portion of users. Also 53% of users in Flixster who have rated at least one item are cold start users. So, considering the performance of the recommendation for cold start users is very important.

Note that the Flixster data set is a binary social network and not an explicit trust network. However, as discussed before, we consider it as a trust network in this thesis. Also, unlike Epinions, Flixster is undirected. We convert each undirected edge into two directed edges.

4.7 Experiments

In this section, we present our experimental results for TrustWalker, LinkWalker, and for top-N recommendation.

4.7.1 Experiments with TrustWalker

This subsection reports our experimental results comparing TrustWalker against state-of-the-art methods for trust-based and item-based recommendation. We implemented TrustWalker as well as the MoleTrust recommendation proposed by Massa [74] and TidalTrust proposed by Golbeck [36]. We also implemented standard user-based collaborative filtering [37] and item-based collaborative filtering [101] as two fundamental similarity-based recommendation methods.

Comparison Partners for TrustWalker

In our experiments, we compare the results for different methods. Following is the description of labels we use to denote each of these algorithms:

- *TidalTrust*. This is the trust-based approach of [36].
- *MoleTrust*. This is the approach used in [74]. It should be noted that we use $max-depth = 6$ for *ModelTrust* as well.
- *CF Pearson*. We implemented the user-based collaborative filtering [37], with the Pearson Correlation as similarity measure.
- *Item-based*. We also implemented the item-based collaborative filtering [101] using Pearson Correlation as the item similarity metric.
- *Random Walk* This is one of the special cases of TrustWalker with $\phi_{u,i,k} = 0$ for all (u, i, k). Also we set different thresholds on the number of steps a random walk. *Random Walk 1* represents the case in which we just walk for one step, and in *Random Walk 6* each random walk could continue until 6 steps.
- *TrustWalker*. This is the TrustWalker method introduced in this Chapter.
- *TrustWalker-orig*. This is the TrustWalker method originally introduce in [47]. This version of TrustWalker does not consider the similarity of users in the probability in computing $P(S_u = v)$.

We set $\epsilon = 0.0001$ for our termination condition. Note that in [47], we performed experiments on several settings of the parameters in TrustWalker. For instance, we reported experimental results on a version of TrustWalker where $\phi_{u,i}$ is independent from k, or the version of TrustWalker where similarity of two items i, j is independent from the size of the set of common users rating them ($UC_{i,j}$). The results showed that the complete model where all the parameters are taken into account outperforms the other models. For the details please refer to the original paper [47].

Evaluation Metrics

To perform experiments, we split the data set into train and test data. We used 5-fold cross validation in our experiments. As used in the most recent research papers [115] [58], we use the Root Mean Squared Error (RMSE) to measure the error in recommendation (see Section 2.3 for details). The smaller the value of RMSE, the more precise a prediction.

Exploiting social network in recommenders does not necessarily enhance the precision of prediction, but it allows to compute the prediction for more pairs of (u, i). In other words, using a social

network enhances the coverage of a recommender system without sacrificing the precision. So we define the *coverage* metric which is the percentage of pairs of $< user, item >$ for which we can predict a rating.

We also use *F-measure* as an evaluation metric. *F-measure* is a well-known measure of accuracy in the information retrieval and machine learning community that considers both precision and coverage [9]. *Precision* is conceptually the opposite of the error metric, and precision values are in the range [0,1]. Since RMSE is in the range [0,4], we use the following formula to convert RMSE into a precision metric:

$$Precision = 1 - \frac{RMSE}{4} \tag{4.24}$$

Now we adopt the standard definition of F-measure [9] as follows:

$$FMeasure = \frac{2 \times Precision \times Coverage}{Precision + Coverage} \tag{4.25}$$

If none of the random walks can find a prediction on the rating, then we say that the recommender can not cover this pair of $< user, item >$.

Experimental Results with TrustWalker

Table 4.3 shows the RMSE, Coverage, and F-Measure for all comparison partners in Epinions, once for cold start users and once for all users. Table 4.4 shows the same measures for the Flixster data set. In the following, we first discuss the results for cold start users and then for all users.

Table 4.3: Experimental results for cold start users and all users in Epinions.

Method	Cold Start Users			All Users		
	RMSE	Coverage(%)	F-Measure	RMSE	Coverage(%)	F-Measure
Item-based	1.551	21.26	0.316	1.232	68.91	0.691
CF Pearson	1.498	16.34	0.259	1.277	67.54	0.688
MoleTrust	1.441	55.36	0.594	1.104	81.03	0.765
TidalTrust	1.223	56.92	0.626	1.109	82.37	0.770
RandomWalk 1	1.105	11.68	0.201	1.168	29.12	0.413
RandomWalk 6	1.221	58.71	0.636	1.101	85.14	0.783
TrustWalker-orig	1.209	70.32	0.700	1.097	93.31	0.816
TrustWalker	1.201	70.17	0.701	1.079	93.22	0.819

Table 4.4: Experimental results for cold start users and all users in Flixster.

Method	Cold Start Users			All Users		
	RMSE	Coverage(%)	F-Measure	RMSE	Coverage(%)	F-Measure
Item-based	1.097	71.59	0.721	0.8938	94.27	0.852
CF Pearson	1.114	69.86	0.710	0.9132	90.37	0.833
MoleTrust	1.083	95.93	0.829	0.8997	95.46	0.856
Tidal Trust	1.106	96.11	0.826	0.8821	96.12	0.861
RandomWalk 1	1.009	65.23	0.697	0.9002	87.32	0.821
RandomWalk 6	1.087	80.21	0.763	0.9032	95.65	0.856
TrustWalker-orig	1.051	97.06	0.838	0.8522	99.89	0.880
TrustWalker	1.042	96.51	0.837	0.8413	99.63	0.881

As shown in the left part of Table 4.3, for cold start users in Epinions TrustWalker achieves lower error than all the other methods except for RandomWalk1. Note that RandomWalk1 has extremely low coverage (12%). In other words, RandomWalk1 can not compute a prediction for most cases, but for those that it can, RandomWalk1 produces high quality results, since it only uses the information provided by direct neighbors. Table 4.3 shows that both versions of TrustWalker outperform all other methods according to the combination of precision and coverage (F-measure). Notice that TidalTrust[36] achieves an RMSE which is only slightly higher than TrustWalker. However, TrustWalker's coverage is 13% more than that of TidalTrust, which makes TrustWalker better in terms of F-Measure.

The results on cold start users for Flixster are presented in Table 4.4. As shown in this table, TrustWalker outperforms existing methods in the Flixster data set, although by a smaller margin than on Epinions. We believe, this is mainly due to the fact that Flixster is generally denser than Epinions, and only using the ratings may provide enough information to perform high quality predictions.

The results for all users in Epinions are shown in right side of Table 4.3. We observe similar relative performance of all methods as for cold start users. It should be noted that all methods perform significantly better over all users since there is less information available for cold start users. TrustWalker outperforms all other methods in terms of F-Measure, although the gain for all users is less than the gain for cold start users. Notice that the errors of both TidalTrust and MoleTrust for all users are very close to TrustWalker, but TrustWalker clearly has a better coverage.

Table 4.4 presents the result for all users in Flixster. Again TrustWalker achieves better results against comparison partners. However the performance gain in Flixster is lower than the gain in

Epinions.

In summary, TrustWalker substantially improves the coverage of existing trust-based approaches while maintaining the same or even slightly better precision. This improvement is achieved by considering ratings for similar items as well as the exact target item. TrustWalker also clearly outperforms collaborative filtering (both user-based and item-based) in terms of coverage because of exploiting the trust-network in its random walks. Moreover, TrustWalker clearly outperforms collaborative filtering methods in terms of precision due to the restriction to ratings from highly trusted users. Since cold start users have only a few ratings, the improvement of coverage using trust-based approaches (specially TrustWalker) compared to collaborative filtering is much more for cold start users than for all users.

For example, in Epinions,TrustWalker's coverage is 70% for cold start users while collaborative filtering approaches have coverage of 16% and 21%. But in case of all users, TrustWalker's coverage is 93% while collaborative filtering approaches have coverage of 69%. TidalTrust and MoleTrust have coverage of 57% and 55% respectively for cold start users while their coverage is 82% and 81% in case of all users.

4.7.2 Experiments with LinkWalker

In this subsection, we report our experimental results for LinkWalker. For each user u, we randomly withhold one of his direct neighbors (v) in the trust network and ask LinkWalker to recommend top N users to u. If v is among the recommended users, then a hit has occurred. The evaluation metric we use is "accuracy" and is defined as the percentage of hits.

We repeat the experiments 10 times and take the average of accuracy over different runs for N=10, N=20, N=50, and N=100, where N is the number of recommended users. The threshold ϵ_1 is set to 0.005, and we set K=2000 for termination of the overall method, where K is the number of random walks to be performed before updating the list of top-N users.

The comparison partner we use in our experiments is the RWR method proposed in [89]. As already pointed out, RWR is a special case of LinkWalker where $\phi_{u,u_0,k} = 0.5$ and similarity if rating patterns among users is ignored ($\forall u, v : sim(u, v) = 1$). Note that if similarity of a source user u and other users can not be computed (due to not having enough items rated), then LinkWalker can perform no better than RWR for that user u. Therefore we also report results on a subset of users for whom we can compute the similarity with at least one direct neighbor. We call these users as "warm users". LinkWalker can actually make a difference compared to RWR for warm users since

it can exploit the rating patterns in the recommendation process.

Tables 4.5 and 4.6 present the accuracy of link prediction on the two real life data sets for all users and warm users for different settings of N.

Table 4.5: Accuracy of Link Prediction for LinkWalker and RWR in Epinions. The results are presented for N=10, 20, 50 and 100. Also, the accuracy of link prediction on warm users for who whom we can compute the similarity with at least one direct neighbor is presented. All accuracy values are percentages.

	Model	N=10	N=20	N=50	N=100
All Users	LinkWalker	11.1	16.36	24.12	30.24
	RWR	9.73	14.12	22.41	28.46
Warm Users	LinkWalker	21.98	28.21	37.65	43.31
	RWR	10.07	15.03	23.82	29.12

Table 4.6: Accuracy of Link Prediction for LinkWalker and RWR in Flixster. The results are presented for N=10, 20, 50 and 100. Also, the accuracy of link prediction on warm users for who whom we can compute the similarity with at least one direct neighbor is presented. All accuracy values are percentages.

	Model	N=10	N=20	N=50	N=100
All Users	LinkWalker	5.82	8.87	14.63	20.11
	RWR	5.09	7.94	13.01	18.58
Warm Users	LinkWalker	10.19	15.1	23.72	30.8
	RWR	5.33	8.23	13.95	19.27

We observe that LinkWalker obtains higher accuracy than RWR in both Epinions and Flixster. In particular, for warm users the accuracy gain of LinkWalker against RWR is very substantial, e.g., LinkWalker doubles the accuracy for N=10 on both data sets. The substantial gain difference between the results for all users and warm users is due to the fact that most users are not among warm users and therefore the results for these users are the same in LinkWalker and RWR. Note that 20% of the users in Epinions and 8% of the users in Flixster are warm users. Hence, if we only focus on warm users where LinkWalker is different from RWR, we can highlight the influence of using rating patterns in LinkWalker on the quality of recommendations.

We would like to mention that all the accuracy values are fairly low for both data sets and for all values of N. We believe that this is mainly due to the strict evaluation procedure and the sparsity

of the data sets. There are many users in the trust network, and many of them could be trustworthy for the source user. LinkWalker may recommend a relatively short list of users that does not include the withheld user, and the evaluation procedure does not consider this as a hit, but in reality the recommended list may actually be trustworthy for the source user. Moreover, it should be noted that both LinkWalker and RWR can preform recommendations for all users, and hence there is no need to compare the coverage of the comparison partners.

4.7.3 Experiments with Top-N Item Recommendation

In this subsection we report our experimental results on top-N item recommendation method proposed in this chapter. In our experiments we set $N = 100$ for the following reasons. Note that we performed experiments on different values of N (i.e. 10,20, and 50) and all them led to similar relative results.

As discussed in Chapter 2, collaborative filtering approaches use a neighborhood size of k and aggregate the items rated by these neighbors to recommend top-N items. Therefore, we report the results of user-based collaborative filtering for different values of k.

In the combined approach, we merge the result of collaborative filtering and trust-based approach. The number of neighbors considered in CF approaches is k_1, and the number of trusted neighbors being considered in trust-based approach is k_2. We performed experiments on different settings of different models and reported the results in the original paper [48]. In this section, we only report the results for the best setting of each model. Here is the list of comparison partners in our experiments:

- *CF-User*: User-based collaborative filtering approach extended for top-N recommendation.
- *CF-Item*: Item-based collaborative filtering approach extended for top-N recommendation.
- *TrustWalkerList*: This is the Random Walk approach. Note that *maxDepth=2* leads to the best results [48].
- *Trust-CF-RW* is combined method where the top-K trusted users are computed using random walks. Our experiments in [48] demonstrate that using random walk for computing the top-K trusted users leads to better results compared to using BFS. It should also be noted that we only present the results for a fixed k_1 and varying values of k_2. k_1 is fixed using the optimum value based on our experiments with *CF-User* ($k_1 = 70$ for Epinions and $k_1 = 40$ for Flixster). Again, for more detailed results, please refer to [48].

Evaluation Metrics

Typically, the leave-one-out method is used to evaluate recommendation systems [36, 74, 101]. In the leave-one-out method for predicting a single rating, we withhold a rating and try to predict it using the trust network and the remaining ratings. For top-N recommendation, we have to adjust the leave-one-out method. We withhold a user's rating on an item, and ask the recommender to recommend top-N items for this user. If the withheld item is among the N recommended items, then we say that a hit has occurred. We compute the *recall* as follows:

$$recall = \frac{\#hits}{L} \tag{4.26}$$

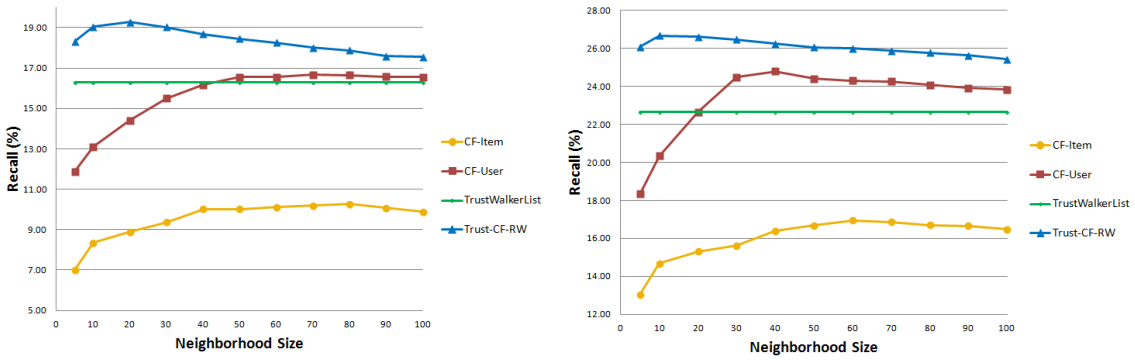
where L is the number of pairs of <user,item> being considered in the leave-one-out method. It should be noted that in the leave-one-out method we do not withhold all items of a user. Only items which have been highly rated will be withheld. In our experiments, we computed the maximum rating value expressed by each user and then we considered only items rated with the value of that maximum rating as items being withheld. It should be noted that recall has been also called *hit-ratio* in related works [30][56][57].

Experimental Results

In this section, we report our experimental results for all users and cold start users on both Epinions and Flixster.

Figure 4.1 shows the recall of comparison partners on all users. Note that neighborhood size is only a parameter for CF-based and combined approaches, so random walk based approaches produce constant curves. The figure shows that the optimum k for user-based collaborative filtering is 70 for Epinions leading to a recall of 16.66% and 40 for Flixster which leads to a recall of 24.78%. Also, the results on Flixster are generally better than the results on Epinions. There could be two possible explanation for this: The Epinions data set has relatively higher number of items compared to Flixster, which could lead to a more divers list of top-N recommendation. Also, items in Flixster are movies only while items in epinions are from different product categories which again can lead to more diversity in the recommended items and lower recall values.

The item-based collaborative filtering achieves poor results compared to the user-based CF, as shown in Figure 4.1. User-based CF for top-N recommendation relies on similar users which have similar rating patterns. These users are more likely to rate the withheld item. But item-based CF relies on items similar to items rated by the user. The withheld item is not necessarily among items



(a) Results on Epinions (b) Results on Flixster

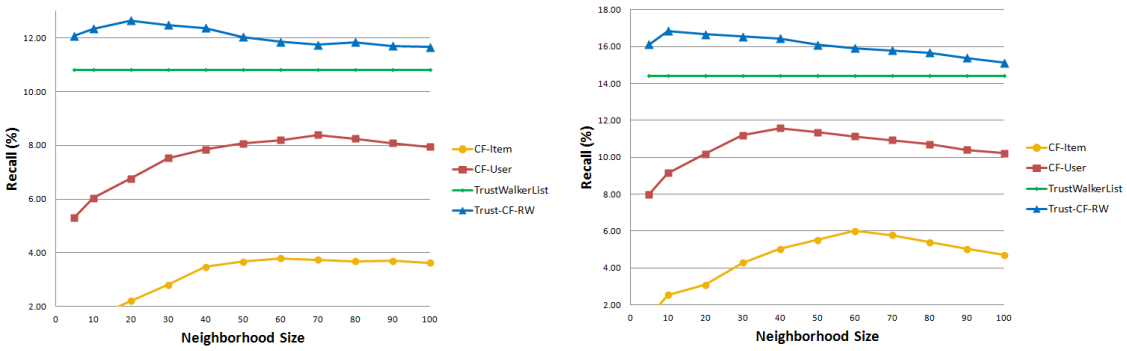
Figure 4.1: Experimental results with top-N recommendation on all users for different comparison partners. In each model, only the setting with the best results are shown.

similar to items rated by the user unless this withheld item is actually similar to other items rated by the user (which is not always true). So, the results of item-based CF for top-N recommendation are generally poor compared to user-based CF.

Intuitively, TrustWalkerList should perform better than CF-User since it is using extra information embedded in the social network, but the results do not show the better performance. We believe that this is due to the leave-one-out method used for evaluation. When a user u trusts another user v it does not necessarily mean that they rate the same items. It means that if they both rate an item, it is more likely for them to rate this item in a similar way. But using leave-one-out and recall as evaluation criteria, we are explicitly looking for the withheld items which may have not been rated by trusted users. Manually checking the results revealed that the recommended items are actually related to the interests of a user, but leave-one-out and recall are unable to capture this. On the other hand, in CF-User we select users who have rated items in a similar way to the source user and aggregate their ratings. If two users rate similar items (they have similar rating patterns), then it is more likely that a similar user also rates the withheld item. Hence, the recall of CF-User is better than that of TrustWalkerList.

The combined model, Trust-CF-RW, makes advantage of the properties of both collaborative filtering and social network-based models. Therefore, as shown in Figure 4.1 outperforms all the comparison partners. In Epinions, the best recall for Trust-CF-RW is 19.26% as opposed to 16.66% for CF-User, which leads to an improvement of 15.5% over CF-User. In Flixster, the recall improvement of the combined model over collaborative filtering based approach is 7.6%.

Cold start users are very important in recommender systems. As discussed before, approximately 50% of users in both data sets are cold start. Figure 4.2 present the experimental results on cold start users for both data sets. For cold start users, all approaches exploiting a social network outperform collaborative filtering based approaches. This is an evidence for confirmation of the claim that using a social network improves the quality of recommendation for cold start users. In Epinions, Trust-CF-RW improves the recall for cold start users by 50.1% which is substantial compared to the 15.5% improvement for all users. Similarly in Flixster, Trust-CF-RW improves the recall for cold start users by 45.6% compared to 7.6% recall improvement for all users. This great improvement for cold start users is mainly due to the fact that very few ratings are expressed by cold start users which makes it hard for collaborative filtering approaches to find similar users or items.



(a) Results on Epinions (b) Results on Flixster

Figure 4.2: Experimental results with top-N recommendation on cold start users for different comparison partners. In each model, only the setting with the best results are shown.

4.8 Conclusion

In this Chapter we proposed memory-based approaches for recommendation in social network. Our proposed method, TrustWalker, is a random walk based method that combines social network-based and similarity-based approaches for recommendation. TrustWalker performs random walks on the social rating network to compute the estimated rating for a specific user on the target item. In each walk, TrustWalker considers not only ratings on the target item, but also those of similar items, with probability increasing with increasing length of the walk. TrustWalker employs transitivity and social correlation for recommendation in social networks.

The idea of TrustWalker was employed to propose LinkWalker, a random walk based method

for top-N link prediction. Random walks in LinkWalker return recommended users rather than recommended ratings as done in TrustWalker. LinkWalker incorporates both transitivity and social selection into link prediction. Finally, we extended TrustWalker to address the top-N item recommendation. Random walks in this model return the items rather than ratings. Again, social correlation and transitivity have been taken into account for recommendation. Experimental results on Flixster and Epinions demonstrated that our proposed models outperform existing state-of-the-art models, in particular for cold start users.

In the next chapter, we propose a model-based approach for recommendation in social network extending matrix factorization.

Chapter 5

Regularized Matrix Factorization for Cold Start Recommendation in Social Rating Networks

5.1 Introduction

One of the major challenges in recommender systems is the cold start problem, which comes in two flavors: cold start users and cold start items. Cold start users are users for whom there is no or limited history of ratings, mainly because they have recently joined the system. For cold start users, there is not enough information to confidently determine the relevant neighbors in memory-based approaches or to train the user model, e.g., latent user factors in model-based recommendation. The cold start user problem has been recognized in the literature and has been addressed by various authors [47][119][74][69]. Cold start items are items that have been rated by very few users, often because the items have only been recently added to the system. Again, methods only based on ratings perform poorly for cold start items since there are few ratings of the cold start item in the neighborhood in memory-based methods or since the item model (e.g. item latent factors) cannot be computed accurately in model-based approaches. The cold start item problem is very important in practice, because new items need more promotion than old, well-known items and hence companies are interested in cold start items for their revenue improvements. Unlike the cold start user problem, the cold start item problem has received surprisingly little attention in the literature with the exception of [102] and [113]. To emphasize the importance of the cold start problems, it should be noted

that in the real life data sets used in our experiments (and in other existing data sets), approximately 50% of the users and items are cold start.

Social network-based recommenders are particularly helpful for cold start users and can make recommendations as long as the user is connected to a large enough component of the social network. As stated before, most existing works on social network-based recommendation are memory-based. Recently, the model-based approach for recommendation in social rating networks has been investigated [69][71]. These methods exploit the matrix factorization technique to learn latent factors for users and items from the observed ratings. Although transitivity of trust has been addressed in memory-based approaches [74][36][47][120], but memory-based approaches do not consider the propagation of trust. Since cold start users are more dependent on the social network compared to users with more ratings, the effect of using trust propagation gets more important for cold start users.

In this chapter, we introduce a comprehensive matrix factorization model, *SocialItemMF* [52], consisting of two main components to deal with cold start users (*SocialMF* [49] [50]) and cold start items (*ItemMF* [52]). Note that the problem definition we address in this section is the rating prediction problem defined in Chapter 2.

In SocialMF, we regularize the latent factors of a user by the latent factors of his direct neighbors in the social network. Regularization by direct neighbors represents social correlation in a matrix factorization model. Using this idea, latent factors of users indirectly connected in the social network will be dependent and hence trust propagation is incorporated into the model. In other words, SocialMF incorporates the effects of social influence [33], social selection [94] [44] and transitivity [44] [94] into matrix factorization to improve the recommendation accuracy. Note that matrix factorization based models, including SocialMF, also take into account the correlational influence (see Chapter 3) since the latent factors are learned using the observed ratings and users with similar rating behaviors will have similar latent factors.

Another advantage of SocialMF over existing models is the ability of learning the latent variable of a user even if no rating has been observed for the user. In many real life social rating networks a very large portion of users do not express any ratings, and they only participate in the social network. Hence, using only the observed ratings does not allow to learn the user factors.

To address cold start items, we propose ItemMF, in which the latent factor of an item is regularized by the latent factors of its neighbors in the item similarity graph. The item similarity graph is a network among items which contains information about items independent from their rating history. To construct the item similarity graph, we exploit publicly available information (such as genre and IMDB rating for movies) to define a similarity measure among items. The details of building an

item graph are described in Section 5.5. The assumption behind the use of an item graph is that neighboring items have similar features (e.g., similar genre and IMDB rating for movies) an therefore similar ratings. The item graph is particularly helpful for cold start items, and direct neighbors of an item can be exploited to learn the latent factors of a cold start item more confidently.

Finally, we combine the ideas employed in SocialMF and ItemMF to present a comprehensive model, SocialItemMF. In SocialItemMF, the latent factors for both users and items are regularized by the social network and item graph, which enables SocialItemMF to handle both cold start users and cold start items. Experimental results on two real life data sets from Epinions and Flixster show that SocialItemMF outperforms existing models. In particular for cold start users and items, the accuracy gain for SocialItemMF compared to the state-of-the-art methods is substantial.

The rest of this chapter is organized as follows: Some related work is discussed in Section 5.2. We introduce SocialMF in Section 5.3. Then, we present ItemMF and how it can be integrated with SocialMF to form SocialItemMF in Section 5.4. Section 5.5 describes how we build item similarity graphs in the real life data sets in our experiments. Our experimental results are reported in Section 5.6.

5.2 Related Work

Most related work such as matrix factorization [99], memory-based approaches for recommendation in social network (e.g., TidalTrust [36], MoleTrust [74], etc.) and model-based approaches for social network-based recommendation (e.g., STE [69] and Sorec [71]) have been discussed in Chapter 2. The social network-based approaches discussed deal with cold start users. As mentioned before, cold start items have not received many attentions from the literature. In this section, we review two works addressing cold start items.

The authors of [102] propose methods for cold start movie recommendation. They assume that actors of movies are surrogates for the movies and recommend movies to a user based on how similar the cast is to movies the user has already rated.

The proposed model in [113] combines Latent Dirichlet Allocation (LDA) [18] and matrix factorization for recommendation of scientific papers. An LDA model is learned from the content of the papers and the item factors are assumed to depend on the topic distribution of the item (paper). This model requires rich textual content to be available for items which is not the case in many applications.

5.3 The SocialMF Model

In this section, we present our approach to incorporate trust propagation into a matrix factorization model for recommendation in social networks. Note that the social network employed in the proposed model can come from different sources. It could be an explicit online social relation as those existing in Facebook, an email exchange network extracted from email logs, or even real life friendships. Basically any graph of users containing data about social engagement among users and carrying information independent from those embedded in the rating data can be used as the input social network for the proposed model. On the other hand, the similarity graph built based upon the similarity of rating patterns contain no information independent from the rating data itself and has no added value.

5.3.1 The Model

Due to social influence [33], the behavior of a user u is affected by his direct neighbors N_u. In other words, the latent factor of u is dependent on the latent factors of all his direct neighbors $v \in N_u$. We formulate this influence as follows:

$$\widehat{U}_u = \frac{\sum_{v \in N_u} T_{u,v} U_v}{\sum_{v \in N_u} T_{u,v}} = \frac{\sum_{v \in N_u} T_{u,v} U_v}{|N_u|} \tag{5.1}$$

where $\widehat{U}_u$ is the estimated latent factor of u given the latent factors of his direct neighbors. Since the social networks we are working with are all binary social networks, all none-zero values of $T_{u,v}$ are 1. We normalize each row of the trust matrix so that $\sum_{v=1}^{N} T_{u,v} = 1$. Now, we have:

$$\widehat{U}_u = \sum_{v \in N_u} T_{u,v} U_v \tag{5.2}$$

The above equation indicates that the estimate of the latent factor of a user is the weighted average of the latent factors of his direct neighbors. The estimated latent factor of user u can be inferred as:

$$\begin{pmatrix} \widehat{U}_{u,1} \\ \widehat{U}_{u,2} \\ ... \\ \widehat{U}_{u,K} \end{pmatrix} = \begin{pmatrix} U_{1,1} & U_{2,1} & ... & U_{N,1} \\ U_{1,2} & U_{2,2} & ... & U_{N,2} \\ ... & .. & ... & ... \\ U_{1,K} & U_{2,K} & ... & U_{N,K} \end{pmatrix} \begin{pmatrix} T_{u,1} \\ T_{u,2} \\ ... \\ T_{u,N} \end{pmatrix} \tag{5.3}$$

Note that taking the social network into account does not change the equation for the conditional distribution of the observed ratings. It only affects the user latent factors. So the conditional probability of observed rating is the same as the conditional probability in the original matrix factorization approach:

$$p(R|U,V,\sigma_R^2) = \prod_{u=1}^{N}\prod_{i=1}^{M}\Big[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_r^2\Big)\Big]^{I_{u,i}^R} \tag{5.4}$$

For the user latent variables, we have two factors: The zero-mean Gaussian prior to avoid overfitting, and the conditional distribution of user latent factors given the latent factors of his direct neighbors. Therefore,

$$\begin{aligned} p(U|T,\sigma_U^2,\sigma_T^2) &\propto p(U|\sigma_U^2)\times p(U|T,\sigma_T^2) \\ &= \prod_{u=1}^{N}\mathcal{N}\Big(U_u|0,\sigma_U^2\mathbf{I}\Big)\times\prod_{u=1}^{N}\mathcal{N}\Big(U_u|\sum_{v\in N_u}T_{u,v}U_v,\sigma_T^2\mathbf{I}\Big) \end{aligned} \tag{5.5}$$

The above distribution is a normal distribution which is a product of two different normal distributions to keep the user latent factors both small and close to the factors of their direct neighbors.

Through a Bayesian inference, we have the following equation for the posterior probability of latent factors given the rating and social trust matrices:

$$\begin{aligned} p(U,V|R,T,\sigma_R^2,\sigma_T^2,\sigma_U^2,\sigma_V^2) &\propto p(R|U,V,\sigma_R^2)p(U|T,\sigma_U^2,\sigma_T^2)p(V|\sigma_V^2) \\ &= \prod_{u=1}^{N}\prod_{i=1}^{M}\Big[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_r^2\Big)\Big]^{I_{u,i}^R}\times\prod_{u=1}^{N}\mathcal{N}\Big(U_u|\sum_{v\in N_u}T_{u,v}U_v,\sigma_T^2\mathbf{I}\Big) \\ &\quad\times\prod_{u=1}^{N}\mathcal{N}\Big(U_u|0,\sigma_U^2\mathbf{I}\Big)\times\prod_{i=1}^{M}\mathcal{N}\Big(V_i|0,\sigma_V^2\mathbf{I}\Big) \end{aligned} \tag{5.6}$$

The graphical model corresponding Equation (5.6) is shown in Figure 5.1. Note that the trust matrix in the above equation is not explicitly shown in the figure. However, the edges among the latent factors of users are representatives of the trust network among users and the degree of trust of user u on user v is $T_{u,v}$.

The log of the posterior probability can be computed as follows:

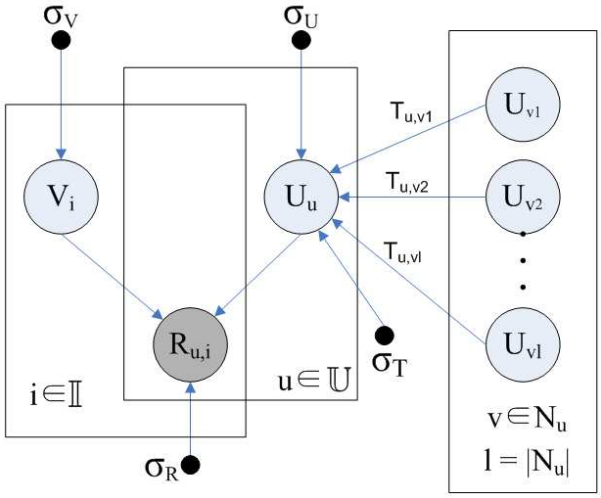


Figure 5.1: SocialMF: Proposed Graphical Model to consider the social network in the factorization of user-item rating matrix.

$$
\begin{aligned}
&lnp(U,V|R,T,\sigma_R^2,\sigma_T^2,\sigma_U^2,\sigma_V^2) = \\
&\quad -\frac{1}{2\sigma_R^2}\sum_{u=1}^{N}\sum_{i=1}^{M} I_{u,i}^R (R_{u,i} - g(U_u^T V_i))^2 - \frac{1}{2\sigma_U^2}\sum_{u=1}^{N} U_u^T U_u - \frac{1}{2\sigma_V^2}\sum_{i=1}^{M} V_i^T V_i \\
&\quad -\frac{1}{2\sigma_T^2}\sum_{u=1}^{N}\Big((U_u - \sum_{v\in N_u} T_{u,v}U_v)^T (U_u - \sum_{v\in N_u} T_{u,v}U_v)\Big) - \frac{1}{2}(\sum_{u=1}^{N}\sum_{i=1}^{M} I_{u,i}^R)\ln\sigma_R^2 \\
&\quad -\frac{1}{2}((N\times K)\ln\sigma_U^2 + (M\times K)\ln\sigma_V^2 + (N\times K)\ln\sigma_T^2) + \mathcal{C} \qquad (5.7)
\end{aligned}
$$

Keeping the parameters (observation noise variance and prior variance) fixed, maximizing the log-posterior over latent factors of users and items is equivalent to minimizing the following objective function, which is a sum of squared errors with quadratic regularization terms:

$$
\begin{aligned}
\mathcal{L}(R,T,U,V) = &\frac{1}{2}\sum_{u=1}^{N}\sum_{i=1}^{M} I_{u,i}^R (R_{u,i} - g(U_u^T V_i))^2 + \frac{\lambda_U}{2}\sum_{u=1}^{N} U_u^T U_u + \frac{\lambda_V}{2}\sum_{i=1}^{M} V_i^T V_i \\
&+ \frac{\lambda_T}{2}\sum_{u=1}^{N}\Big((U_u - \sum_{v\in N_u} T_{u,v}U_v)^T (U_u - \sum_{v\in N_u} T_{u,v}U_v)\Big) \qquad (5.8)
\end{aligned}
$$

In the above equation, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, and $\lambda_T = \sigma_R^2/\sigma_T^2$. We can find a local minimum of the objective function in Equation (5.8) by performing stochastic gradient descent on U_u and V_i for all users u and all items i.

5.3.2 Discussion

In this section, we discuss some desirable properties of SocialMF and compare it against the closely related STE model [69].

The SocialMF model addresses the transitivity of trust in social networks. In other words, our model takes the trust propagation into account. According to the graphical model, the latent factor of any user is dependent on the latent factors of his direct neighbors. Recursively, the latent factor of each direct neighbor is dependent on the latent factor of his direct neighbors. This effect is modeled in the conditional distributions by considering the latent factor of a user being a normal distribution around the average of the latent factors of his neighbors. On the other hand, the STE model [69] does not support trust propagation and they list trust propagation as future work.

In the baseline MF approach [99] and the STE model [69], the latent factors are being learned based only on the observed ratings. However, in real life social rating networks, a huge portion of users have expressed no ratings and they participate only in the social network. So their factors can not be learned based on their observed ratings. However, our model can handle these users very well. The SocialMF model learns to tune the latent factors of these users close to their neighbors. So, despite not having any expressed ratings, the latent factors of these users will be learned to be close to their neighbors. Basically, the social trust relations among users is an observed dependency among the latent factors of users. It should be noted that since evaluating the learned factors is typically based on the withheld observed ratings, we are currently not able to evaluate the latent factors learned for users with no expressed ratings.

5.3.3 Complexity analysis

The main cost in learning the parameters is computing $\mathcal{L}$ and its gradients against latent factors of users and items. Assuming the average number of ratings per user is $\overline{r}$, and the average number of direct neighbors per user is $\overline{t}$, the complexity of evaluation of $\mathcal{L}$ is $O(N\overline{r}K + N\overline{t}K)$. Since both the rating matrix R and trust matrix T are very sparse, $\overline{t}$ and $\overline{r}$ are relatively small. So the computation of the objective function $\mathcal{L}$ is very fast and linear with respect to the number of users in the social rating network. The computational complexity of computing the gradients is $O(N\overline{r}K + N\overline{t}^2K)$

which is linear with respect to the number of users in the social rating network. Note that the cost of computing the gradient in STE [69] is $O(N\overline{r}\overline{t}^2K)$. So SocialMF is $\frac{\overline{r}\overline{t}^2}{\overline{r}+\overline{t}^2}$ times faster than STE in computing the gradient in each iteration of parameter learning process.

For each rating estimation, the model proposed in [69] needs to take the average of estimated ratings for direct neighbors which makes it slower in prediction compared to SocialMF proposed in this dissertation.

5.4 The ItemMF and the SocialItemMF model

In this section, we present our proposed model to extend matrix factorization for addressing cold start items. We first describe ItemMF, a matrix factorization based model in which latent factors of every item is regularized by the latent factors of its neighbors in the item similarity graph. Then we combine ItemMF with SocialMF to present SocialItemMF, a model to address both cold start users and cold start items.

An item similarity graph S (or item graph for short) is a graph among items where items with high similarity are connected by edges. The edge weights denote the similarity values between items. To construct the similarity graph, we use information independent from the rating history of items. We can exploit the public domain data available for items in each specific data set to compute the similarity of items. For instance, in a movie data set (Flixster), we can use the IMDB rating of the movie together with the movie genre to compute a similarity measure. In a product review data set (Epinions), we can use the product name and the product category to compute item similarities. The computed similarities are pruned to eliminate low similarity pairs of items from the item graph. The details of item graph construction for different data sets in our experiments are discussed in Section 5.5.

The item graph is represented by its adjacency matrix $S = [S_{i,j}]_{M\times M}$. Non-zero cells $S_{i,j}$ in S denote the existence of an edge $< i, j >$ in the item graph with weight $S_{i,j}$, the value of similarity between u and v. Obviously, S is an undirected graph.

5.4.1 The ItemMF model

Properties of an item are similar to the properties of its direct neighbors, N_i, in the item graph S. In other words, the latent factor of an item i is dependent on the latent factors of all its neighbors $j \in N_i$. Similar to SocialMF, we formulate this correlation as follows:

$$\widehat{V}_i = \frac{\sum_{j \in N_i} S_{i,j} V_j}{\sum_{j \in N_i} S_{i,j}} \tag{5.9}$$

where $\widehat{V}_j$ is the estimate latent factor of the item i given the latent factors of its direct neighbor in the item graph. If we normalize each row of the adjacency matrix S so that $\sum_{j=1}^{M} S_{i,j} = 1$, we will have:

$$\widehat{V}_i = \sum_{j \in N_i} S_{i,j} V_j \tag{5.10}$$

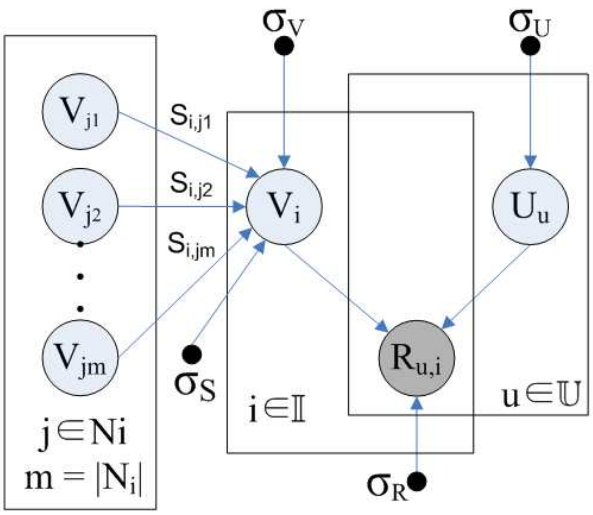


Figure 5.2: The graphical model for ItemMF

Figure 5.2 presents the graphical model corresponding to ItemMF. Similar to SocialMF, the conditional probability of the observed ratings can be computed using Equation (5.4). Through a Bayesian inference, the posterior likelihood of the latent factors given the ratings and the item graph is computed as follows:

$$p(U,V|R,S,\sigma_R^2,\sigma_S^2,\sigma_U^2,\sigma_V^2) \propto p(R|U,V,\sigma_R^2)p(U|\sigma_U^2)p(V|S,\sigma_V^2,\sigma_S^2)$$
$$= \prod_{u=1}^{N}\prod_{i=1}^{M}\Big[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_R^2\Big)\Big]^{I_{u,i}^R} \times \prod_{i=1}^{M}\mathcal{N}\Big(V_i|\sum_{j\in N_i} S_{i,j}V_j,\sigma_S^2\mathbf{I}\Big)$$
$$\times \prod_{u=1}^{N}\mathcal{N}\Big(U_u|0,\sigma_U^2\mathbf{I}\Big) \times \prod_{i=1}^{M}\mathcal{N}\Big(V_i|0,\sigma_V^2\mathbf{I}\Big) \quad (5.11)$$

Similar to the procedure in SocialMF, maximizing the log-posterior over latent factors of users and items is equivalent to minimizing the following objective function, which is again a sum of squared errors with quadratic regularization terms:

$$\mathcal{L}(R,S,U,V) = \frac{1}{2}\sum_{u=1}^{N}\sum_{i=1}^{M} I_{u,i}^R(R_{u,i} - g(U_u^T V_i))^2 + \frac{\lambda_U}{2}\sum_{u=1}^{N} U_u^T U_u + \frac{\lambda_V}{2}\sum_{i=1}^{M} V_i^T V_i$$
$$+ \frac{\lambda_S}{2}\sum_{i=1}^{M}\Big((V_i - \sum_{j\in N_i} S_{i,j}V_j)^T(V_i - \sum_{j\in N_i} S_{i,j}V_j)\Big) \quad (5.12)$$

In the above equation, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, and $\lambda_S = \sigma_R^2/\sigma_S^2$.

Using stochastic gradient descent on U_u and V_i, we compute the optimized latent factor with respect to the objective function described above.

5.4.2 The SocialItemMF model

SocialMF employs a social network to address cold start users. ItemMF uses an item graph to handle the cold start items issue. In this section we combine the ideas presented in SocialMF and ItemMF to present SocialItemMF, a comprehensive model that is able to handle both cold start users and cold start items. In SocialItemMF, latent factors of users are regularized by the latent factors of their direct neighbors in the social network. Also, latent factors of items are regularized by those of the direct neighbors in the item graph. In other words, the user and item latent factors are estimated as follows:

$$\widehat{U}_u = \sum_{v\in N_u} T_{u,v}U_v \quad (5.13)$$

$$\widehat{V}_i = \sum_{j\in N_i} S_{i,j}V_j \quad (5.14)$$

Note that both $T_{u,v}$ and $S_{i,j}$ are normalized so that $\sum_{i=1}^{N} T_{u,v} = 1$ and $\sum_{j=1}^{M} S_{i,j} = 1$. Figure 5.3 illustrates the graphical model corresponding the SocialItemMF model. Again, the conditional probability of the rating can be computed as follows:

$$p(R|U,V,\sigma_R^2) = \prod_{u=1}^{N}\prod_{i=1}^{M}\Big[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_r^2\Big)\Big]^{I_{u,i}^R} \tag{5.15}$$

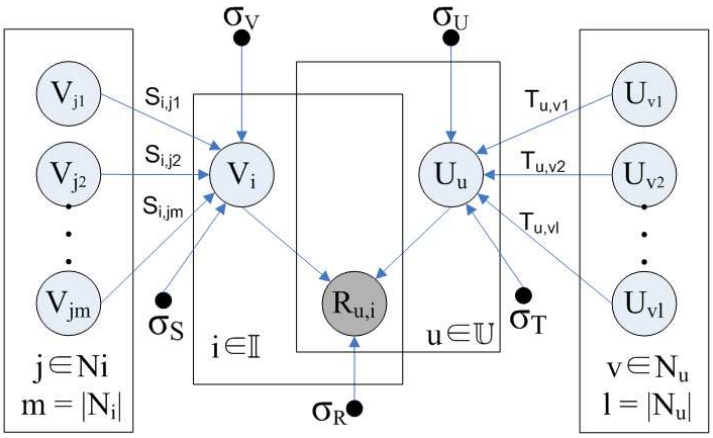


Figure 5.3: The graphical model for SocialItemMF

The posterior likelihood of the latent factors given the observed rating, the social network and the item similarity graph can be computed using a Bayesian inference as follows:

$$\begin{aligned}
p(U,V|R,T,S,\sigma_R^2,\sigma_T^2,\sigma_S^2,\sigma_U^2,\sigma_V^2) &\propto p(R|U,V,\sigma_R^2)p(U|T,\sigma_U^2,\sigma_T^2)p(V|S,\sigma_V^2,\sigma_S^2) \\
&= \prod_{u=1}^{N}\prod_{i=1}^{M}\Big[\mathcal{N}\Big(R_{u,i}|g(U_u^T V_i),\sigma_R^2\Big)\Big]^{I_{u,i}^R} \\
&\times \prod_{u=1}^{N}\mathcal{N}\Big(U_u|\sum_{v\in N_u} T_{u,v}U_v,\sigma_T^2\mathbf{I}\Big) \times \prod_{i=1}^{M}\mathcal{N}\Big(V_i|\sum_{j\in N_i} S_{i,j}V_j,\sigma_S^2\mathbf{I}\Big) \\
&\times \prod_{u=1}^{N}\mathcal{N}\Big(U_u|0,\sigma_U^2\mathbf{I}\Big) \times \prod_{i=1}^{M}\mathcal{N}\Big(V_i|0,\sigma_V^2\mathbf{I}\Big)
\end{aligned} \tag{5.16}$$

Maximizing the log-posterior is equivalent to minimizing the following objective function, which is a sum of squared errors with quadratic regularization terms for both user and item latent factors:

$$\mathcal{L}(R,T,S,U,V) = \frac{1}{2}\sum_{u=1}^{N}\sum_{i=1}^{M} I^R_{u,i}(R_{u,i} - g(U_u^T V_i))^2 + \frac{\lambda_U}{2}\sum_{u=1}^{N} U_u^T U_u + \frac{\lambda_V}{2}\sum_{i=1}^{M} V_i^T V_i + \frac{\lambda_T}{2}\sum_{u=1}^{N}\Big((U_u - \sum_{v\in N_u} T_{u,v}U_v)^T(U_u - \sum_{v\in N_u} T_{u,v}U_v)\Big) + \frac{\lambda_S}{2}\sum_{i=1}^{M}\Big((V_i - \sum_{j\in N_i} S_{i,j}V_j)^T(V_i - \sum_{j\in N_i} S_{i,j}V_j)\Big) \quad (5.17)$$

In the above Equation, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, $\lambda_S = \sigma_R^2/\sigma_S^2$, and $\lambda_T = \sigma_R^2/\sigma_T^2$.

We use stochastic gradient descent to optimize the latent factors with respect to the objective function. Gradients of the objective function $\mathcal{L}$ with respect to U_u and V_i is computed as follows:

$$\frac{\partial \mathcal{L}}{\partial U_u} = \sum_{i=1}^{M} I^R_{u,i} V_i g'(U_u^T V_i)(g(U_u^T V_i) - R_{u,i}) + \lambda_U U_u + \lambda_T(U_u - \sum_{v\in N_u} T_{u,v}U_v)) - \lambda_T \sum_{\{v|u\in N_v\}} T_{v,u}\Big(U_v - \sum_{w\in N_v} T_{v,w}U_w\Big) \quad (5.18)$$

$$\frac{\partial \mathcal{L}}{\partial V_i} = \sum_{u=1}^{N} I^R_{u,i} U_v g'(U_u^T V_i)(g(U_u^T V_i) - R_{u,i}) + \lambda_V V_i + \lambda_S(V_i - \sum_{j\in N_i} S_{i,j}V_j)) - \lambda_S \sum_{\{j|i\in N_j\}} S_{j,i}\Big(V_j - \sum_{l\in N_j} S_{j,l}V_l\Big) \quad (5.19)$$

where $g'(x)$ is the derivative of logistic function and is equal to $g'(x) = e^{-x}/(1+e^{-x})^2$. To reduce the model complexity, we set $\lambda_U = \lambda_V$ in all our experiments. The initial values of U and V are samples from normal noises with zero mean. In each iteration, U and V are updated based on the latent variables from the previous iteration.

Note that in both ItemMF and SocialItemMF, latent factors for an item can be learned even if ratings are not provided for items. This is particularly helpful for cold start items, or items in reviewing websites that despite having some reviews not explicit rating has been expressed for the item.

5.4.3 Complexity analysis

Similar to the discussion in Section 5.3.3, the main computational cost in each iteration of learning in ItemMF and SocialItemMF is computation of the gradients. Table 5.1 compares the computational

costs of the proposed models in this chapter and the stat-of-the-art model STE [69] from literature.

Table 5.1: Comparison of computational complexity for computing the gradients in different models

Model	Computational Cost
STE	$O(N\overline{r}\overline{t}^2K)$
SocialMF	$O(N\overline{r}K + N\overline{t}^2K)$
ItemMF	$O(N\overline{r}K + M\overline{s}^2K)$
SocialItemMF	$O(N\overline{r}K + N\overline{t}^2K + M\overline{s}^2K)$

Here, $\overline{r}$ is the average number of ratings per user, $\overline{t}$ is the average number of direct neighbors per user in the social network, and $\overline{s}$ is the average number of direct neighbors per item in the item similarity graph. Since the matrices corresponding the ratings, social network, and the item graph are very sparse, $\overline{r}$, $\overline{t}$, and $\overline{s}$ are relatively small. Therefore, computational complexity of computation of the objective function $\mathcal{L}$ and its derivatives is very fast and linear with respect to the number of users and items.

5.5 Data sets and Construction of Item Graphs

In our experiments we used two real life data sets from Epinions.com and Flixster.com. For a detailed description of the data sets, please refer to Chapter 4. In the following, we describe how we build an item similarity graph for each data set.

To construct the item (similarity) graphs, we use information independent from the rating history of items. We exploit the public domain data available for both Flixster and Epinions to compute the similarity of item pairs. After computation of similarities for all possible pairs, we prune the graph by setting similarity in a way that the average number of neighbors for items is close to the average number of neighbors for users to have item graphs with density similar to the social network among users. In the following, we describe the details of construction of item graphs in the Flixster and the Epinions data sets.

Item Graph in Epinions

In Epinions, product name and product category are publicly available for all products and can be easily collected. Fortunately, the version of the Epinions data set we used in our experiments[1] provides both product name and product category for all items. Product names can contain more than one word. After stemming and removing stop words from product names, we used the *Jaccard coefficient* between the bag-of-words representing items as the similarity measure. Note that only items from the same category are considered to be similar and items from different categories are always considered to have zero similarity.

$$sim(i,j) = I(cat_i, cat_j) \in \frac{|name_i \cap name_j|}{|name_i \cup name_j|} \tag{5.20}$$

In the above equation, $I(cat_i, cat_j)$ is an indicator that takes the value 1 if items i and j are from the same category and 0 otherwise. $name_i$ is the set of keywords representing the product name of item i.

Item Graph in Flixster

The items in Flixster are all movies. Therefore, we used the publicly available data from IMDB.com to extract the genre and IMDB ratings of movies. IMDB rating for a movie i, $imdb_i$, is a real number in the range [0,1]. A movie can have more than one genre, and we denote the set of genres for movie i by $genres_i$. Similarity of movies according their IMDB ratings is computed using a simple distance measure and similarity of items according to their genre is computed using the Jaccard coefficient. We compute the product of these two similarity metrics to obtain the overall movie similarity as follows:

$$sim(i,j) = \frac{10 - |imdb_i - imdb_j|}{10} \times \frac{|genres_i \cap genres_j|}{|genres_i \cup genres_j|} \tag{5.21}$$

It should be noted that the information exploited to construct the item graphs in both data sets are from public domains. We argue that similar public domain information is available for many other recommendation data sets. As long as the information used in the item graph is orthogonal to the information in the rating matrix, the item graph can help to improve the quality of recommendation for cold start items.

[1] http://alchemy.cs.washington.edu/data/epinions/

5.6 Experiments

In this section, we report our experimental results including those for existing methods. In particular, we present the results for cold start users and cold start items.

5.6.1 Experimental Setup

We perform 5-fold cross validation in our experiments. In each fold we have 80% of data as the training set and the remaining 20% as the test data. The evaluation metric we use in our experiments is RMSE (see Section 2.3 for details). To evaluate the performance of our method we consider three comparison partners:

- *CF*: This is the well-known user-based collaborative filtering method which is a memory-based approach.
- *BaseMF*: This method is the baseline matrix factorization approach proposed in [99], which does not take the social network into account.
- *STE*: This is the model proposed in [69], which takes into account the social network in a way different from SocialMF. We set $\alpha = 0.4$ for STE in our experiments which is the optimum value according to the results of experiments in [69].
- *MA*: This is the model proposed by Ma et al. in [70][2].
- *SocialMF*: The proposed model for social network-based recommendation that extends matrix factorization by regularizing latent user factors. SocialMF is particularly designed to help with cold start users.
- *ItemMF*: The proposed model to address cold start items that extends matrix factorization by regularizing latent item factors.
- *SocialItemMF*: The comprehensive model that combines SocialMF and ItemMF by regularizing both user and item latent factors.

In all our experiments, we set $\lambda_U = \lambda_V = 0.1$[3].

[2] As in our paper [49], we are using STE as the main comparison partner in this section. The experimental evaluation of MA [70] was performed only in 2012 to make the comparison more comprehensive.

[3] These values have been experimentally tuned.

5.6.2 Experimental Results

Table 5.2 reports the RMSE values of all comparison partners on the Epinions data set. The parameter λ_T is set to 5 for experiments on Epinions. Table 5.2 shows that SocialMF outperforms the existing methods. Note that since collaborative filtering has no latent factors, there is no dimensionality K associated with it and hence the result for different values of K are the same.

SocialItemMF improves the RMSE of STE by 7.9% for K=5 and by 7.1% for K=10. To show how substantial our gain is, note that the gain of STE over the baseline MF method is 2.5% and the gain of SocialMF over STE is more than 3 times that gain. As another evidence for substantiality of these RMSE reductions, note that in the Netflix prize competition[4], there was a $1 Million reward for a reduction of the RMSE by 10%. Also, SocialItemMF achieves a light gain of 1.3% over SocialMF, demonstrating that regularizing item factors further improves the quality of recommendation achieved by regularizing the user latent factors in SocialMF. Note that ItemMF performs worse than SocialMF, meaning that the social network is much more important than the item graph for rating prediction in Epinions.

Table 5.2: RMSE values for comparison partners on Epinions with different settings of dimensionality K.

Method	K=5	K=10
CF	1.180	1.180
BaseMF	1.175	1.195
STE	1.145	1.150
MA	1.113	1.120
SocialMF	1.075	1.085
ItemMF	1.147	1.159
SocialItemMF	1.061	1.074

RMSE values for Flixster are presented in Table 5.3. Again, SocialItemMF clearly outperforms the existing methods. In Flixster, the improvement of the RMSE for SocialItemMF over STE is 8.4% which is more than 5 times of the gain of STE over baseline MF (1.5%).

It should be noted that the results for Flixster are generally better than the results for Epinions for all methods. This may be because of the fact that the items in Epinions are from multiple categories such as DVD players, cameras, printers, laptops, etc., while the items in Flixster are all movies,

[4]http://www.netflixprize.com

which simplifies the recommendation task. Another explanation for the better results on Flixster could be that Flixster is a denser data set since there are more social relations and ratings per user in Flixster compared to Epinions.

Table 5.3: RMSE values for comparison partners on Flixster with different settings of dimensionality K.

Method	K=5	K=10
CF	0.911	0.911
BaseMF	0.878	0.863
STE	0.864	0.852
MA	0.842	0.831
SocialMF	0.821	0.815
ItemMF	0.853	0.842
SocialItemMF	0.795	0.786

Intuitively, increasing K should add more flexibility to the model and hence should improve the results. However, comparing results of Tables 5.2 and 5.3 for different values of K shows that increasing K in Epinions did not improve the results, while increasing K in Flixster improved the results. We believe that these results for Epinions are due to the fact that the Epinions data set is smaller than Flixster and increasing K means more parameters in the model which leads to overfitting. Flixster, on the other hand, is a huge data set, and increasing K to 10 does not lead to overfitting.

5.6.3 Impact of λ_T and λ_S on results

Parameter λ_T controls the influence of the social network on the behavior of users. Larger values of λ_T in the objective function of Equation (5.17) indicate more impact of the social network on the behavior of users. Very small values of λ_T make our model close to the baseline MF approach. However, very large values of λ_T lead to a model in which having latent factors close to those of direct neighbors dominates having a lower squared error in the training phase. Similarly, λ_S controls the influence of the item graph on the behavior of items.

Figure 5.4 compares the RMSE of the SocialMF model for a range of values for λ_T in both data sets. As shown in this figure, SocialMF achieves its best results on Epinions for $\lambda_T = 5$, and $\lambda_T = 1$ for Flixster. Note that a fairly broad range of λ_T values is close to optimal and the same value of 2

could be chosen in both data sets as a near optimal λ_T value.

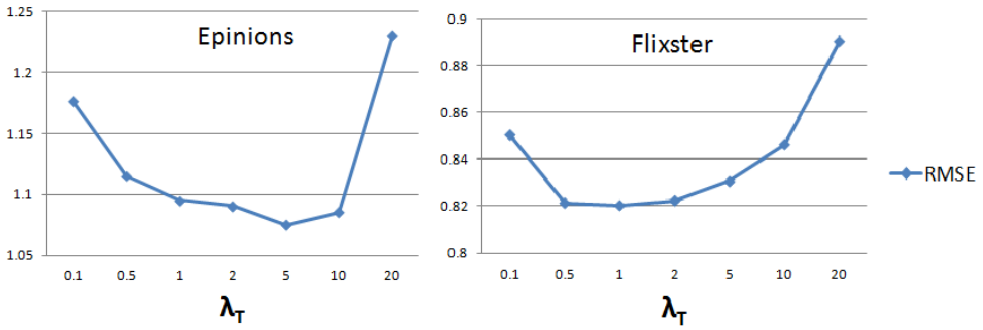


Figure 5.4: Impact of different values of λ_T on the performance of prediction in SocialMF for Epinions and Flixster.

Figure 5.5 compares the RMSE of the ItemMF model for different a range of values for λ_S in both data sets. As shown in these Figures, ItemMF has its best results on Epinions for $\lambda_S = 7$, and $\lambda_S = 2$ for Flixster. Again, the optimal values are similar in both datasets, and a value in the range [2,5] achieves near optimal performance in both data sets.

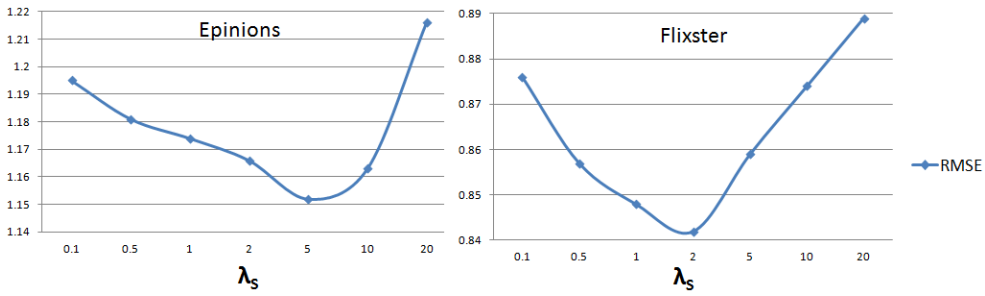


Figure 5.5: Impact of different values of λ_S on the performance of prediction in ItemMF for Epinions and Flixster.

In all experimental results, the values of λ_S and λ_T are set to their optimum value according to the sensitivity analysis shown in Figures 5.4 and 5.5.

5.6.4 Performance on cold start users and items

The main goal of using social networks and item graph in a recommender system is to improve the quality of recommendation, in particular for cold start users and items. We consider users who have expressed less than 5 ratings as cold start users [75][47] and items with less than 5 ratings as cold start items. Note that in both Flixster and Epinions more than 50% of users are cold start users[5] and 48% of items in Flixster and 80% of items in Epinions are cold start items. Hence accuracy of any recommendation algorithm for cold start users and items becomes very important.

Table 5.4 shows that for cold start user, proposed SocialItemMF outperforms all other methods. The improvement of the RMSE for cold start users compared to STE is 13.1% for Epinions and 12.1% for Flixster. The gain for cold start users is more than the gain for all users which we discussed in previous subsection. This implies that SocialItemMF handles cold start users better than STE. We believe this is mainly due to the consideration of trust propagation in our model. Note that SocialMF also achieves a substantial RMSE gain against the STE model, although it is less significant than that of SocialItemMF. Not surprisingly, ItemMF does not perform very well for cold start users and only outperforms the baselineMF model.

Table 5.4: RMSE values on cold start users.

Method	Epinions	Flixster
CF	1.361	1.228
BaseMF	1.352	1.213
STE	1.295	1.152
MA	1.224	1.101
SocialMF	1.159	1.057
ItemMF	1.327	1.186
SocialItemMF	1.145	1.028

Table 5.5 compares the RMSE values of the comparison partners for cold start items in both data sets. According to this Table, the improvement of RMSE for cold start items for SocialItemMF compared to STE is 20.6% in Epinions and 47.9% in Flixster. The RMSE gain for cold start items is very substantial compared to that of all items, implying that SocialItemMF is particularly handling cold start items better than existing models. Also, compared to the RMSE gain for cold start users,

[5] In Flixster, we do not take into account the users with no ratings in this statistics.

the improvement for cold start items is much more substantial than the improvement for cold start users, confirming the importance of item graphs for cold start items.

Table 5.5: RMSE values on cold start items.

Method	Epinions	Flixster
CF	1.408	1.483
BaseMF	1.392	1.471
STE	1.351	1.403
MA	1.275	1.361
SocialMF	1.214	1.328
ItemMF	1.175	1.083
SocialItemMF	1.112	0.948

Note that the RMSE gain for ItemMF is more substantial in Flixter than in Epinions. We believe this is mainly due to the nature of the item graphs for these two data sets. In Epinions, the item similarities are computed based on product names and categories, while the item similarities in Flixster are computed based on genres and IMDB ratings of the movies. The IMDB rating is clearly more indicative of the quality of an item than the product name. Therefore, similar items in Flixster are more likely to have similar quality and rating, leading to a more informative item graph.

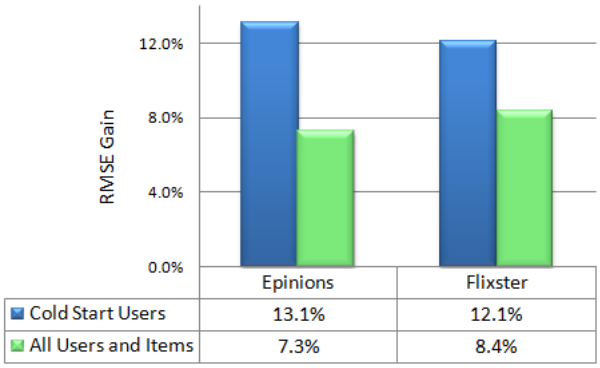
Figure 5.6 summarizes the improvement of RMSE in SocialItemMF over STE for cold start users and items.

5.6.5 Analysis of learning runtime

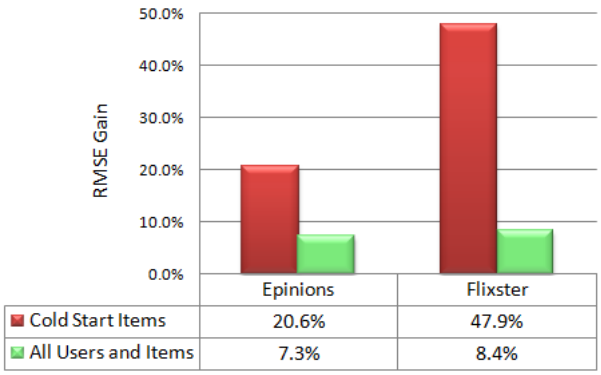
In section 4 and 5, we analyzed the runtime complexity, i.e. the complexity of computing the gradients of the objective function $\mathcal{L}$ in our proposed models (SocialMF, ItemMF, and SocialItemMF) and STE. In this section we compare the actual runtimes, conducting experiments on a Core2 Duo 2.16 GHz machine with Windows XP and 2 GB of memory.

Table 5.6 compares the actual runtime of a single iteration of the training phase for the comparison partners.

Note that according to our discussion in the previous sections, SocialMF should theoretically be $\frac{\bar{r}\bar{t}^2}{\bar{r}+\bar{t}^2}$ times faster than STE. Since the Flixster data set is denser than the Epinions data set, the improvement of runtime efficiency for Flixster is stronger than for Epinions.



(a) RMSE gain for cold start users



(b) RMSE gain for cold start items

Figure 5.6: Comparison of RMSE gain of SocialItemMF over STE for cold start users and cold start items

It should also be noted that the average number of iterations for SocialMF and ItemMF to converge is around 700, while the number of iterations for SocialItemMF is close to 750 and the number of iterations for STE is around 550. Table 5.7 shows the total time required to learn the parameters for each model. As shown in this Table, the STE model is much slower than SocialMF, ItemMF and SocialItemMF. The training phase of SocialMF is 40 times faster than that of the STE model for Flixster and more than 7 times faster for Epinions.

5.7 Conclusions

In this Chapter, we proposed a comprehensive matrix factorization based model, SocialItemMF, consisting of two main components to deal with cold start users (SocialMF) and cold start items

Table 5.6: Runtime comparison of a single iteration in training.

Model	Epinions	Flixster
SocialMF	2.8 sec	29 sec
ItemMF	2.7 sec	31 sec
SocialItemMF	4.0 sec	41 sec
STE	37 sec	27 min

Table 5.7: Total time required to learn the parameters of models.

Model	Epinions	Flixster
SocialMF	40 min	5.5 hr
ItemMF	37 min	5.6 hr
SocialItemMF	61 min	7.8 hr
STE	5 hr	9 days

(ItemMF). SocialMF regularizes the latent factors of a user by the latent factors of his direct neighbors in the social network, representing the effect of social influence in the model. Similar to the most related model, STE, presented in [69], SocialMF learns the latent factors of users and items. Different from STE, the latent factor of each user is dependent on the latent factors of his direct neighbors in the social network. This allows SocialMF to handle the transitivity of trust and trust propagation, which is not captured by the STE model. Trust propagation has been shown to be a crucial phenomenon in the social sciences, in social network analysis and in social network-based recommendation. Also if a user has not expressed any ratings, his latent factors can be learned as long as he is connected to the social network via a social relation.

In ItemMF, the latent factors of items are regularized by the latent factors of their neighbors in the item (similarity) graph. The item graphs are constructed using public domain data available for items that are independent from the rating history, e.g. movie genres and IMDB ratings in Flixster and product name and category in Epinions. The assumption behind the use of an item graph is that neighboring items have similar features and therefore similar ratings. The use of an item graph is particularly helpful for cold start items. Note that even if there are no ratings for an item, its latent factors can be learned by ItemMF using the latent factors of its neighbors in the item graph. The proposed SocialItemMF model is a combination of SocialMF and ItemMF, in which the latent

factors for both users and items are regularized by the social network and item graph, respectively. Therefore SocialItemMF can handle both cold start users and cold start items.

We performed experiments on two real life data sets from Epinions and Flixster, demonstrating the merits of our proposed models. SocialMF achieves substantial RMSE gain over existing models, in particular for cold start users. ItemMF outperforms the comparison partners, particularly for cold start items. Experimental results show that SocialItemMF benefits from the desirable properties of both SocialMF and ItemMF and outperforms state-of-the-art models. In particular, SocialItemMF achieves substantial RMSE gain over comparison partners for cold start users and cold start items.

In the next Chapter, we propose an advanced model-based approach that not only addresses the rating prediction problem but can also be employed to perform link prediction and community inference.

Chapter 6

A Generalized Stochastic Block Model for Comprehensive Community-based Recommendation in Social Rating Networks

6.1 Introduction

Exploiting social networks in recommendation works because of the effects of selection [94] [44] and social influence [33]. These effects have been modeled in Chapter 3 and have been exploited in recommender systems in Chapters 4 and 5. Social influence and selection together lead to the formation of communities of like-minded and well-connected users. Social influence can produce network-wide uniformity, as a new behavior spreads across the links. Selection, on the other hand, tends to drive the network toward smaller clusters of like-minded individuals [45]. The tendency of people to come together and form groups is inherent in the structure of society [8], and studying the ways in which such groups take shape and evolve over time is a theme that runs through large parts of social science research [25]. Users may belong to multiple groups, i.e., when a social actor interacts with different partners, different social contexts may apply and thus the actor may be acting according to the different roles they can possibly play [3]. For example, when a university professor interacts with his student on a social networking website, he belongs to the professors' group, but when interacting with his son on the same social networking website, he will play the role of a

father rather than a university professor. Similarly, in a social rating network, the context or group of an item could be different when different users from different groups rate it. For example, a digital camera may belong to the group of advanced cameras when being rated by a professional photographer, and the same camera can be considered in the group of expensive cameras for an amateur photographer.

Exploiting the clustering of users is one of the most important approaches for model-based recommendation [109]. To make a recommendation for a user, a clustering-based approach aggregates the ratings in the cluster of that user. However, only a few clustering algorithms allow clusters to overlap, which is crucial for communities in social rating networks. One of these methods, the probabilistic EM clustering method [17], assumes that data has been generated from a mixture of Gaussian models and learns for every data point a distribution of membership over the groups. Mixture models are not immediately applicable to relational data such as social networks because they assume that the objects are conditionally independent given their cluster assignments [3]. The stochastic blockmodel [104] is an adaptation of mixture modeling to relational data. In that model, each object belongs to one cluster and the relationships between objects are governed by their corresponding pair of clusters. The stochastic blockmodel suffers from the limitation that each object can only belong to one cluster, or in other words, can play a single latent role. The mixed membership stochastic blockmodel (MMB) [3] relaxes the assumption of single-latent role for actors and learns a membership distribution over different groups. Every user is associated with a probability distribution over the groups, i.e., he belongs to each group with a different degree of membership. Mixed membership models, such as LDA [18], have re-emerged in recent years as a flexible modeling tool for data where the single cluster assumption is violated [3].

In this chapter, we introduce a generalized stochastic blockmodel (GSBM) [54] that models not only the social relations but also the rating behavior. The proposed model learns the mixed group membership assignments for both users and items in an SRN. GSBM fills the gap between cluster-based models and social network-based approaches for recommendation. GSBM is capable of predicting both types of user behavior, rating of items and the creation of links to other users.

Basically, GSBM is a generative model that captures the behavior of users in a social rating network, including the creation of social relations and the ratings of items. Every user acts according to different groups he belongs to in each of his actions, whether he is creating a social relation or is rating an item. The items that are being rated by users also belong to different groups based on the users and the context in which they are being rated. The proposed GSBM ia a probabilistic method for finding communities in a social rating network taking into account both types of users

behaviors. Every user and every item has a mixed membership assignment latent vector. In every action (creating a social relation, or rating an item), the user is probabilistically considered to be acting as a member of one of the groups. Also every item is considered to belong to a latent group when it is being rated.

The rest of this chapter is organized as follows: In Section 6.2, some related work are discussed. The generalized stochastic block model is proposed in Section 6.3. The variational inference method for learning the model parameters is discussed in Section 6.4. Implementation details for GSBM are discussed in Section 6.5. Section 6.6 presents the design of experiments and the experimental results on two real life data sets.

6.2 Related Work

Related work on recommendation in social networks, both memory-based and model-based approaches, have already been discussed in Chapter 2. In this chapter, we review clustering-based approaches for recommendation. Then, we briefly discuss the mixed membership stochastic block model (MMB) that is the basis for our proposed model.

Since GSBM is a clustering-based model for social rating networks, we briefly review the clustering-based approaches for recommendation. EM clustering has already been discussed in the introduction. A standard model-based recommendation algorithm uses k-means to cluster similar users [109]. Given a set of user rating profiles, the space is partitioned into k groups of users that are close to each other based on a measure of similarity. The discovered user clusters are then applied to the user-based neighborhood formation task, rather than individual profiles as used in the user-based collaborative filtering. To make a recommendation for a user u and target item i, a neighborhood of users belonging to the same group as the user u who also have rated i are considered and their ratings on item i are aggregated to compute the prediction.

The mixed membership stochastic blockmodel (MMB) [3] is a generative model to capture the behavior of users when creating social relations. Every user is associated with a probability distribution over the groups. For every user pair (u, v), the group assignments of u and v are sampled from the probability distribution associated to them. The probabilities of interaction among different groups are defined by a matrix of Bernoulli rates. The link creation from u to v is then sampled from the Bernoulli distribution corresponding to the interaction probability of the groups assigned to the users. MMB only models the social relations and does not model the rating behavior of users. We extend MMB to also capture the rating behavior of users.

6.3 The Generalized Stochastic Blockmodel

The effects of social influence and selection in an SRN lead to the formation of groups of users and items. EM clustering methods assume overlapping group assignments among users based on a mixture of Gaussian models, but they cluster the users based only on their attributes and do not consider the social relations. On the other hand, mixed membership stochastic blockmodels (MMB) offer a mixed membership model for relational data, considering only the relational data but not the attributes of users. In this section, we extend MMB and propose a generalized stochastic block model (GSBM) to capture the groups of users and items in SRNs.

In this chapter, we assume that a social rating network is given. The set of users in the SRN is denoted by $\mathcal{U}$ where $|\mathcal{U}| = N$ and the set of items is denoted by $\mathcal{I}$ where $|\mathcal{I}| = M$. The social relations are considered to be directed, and $t_{u,v}$ is the binary random variable indicating whether there is a social relation from user u to user v. Rating values are assumed to be integer numbers in the range [1,5]. Hence the rating of a user u on an item i can be represented by an indicator vector $\overrightarrow{R}_{u,i}$ in which all elements are set to zero except for the one corresponding to the rating value.

6.3.1 The proposed GSBM

GSBM assumes K_1 groups among users and K_2 groups among items. Every user u is associated with a latent group assignment vector $\overrightarrow{\Pi}_u$, where $\Pi_{u,t}$ denotes the probability of user u belonging to group t. Similarly, every item i is associated with a latent group membership vector $\overrightarrow{\Delta}_i$. For every user u, the indicator vector $\overrightarrow{z}_{u\rightarrow v}$ denotes the group membership of user u when interacting with user v. Also $\overrightarrow{z}_{u\leftarrow v}$ denotes the group membership of user v when being interacted by user u. The probabilities of interactions among different groups of users are defined by a matrix of Bernoulli rates $B_{T_{(K_1\times K_1)}}$, where $B_{T_{l,t}}$ represents the probability of having a social relation between a user from group l and a user from group t.

Indicator vector $\overrightarrow{x}_{u\rightarrow i}$ denotes the group membership of user u while rating item i, and $\overrightarrow{x}_{u\leftarrow i}$ denotes the group membership of item i when being rated by user u. These indicator vectors are sampled from the latent group membership vectors $\overrightarrow{\Pi}_u$ and $\overrightarrow{\Delta}_i$. The multinomial distribution of the rating expressed by a user from a particular user group on an item from a specific item group is defined by a matrix $B_{R_{(K_1\times K_2)}}$, where $B_{R_{l,t}}$ is a probability vector representing the probabilities of the multinomial distribution for the rating expressed by a user in group l on an item in group t.

The proposed GSBM assumes the following generative model for social relations and ratings:

- For each user $u \in \mathcal{U}$:
  - Draw a K_1 dimensional mixed membership vector: $\overrightarrow{\Pi}_u \sim Dirichlet(\overrightarrow{\alpha})$
- For each item $i \in \mathcal{I}$:
  - Draw a K_2 dimensional mixed membership vector: $\overrightarrow{\Delta}_i \sim Dirichlet(\overrightarrow{\beta})$
- For each pair of users $(u, v) \in \mathcal{U} \times \mathcal{U}$:
  - Draw membership indicator for the initiator: $\overrightarrow{z}_{u \to v} \sim Multinomial(\overrightarrow{\Pi}_u)$
  - Draw membership indicator for the receiver: $\overrightarrow{z}_{u \leftarrow v} \sim Multinomial(\overrightarrow{\Pi}_v)$
  - Sample the social relation: $t_{u,v} \sim Bernoulli(\overrightarrow{z}^{\top}_{u \to v} B_T \overrightarrow{z}_{u \leftarrow v})$
- For each pair of a user and an item $(u, i) \in \mathcal{U} \times \mathcal{I}$:
  - Draw membership indicator for the user: $\overrightarrow{x}_{u \to i} \sim Multinomial(\overrightarrow{\Pi}_u)$
  - Draw membership indicator for the item: $\overrightarrow{x}_{u \leftarrow i} \sim Multinomial(\overrightarrow{\Delta}_i)$
  - Sample the rating value: $\overrightarrow{R}_{u,i} \sim Multinomial(\overrightarrow{x}^{\top}_{u \to i} B_R \overrightarrow{x}_{u \leftarrow i})$

6.3.2 Modeling the Sparsity

It is useful to distinguish two sources of non-interaction among users: they may be the results of rarity of interactions in general, or they may be an indication that the pair of relevant groups rarely interact. Thus, the authors of [3] introduced a sparsity parameter $\rho_T \in [0, 1]$ to characterize the source of non-interaction. Instead of sampling $t_{u,v}$ directly from Bernoulli with aforementioned parameter, they down-weight the probability of successful social interaction to $(1 - \rho_T) \overrightarrow{z}^{\top}_{u \to v} B_T \overrightarrow{z}_{u \leftarrow v}$.

For ratings, we introduce two levels of sparsity: A general sparsity parameter ρ_R, and a group level sparsity matrix B_Y whose elements indicate the rarity of ratings by a group of users on a group of items. In this case the process of generating a rating would be as follows:

For every pair of a user and item $(u, i) \in \mathcal{U} \times \mathcal{I}$:

- Draw membership indicator for the initiator: $\overrightarrow{x}_{u \to i} \sim Multinomial(\overrightarrow{\Pi}_u)$
- Draw membership indicator for the receiver: $\overrightarrow{x}_{u \leftarrow i} \sim Multinomial(\overrightarrow{\Delta}_i)$
- Sample the rating decision maker: $Y_{u,i} \sim Bernoulli((1 - \rho_R) \overrightarrow{x}_{u \to i} B_Y \overrightarrow{x}_{u \leftarrow i})$
  where B_Y is a $K_1 \times K_2$ matrix indicating the rating probabilities among different groups of users and items. ρ_R is the general rating sparsity parameter.

- Sample the rating value: $\overrightarrow{R}_{u,i} \sim Multinomial(\overrightarrow{x}^{\top}_{u \to i} B_R \overrightarrow{x}_{u \leftarrow i})$
  Note that this step will be skipped if $Y_{u,i} = 0$.

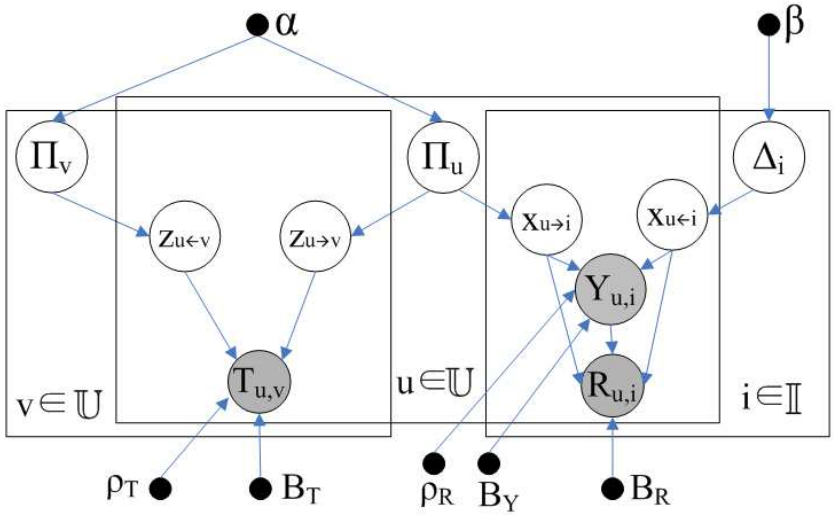


Figure 6.1: The graphical model underlying GSBM.

Figure 6.1 illustrates the graphical model underlying the proposed GSBM. Note that the random variable $t_{u,v}$ is a binary variable and is observed for all user pairs (u, v). If the social relation does not exist, then $t_{u,v} = 0$. Also, it should be noted that $Y_{u,i}$ is as an observed binary random variable defined for all pairs (u, i). Basically, $Y_{u,i}$ is the binary version of the observed ratings. If a rating $R_{u,i}$ does not exist, then $Y_{u,i} = 0$, i.e. it is observed as zero.

6.4 Posterior Inference and Parameter Estimation

As proposed in [3] for the MMB model, we use variational-EM[1] to learn the parameters of the GSBM model. The E-step is a variational inference on the latent variables. After learning the variational parameters for the latent variables, we perform the M-step similar to the conventional EM algorithm.

[1]Note that EM is the Expectation Maximization [29] algorithm used in machine learning. For details, please refer to [17, 29].

6.4.1 Variational Inference

The likelihood of the set of observed social relations T is computed as follows:

$$p(T, Z_{\rightarrow}, Z_{\leftarrow} | B_T, \overrightarrow{\Pi}_{1:N}, \rho_T) = \prod_u \prod_v P(t_{u,v} | \overrightarrow{z}_{u \rightarrow v}, \overrightarrow{z}_{u \leftarrow v}, B_T, \rho_T) P(\overrightarrow{z}_{u \rightarrow v} | \overrightarrow{\Pi}_u) P(\overrightarrow{z}_{u \leftarrow v} | \overrightarrow{\Pi}_v) \quad (6.1)$$

The likelihood of the set of observed ratings R is computed as following:

$$p(R, Y, X_{\rightarrow}, X_{\leftarrow} | B_R, B_Y, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}, \rho_R) = \prod_u \prod_i p(\overrightarrow{R}_{u,i} | \overrightarrow{x}_{u \rightarrow i}, \overrightarrow{x}_{u \leftarrow i}, B_R, Y_{u,i}) p(Y_{u,i} | \overrightarrow{x}_{u \rightarrow i}, \overrightarrow{x}_{u \leftarrow i}, B_Y, \rho_R) \times p(\overrightarrow{x}_{u \rightarrow i} | \overrightarrow{\Pi}_u) p(\overrightarrow{x}_{u \leftarrow i} | \overrightarrow{\Delta}_i) \quad (6.2)$$

Now, the likelihood of the complete observed data is as follows:

$$p(T, R, Y, Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow}, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M} | \alpha, \beta, B_T, B_R, B_Y, \rho_T, \rho_R) = p(T, Z_{\rightarrow}, Z_{\leftarrow} | B_T, \overrightarrow{\Pi}_{1:N}, \rho_T) p(R, Y, X_{\rightarrow}, X_{\leftarrow} | B_R, B_Y \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}, \rho_R) \times \prod_u p(\overrightarrow{\Pi}_u | \overrightarrow{\alpha}) \prod_i p(\overrightarrow{\Delta}_i | \overrightarrow{\beta}) \quad (6.3)$$

To learn the values for the latent variables we need to compute the posterior distribution of the latent variables given the observed data. The normalization constant of this posterior distribution is the marginal probability of the data which requires integral over all latent variables. This integral is not solvable in closed form. We use variational inference to approximate this. The main idea behind variational inference is to assume a distribution of the latent variables with free parameters, and then fit those parameters such that the distribution is close in KL-divergence to the true posterior. The variational distribution is simpler than the true posterior since it approximates the posterior by assuming independence among different latent variables.

The log of the marginal probability is bounded using Jensen's inequality[2] as follows:

$$\begin{aligned}\log p(T, R, Y, \alpha, \beta, B_T, B_R, B_Y, \rho_T, \rho_R) \geq \\ \mathbb{E}_q\big[\log p(T, R, Y, Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow}, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M} \\ |\alpha, \beta, B_T, B_R, B_Y, \rho_T, \rho_R)\big] \\ - \mathbb{E}_q\big[\log q(Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow}, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M})\big] \quad (6.4)\end{aligned}$$

The right hand side of Equation (6.4) is a lower bound for the log likelihood and we denote it by $\mathcal{L}$ in the rest of this chapter.

We introduce a distribution q of the latent variables depending on a set of free variational parameters γ, δ, ϕ, and ω. We specify q in a fully-factorized way as follows:

$$\begin{aligned}q(\overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}, Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow} | \overrightarrow{\gamma}_{1:N}, \overrightarrow{\delta}_{1:M}, \Phi_{\rightarrow}, \Phi_{\leftarrow}, \Omega_{\rightarrow}, \Omega_{\leftarrow}) = \\ \prod_u q_1(\overrightarrow{\Pi}_u | \overrightarrow{\gamma}_u) \prod_u \prod_v q_2(\overrightarrow{z}_{u\rightarrow v} | \overrightarrow{\phi}_{u\rightarrow v}) q_3(\overrightarrow{z}_{u\leftarrow v} | \overrightarrow{\phi}_{u\leftarrow v}) \\ \prod_i q_4(\overrightarrow{\Delta}_i | \overrightarrow{\delta}_i) \prod_u \prod_i q_5(\overrightarrow{x}_{u\rightarrow i} | \overrightarrow{\omega}_{u\rightarrow i}) q_6(\overrightarrow{x}_{u\leftarrow i} | \overrightarrow{\omega}_{u\leftarrow i}) \quad (6.5)\end{aligned}$$

In the above equation, q_1 and q_4 are Dirichlet distributions (similar to the distribution in the generative model). q_2, q_3, q_5 and q_6 are multinomial distributions with variational parameters. To optimize the variational parameters, we compute the derivatives of the lower bound for the likelihood with respect to all variational parameters and set them to zero. The learned variational parameters are as follows[3]:

$$\begin{aligned}\phi_{u\rightarrow v_l} \propto \exp\big(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1} \gamma_{u,t})\big) \times \\ \prod_{t=1}^{K_1} \big[((1-\rho_T) B_{T_{l,t}})^{t_{u,v}\phi_{u\leftarrow v_t}} (1 - (1-\rho_T) B_{T_{l,t}})^{(1-t_{u,v})\phi_{u\leftarrow v_t}}\big] \quad (6.6)\end{aligned}$$

[2] http://en.wikipedia.org/wiki/Jensen's_inequality

[3] For details of the computation of the lower bound and its derivatives, please refer to Appendix B

$$\phi_{u\leftarrow v_l} \propto \exp\big(\psi(\gamma_{v,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{v,t})\big)\times \prod_{t=1}^{K_1}\big[((1-\rho_T)B_{T_{l,t}})^{t_{u,v}\phi_{u\to v_t}}(1-(1-\rho_T)B_{T_{l,t}})^{(1-t_{u,v})\phi_{u\to v_t}}\big] \quad (6.7)$$

In the above equation, ψ is the digamma function. Also for ω values, we have:

$$\omega_{u\to i_l} \propto \exp\big(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t}\big) \times \Big(\prod_{n=1}^{K_2}\big[\prod_{t=1}^{5}B_{R_{l,n,t}}{}^{R_{u,i,t}}\big]^{\omega_{u\leftarrow i_n}}\Big)^{Y_{u,i}}\times \prod_{n=1}^{K_2}\big[((1-\rho_R)B_{Y_{l,n}})^{Y_{u,i}\omega_{u\leftarrow i_n}}(1-(1-\rho_R)B_{Y_{l,n}})^{(1-Y_{u,i})\omega_{u\leftarrow i_n}}\big] \quad (6.8)$$

$$\omega_{u\leftarrow i_l} \propto \exp\big(\psi(\delta_{i,l}) - \psi(\sum_{t=1}^{K_2}\delta_{i,t}\big) \times \Big(\prod_{m=1}^{K_1}\big[\prod_{t=1}^{5}B_{R_{m,l,t}}{}^{R_{u,i,t}}\big]^{\omega_{u\to i_m}}\Big)^{Y_{u,i}} \times \prod_{m=1}^{K_1}\big[((1-\rho_R)B_{Y_{l,m}})^{Y_{u,i}\omega_{u\to i_m}} (1-(1-\rho_R)B_{Y_{l,m}})^{(1-Y_{u,i})\omega_{u\to i_m}}\big] \quad (6.9)$$

The learned variational parameters for group memberships are as follows:

$$\gamma_{u,l} = \alpha_l + \sum_v \phi_{u\to v_l} + \sum_v \phi_{v\leftarrow u_l} + \sum_i \omega_{u\to i_l} \quad (6.10)$$

$$\delta_{i,l} = \beta_l + \sum_u \omega_{u\leftarrow i_l} \quad (6.11)$$

6.4.2 Parameter Estimation

We compute the empirical Bayes estimates of the model hyper parameters $\{\overrightarrow{\alpha}, \overrightarrow{\beta}, B_T, B_R, \rho_T, \rho_R\}$ in the M-step of the variational expectation-maximization (EM) algorithm. A closed form solution for the approximate maximum likelihood estimate of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ does not exist [3]. We use a linear

time Newton-Raphson method. The approximate maximum likelihood estimator of B_T, B_Y and B_R are:

$$B_{T_{i,j}} = \frac{\sum_u \sum_v t_{u,v} \phi_{u \to v_i} \phi_{u \leftarrow v_j}}{(1-\rho_T) \sum_u \sum_v \phi_{u \to v_i} \phi_{u \leftarrow v_j}} \tag{6.12}$$

$$B_{Y_{m,n}} = \frac{\sum_u \sum_i Y_{u,i} \omega_{u \to i_m} \omega_{u \leftarrow i_n}}{(1-\rho_R) \sum_u \sum_i \omega_{u \to i_m} \omega_{u \leftarrow i_n}} \tag{6.13}$$

$$B_{R_{m,n,l}} = \frac{\sum_u \sum_i Y_{u,i} (\omega_{u \to i_m} \omega_{u \leftarrow i_n} R_{u,i,l})}{\sum_u \sum_i Y_{u,i} (\omega_{u \to i_m} \omega_{u \leftarrow i_n})} \tag{6.14}$$

Also, for the sparsity parameter, we have:

$$1-\rho_T = \frac{\sum_u \sum_v \sum_i \sum_j \phi_{u \to v_i} \phi_{u \leftarrow v_j} t_{u,v}}{\sum_u \sum_v \sum_i \sum_j \phi_{u \to v_i} \phi_{u \leftarrow v_j} B_{T_{i,j}}} \tag{6.15}$$

$$1-\rho_R = \frac{\sum_u \sum_i \sum_m \sum_n \omega_{u \to i_m} \omega_{u \leftarrow i_n} Y_{u,i}}{\sum_u \sum_i \sum_m \sum_n \omega_{u \to i_m} \omega_{u \leftarrow i_n} B_{Y_{m,n}}} \tag{6.16}$$

6.5 Implementation

The variational-EM algorithm requires a lot of memory and needs to store $NK_1 + 2N^2K_1 + MK_2 + NMK_1 + NMK_2$ variational parameters for the latent variables which is $O(N^2K_1 + NMK_1 + NMK_2)$. To reduce the space complexity, we use the nested variational algorithm introduced in [3]. In each variational cycle, the nested variational algorithm (presented in algorithm 1) only needs to store $NK_1 + 2K_1 + NK_2 + MK_2 + K_1 + K_2$ parameters which is $O(NK_1 + NK_2 + MK_2)$.

Algorithm 1 presents the nested variational inference method for the GSBM model. In this algorithm, γ_u, γ_v, δ_i, the interaction probability matrices B_T, B_y and B_R are partially updated in each iteration. The main idea behind nested variational inference is the scheduling of updates for variational parameters. In the nested variational algorithm, variational parameters γ and δ and the interaction probability matrices B_T, B_y and B_R are partially updated in the E-step which leads to a great reduction in space requirement of the algorithm.

Algorithms 2 and 3 are used to compute the variational parameters ϕ and ω. Note that functions f_1 and f_2 in lines 5 and 9 of Algorithm 2 correspond to Equations (6.6) and (6.7), respectively. Also, functions f_3 and f_4 in lines 5 and 9 of Algorithm 3 correspond to Equations (6.8) and (6.9), respectively. While being much more space efficient, the nested variational inference leads to increased runtime due to the partial updates, However, algorithm 1 is easily parallizable on a shared memory

Algorithm 1 Variational Inference for Parameters

1: **Input:** B_T, B_R, $\overrightarrow{\alpha}$, $\overrightarrow{\beta}$, ρ_T, ρ_R
2: initialize $\forall u, l : \gamma_{u,l} = \frac{2N}{K_1}$ and $\forall i, s : \delta_{i,s} = \frac{2M}{K_2}$
3: **repeat**
4: **for** $u = 1$ to N **do**
5: **for** $v = 1$ to N **do**
6: using Algorithm 2, compute variational $\phi^{t+1}_{u \to v}$, $\phi^{t+1}_{u \leftarrow v}$.
7: partially update γ^{t+1}_u, γ^{t+1}_v and B^{t+1}_T
8: **end for**
9: **for** $i = 1$ to M **do**
10: using Algorithm 3, compute variational $\omega^{t+1}_{u \to i}$, $\omega^{t+1}_{u \leftarrow i}$.
11: partially update γ^{t+1}_u, δ^{t+1}_i, B^{t+1}_Y and B^{t+1}_R
12: **end for**
13: **end for**
14: **until** convergence

architecture. It should be noted that variational parameters γ and δ, and the interaction probability matrices B_T, B_Y and B_R are shared by all the processors. The main *for* loop in Algorithm 1 (lines 4-13) can be run in parallel since different iterations are independent. Therefore, GSBM gains linear speedup when it runs in parallel. We used *Intel Cilk Plus*[4] to implement the parallel version of our algorithm. Intel Cilk Plus is an extension to C and C++ that offers a quick, easy and reliable way to improve the performance of programs on multi-core processors. We used Cilk to implement the nested variational method presented in Algorithm 1 on a 20 core server using 16 GB of the available main memory.

6.6 Experiments

In this section, we present our experimental studies on two real life data sets from Epinions and Flixster. The detailed description and some statistics of these data sets are presented in the following subsection. We split the data set into 80% training and 20% test data, splitting both the set of social relations and the set of ratings expressed by users. We learn a GSBM using the training data. As discussed before, GSBM has two major applications: prediction of user behavior, and discovery of communities. We perform experiments to evaluate the accuracy of GSBM in predicting user behavior, in terms of both rating prediction and link prediction. We make predictions for withheld

[4]http://software.intel.com/en-us/articles/intel-cilk-plus/

Algorithm 2 Variational Computation for $\phi_{u\to v}$, $\phi_{u\leftarrow v}$

Input: $\overrightarrow{\gamma_u}$, $\overrightarrow{\gamma_v}$,
initialize $\forall l : \phi^0_{u\to v_l} = \phi^0_{u\leftarrow v_l} = \frac{1}{K_1}$.
repeat
  for $l = 1$ to K_1 **do**
    update $\phi^{s+1}_{u\to v} \propto f_1(\phi^s_{u\leftarrow v}, \overrightarrow{\gamma_u}, B_T)$
  end for
  normalize $\phi^{s+1}_{u\to v}$ to sum to 1
  for $l = 1$ to K_1 **do**
    update $\phi^{s+1}_{u\leftarrow v} \propto f_2(\phi^s_{u\to v}, \overrightarrow{\gamma_v}, B_T)$
  end for
  normalize $\phi^{s+1}_{u\leftarrow v}$ to sum to 1
until convergence

Algorithm 3 Variational Computation for $\omega_{u\to i}$, $\omega_{u\leftarrow i}$

Input: $\overrightarrow{\gamma_u}$, $\overrightarrow{\delta_i}$,
initialize $\forall l : \omega^0_{u\to v_l} = \frac{1}{K_1}, \forall m : \omega^0_{u\leftarrow i_m} = \frac{1}{K_2}$.
repeat
  for $l = 1$ to K_1 **do**
    update $\omega^{s+1}_{u\to i} \propto f_3(\omega^s_{u\leftarrow i}, \overrightarrow{\gamma_u}, B_R)$
  end for
  normalize $\omega^{s+1}_{u\to i}$ to sum to 1
  for $m = 1$ to K_2 **do**
    update $\omega^{s+1}_{u\leftarrow i} \propto f_4(\omega^s_{u\to i}, \overrightarrow{\delta_i}, B_R)$
  end for
  normalize $\omega^{s+1}_{u\leftarrow i}$ to sum to 1
until convergence

links/ratings by applying the trained model and compare the predictions to the withheld ground truth. Since there is no ground truth for communities in the social rating networks, we cannot directly evaluate the quality of community discovery. However, since prediction of user behavior is based on the community inference made by GSBM, the quality of predicting user behavior indirectly indicates the quality of community discovery performed by GSBM.

6.6.1 Data Sets

In our experiments, we used the real life data sets described in Chapter 4. Since GSBM, like MMB, models the behavior of users at the interaction level, it learns many latent variables. Therefore,

the model is slower than existing models. On the other hand, the model is capable of not only performing rating prediction, but also link prediction and community discovery. For the sake of efficiency, we used random samples of the Flixster and the Epinions data set in our experiments. Table 6.1 shows the general statistics of the sampled data sets from Epinions and Flixster used in our experiments. Note that the size of our samples is still fairly large compared to the data set used in [3] (which contained less than 1000 users). It should be noted that convergence of Algorithm 1 on each data set takes around 24 hours in the implementation described in section 6.4.

Table 6.1: General statistics of the Flixster and the Epinions data set used in the experiments with GSBM.

Statistics	Flixster	Epinions
Social Relations	41.5K	41.8K
Ratings	55.2K	62.2K
Users	10.6K	11.5K
Items	4.1K	8.8K

6.6.2 Experiments on Rating Prediction

GSBM can be used for rating prediction. After learning the parameters of GSBM from the training data, rating prediction can be performed using the learned latent variables. Given a user u and an item i we compute the predicted rating $\hat{r}_{u,i}$ as follows:

$$\hat{r}_{u,i} = \frac{\sum_{m=1}^{K_1} \sum_{n=1}^{K_2} \left(\gamma_{u,m} \delta_{i,n} \sum_{l=1}^{5} (l \times B_{R_{m,n,l}}) \right)}{\left(\sum_{m=1}^{K_1} \gamma_{u,m} \right) \left(\sum_{n=1}^{K_2} \delta_{i,n} \right)} \tag{6.17}$$

As is the standard in recommender systems, we use RMSE (root mean squared error) to evaluate the accuracy of the recommendations inferred from GSBM comparing them to the withheld ground truth ratings in the test data. We chose the following comparison partners:

- State-of-the-art methods for recommendation in social networks: TrustWalker [47], Tidal-Trust [36], MoleTrust [74] and SocialMF [49].
- State-of-the-art methods for recommendation using only the user item matrix: user based collaborative filtering method (CF [37]) and matrix factorization based approach (MF [99]).

Table 6.2: RMSE values of different comparison partners on Flixster and Epinions.

Model	Flixster	Epinions
CF	0.913	1.181
MF	0.911	1.175
TrustWalker	0.841	1.079
TidalTrust	0.887	1.109
MoleTrust	0.899	1.104
SocialMF	0.815	1.075
GSBM	0.884	1.092

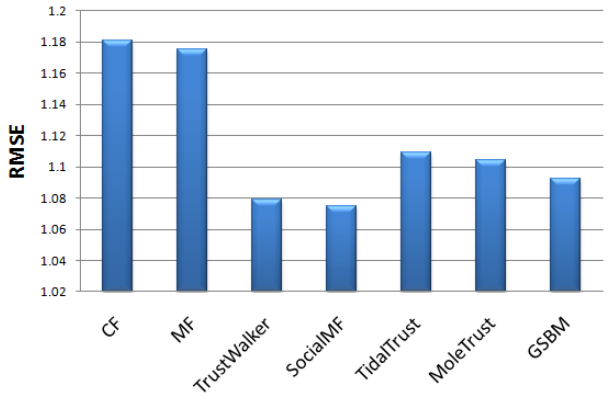


Figure 6.2: RMSE values for the comparison partners on Epinions.

Table 6.2 and Figures 6.2 and 6.3 present the RMSE values for the comparison partners. The proposed GSBM achieves a very good RMSE, clearly outperforming the standard recommendation methods CF and MF and social network-based approaches such as MoleTrust and TidalTrust. Compared to the more sophisticated methods for recommendation in social networks, such as SocialMF and TrustWalker, GSBM achieves lower but rather close performance. Note that all the comparison partners do only rating prediction, while GSBM is more comprehensive and also performs link prediction as well as community discovery. A more detailed discussion of the experimental results is presented in Section 6.4.

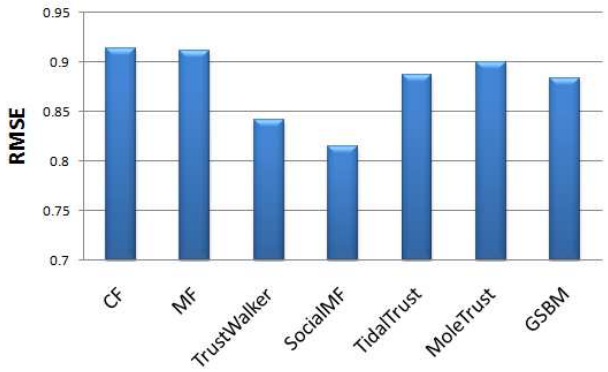


Figure 6.3: RMSE values for the comparison partners on Flixster.

6.6.3 Experiments on Link Prediction

The trained model can be employed for the task of link prediction in a social rating network. The probability $p(t_{u,v})$ of a link from user u to user v can be computed as follows:

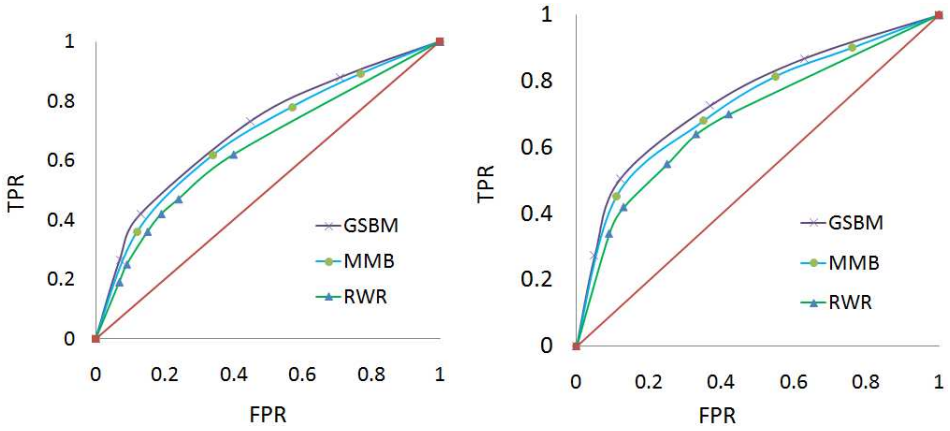
$$p(t_{u,v}) \propto \frac{\sum_{m=1}^{K_1}\sum_{n=1}^{K_1}\left(\gamma_{u,m}\gamma_{v,n}B_{T_{m,n}}\right)}{\left(\sum_{m=1}^{K_1}\gamma_{u,m}\right)\left(\sum_{n=1}^{K_1}\gamma_{v,n}\right)} \tag{6.18}$$

We threshold the link probability to predict whether or not u creates a link to v. We use the ROC curve[5], more specifically the area under the ROC curve, to compare our proposed model with existing models. Note that since we need to determine the false positive rate, we randomly add some pairs of users who do not have a social relation in the training data to the test data to have some ground truth for negative results[6]. We consider the mixed membership stochastic blockmodel (MMB) as the baseline method. We use the random walk with restart method (RWR) [89] as another comparison partner. RWR is a state-of-the-art probabilistic method for link prediction in social networks. The probabilistic nature of RWR together with its good performance allows us to generate ROC curves used in our performance evaluation.

Figure 6.4 presents the ROC curves for link prediction with GSBM, MMB and RWR on Flixster and Epinions. As shown in the figure, both GSBM and MMB clearly outperform RWR in Flixster and Epinions. Comparing the ROC curves of GSBM and MMB, we observe that GSBM achieves slightly better performance. We believe that this improvement is due to the exploitation of the

[5]http://en.wikipedia.org/wiki/Receiver_operating_characteristic

[6]The number of positive and negative test data is the same in our experiments.



(a) Epinions (b) Flixster

Figure 6.4: ROC curves for link prediction in GSBM.

additional information in the user rating matrix which allows GSBM to take the selection effect into account.

6.6.4 Discussion

Our experimental results show that GSBM achieves high quality both in rating prediction and link prediction. In rating prediction, GSBM clearly outperforms all comparison partners that only consider the rating matrix (CF and MF) and also outperforms the social recommenders that do not take the rating patterns into account for guiding the traversal of the social network (TidalTrust and MoleTrust). The more sophisticated social recommender methods that use the rating patterns in the search of the social network, however, slightly outperform GSBM. Note that all these comparison partners do only rating prediction, while GSBM covers more tasks and also performs link prediction as well as community discovery. Therefore, GSBM is generally expected to sacrifice some rating prediction accuracy in order to optimize all three tasks simultaneously.

In link prediction, GSBM achieves better quality than all the comparison partners. We believe that this is mainly due to the fact that GSBM can exploit the selection effect in social networks by also considering the rating patterns for the link prediction task. To summarize, our experimental results demonstrate that GSBM achieves high quality in both rating prediction and link prediction while providing a more comprehensive framework than existing methods that perform only one of these tasks.

6.7 Conclusion

In this chapter, we adopted the idea of overlapping and probabilistic group membership presented in the mixed membership stochastic blockmodel (MMB) [3] and extended it to model not only the creation of the social relations, but also the rating behavior for groups of users and items. The proposed generalized stochastic blockmodel, GSBM, is capable of predicting the future behavior of users, both the social relation creation and the ratings of items. Besides, GSBM can infer communities among users with probabilistic membership assignments.

We performed experiments on two real life data sets from Flixster.com and Epinions.com, employing GSBM for rating prediction and link prediction. Experimental results show that GSBM achieves high quality results in both tasks. Although GSBM does not outperform all comparison partners in rating prediction, it achieves results very close to the state-of-the-art methods while being able to perform multiple tasks (community discovery, rating prediction and link prediction). Due to exploitation of the rating matrix, the link prediction quality of GSBM is better than that of MMB. In other words, GSBM takes the selection effect into account which improves the performance in link prediction.

Chapter 7

Discussion

In this thesis we investigated several models for recommendation in social rating networks, including both memory-based and model-based approaches. We performed experiments on two real life data sets demonstrating that the proposed models achieve high quality compared to state-of-the-art methods. Yet, there are questions that we have not addressed in this dissertation. Questions such as: Given a social rating network data set, how do you say whether social network-based recommendation works on this data set? How can we compare the different models proposed in this thesis? What are the principles behind these models that could be extended to other problems and the whole field of social network-based recommendation? The proposed models substantially improve the state-of-the-art models, but how significant are the results? To what extent we can attribute the results to randomness? Finally, social network-based recommendation is particularly helpful for cold start users. Basically, if a user has expressed enough ratings, then the social network may not even be necessary for him to get high quality recommendation. In other words, we may not need the social network to model users with enough ratings. Now, the question is what is enough? How many ratings should a user have so that the difference of the social network-based and traditional recommendation for him is very little?

In this chapter, we discuss the above mentioned questions and propose some statistics to present more insights on social network-based recommendation, the proposed models, and the experimental results.

7.1 Is it helpful to use social network-based recommendation?

Given a social rating network data set, before performing the actual evaluation of a social network-based recommender, is it possible to verify whether social network-based recommendation is useful? Is it worth investing on collecting a social network among users? In this section we address these questions.

The social network-based approach is most effective in systems where users' evaluations of items are subjective rather than objective. In other words, any social network in which people can have different tastes on items is potentially a strong candidate for social network-based recommendation. Movies, books, music and travel are examples of items that fit the social network-based approach. On the other hand, cameras, printers, etc. are items for which opinions of friends in a social network are not as important. Basically, items for which relevant features (e.g. resolution, zoom, and shutter lag for digital cameras) can be identified are potentially not so dependent on the social data. However, for cold start users, all these data sets might require the additional information provided by social network.

The above discussions are all based on intuitions and provide a general guideline. In addition, we can perform some statistical test to check whether social networks make a difference in the quality of recommendation in a data set. To illustrate the significance of the effect of social networks in modeling users' behavior, we compare the similarity of ratings expressed on common items by random pairs of users against pairs of users that are direct neighbors in the social network. Basically for every single item rated by both users in a pair, we compute the Manhattan distance[1] of their ratings on the item and compute the average of these differences over all pairs of users to define a metric of similarity among a specific set of user pairs in a data set.

$$F_P = \frac{\sum_{(u,v)\in P}\left(\sum_{i\in C_{u,v}} |r_{u,i} - r_{v,i}|\right)}{\sum_{(u,v)\in P}(\sum_{i\in C_{u,v}} 1)} \tag{7.1}$$

In Equation (7.1), F_P is the metric computing the similarity of ratings expressed by the set P of user pairs on common rated items. In our experiments we have a set of all possible pairs of users and a set of pairs of users that are direct neighbors in the social network. Also, $C_{u,v}$ is the set of items rated by both users u and v.

Table 7.1 compares the F value for random pairs of users and direct neighbors in both Epinions

[1] http://en.wikipedia.org/wiki/Manhattan_distance

Table 7.1: Comparison of similarities in different set of user pairs

Data set	F_P	
	Random User Pairs	Direct Neighbors
Epinions	1.106	0.972
Flixster	1.155	1.081

and Flixster. As shown in this table, on average direct neighbors rate items in a substantially more similar way compared to random pairs of users. To show the significance of the difference and to make sure that the different is not attributed to random noise, we performed a statistical Z-test[2]. Our results show that the differences are statistically significant.

The results presented in Table 7.1 show that there is some non-random information embedded in the social networks of Epinions and Flixster, and it is up to social network-based models for recommendation to dig into the social network and make advantage of the additional information in the social network. Therefore, using a social network-based approach is potentially promising for Epinions and Flixster.

7.2 Comparison of different proposed models

In this thesis, we proposed several methods for recommendation in social networks, including both memory-based and model-based approaches. In this section we compare these models and present their commonalities and differences. Also, we discuss the principles and intuitions behind these models and how we can employ and extend them to other problems, particulary recommendation problems in social networks. In the following, we first discuss the general properties of the models proposed in this dissertation.

- **TrustWalker** [47] is a memory-based approach. The main advantage of a memory-based approach is that there is no training phase. On the other hand, the main disadvantage of TrustWalker as a memory-based approach is its low speed in the prediction (test) phase. TrustWalker needs to explore the social rating network in order to look for predicted ratings for an item, which is time consuming. Although TrustWalker is slow in predictions, since it is clear what ratings from which users are being used to compute the prediction, TrustWalker is

[2]http://en.wikipedia.org/wiki/Z-test

explainable compared to model-based approaches where we only learn latent factors for users and items which might not be easily interpretable.

- **SocialMF** [49] is a model-based approach extending matrix factorization. SocialMF has a training phase to learn the latent factors for users and items. On the other hand, SocialMF is very fast in the prediction (test) phase since it uses the learned latent factors to predict the ratings and does not explore the social rating network for prediction. The dependency on the learning phase means that newly arrived data is not considered in the model until the model is re-trained. It should be noted though that online learning methods for matrix factorization [111] [22] have been presented that are able to learn the factors incrementally.

- **GSBM** [54] is a model-based approach generalizing stochastic block models. GSBM is a comprehensive model capable of performing rating prediction, link prediction, and clustering. Therefore, there is a trade-off between the accuracy of rating prediction and the range of tasks that GSBM is able to deal with.

Generally, model-based approaches are expected to achieve better results in terms of RMSE compared to memory-based approaches. The reason is that model-based approaches learn the behavior of users and can uncover the latent factors affecting the behavior of users in a rating system. However, since model-based approaches rely on the learned models only and ignore the rating data in the prediction phase, they may loose some informative data that has not been capture by the model but is available to be used by memory-based approaches. Overall, it all depends on how effective the modeling is, or how well a memory-based approach can exploit the information available. In our experiments, SocialMF slightly outperforms TrustWalker. Also, GSBM achieves results comparable to the results of SocialMF and TrustWalker but not as good as them. One explanation could be the fact that GSBM is a comprehensive model capable of performing rating, link prediction and community inference which leads to a trade-off between the accuracy and the number of tasks GSBM is capable of addressing. Note that TrustWalker, SocialMF, and GSBM outperform all the state-of-the-art methods for recommendation in social networks, both memory-based and model-based ones. Also, GSBM outperforms existing models for link prediction.

There is another difference between SocialMF and TrustWalker. In TrustWalker, the social network has higher priority than the rating matrix. Basically, TrustWalker walks on the social network and the rating matrix is used for transition probabilities, stopping criteria, and item selection. In other words, if a user is not connected to the network but has a lot of expressed ratings, TrustWalker

is not able to compute any recommendation for this user. On the other hand, SocialMF decomposes the rating matrix into the product of the latent factors of users and items. The social network is used to regularize the latent factors. In SocialMF (see Equation (5.8)), if a user has expressed many ratings, then the effect of the social network in the learning phase would be less than the effect for cold start users. As we discuss later in section 7.4, if a user has expressed enough ratings the social network does not improve the recommendation accuracy substantially. Therefore, this property of SocialMF is actually an advantage over TrustWalker.

The decision of what method to employ for a recommendation system depends on various issues. If the objective is to have the lowest average error and fast prediction, then model-based approaches such as SocialMF are recommended. If one does not want to spend time and resources on the training phase while having high quality recommendations, it is better to go for a memory-based approach such as TrustWalker. If explainability of the recommendations is an issue, memory-based approaches such as TrustWalker are preferable over model-based approaches. Finally, if one is looking for a more comprehensive model that is capable of performing rating prediction, link prediction, and clustering, GSBM is the model of choice. However, due to the large number of parameters to be learned by GSBM, it is not very scalable.

The principles and intuitions of the models proposed in this dissertation can be extended to other problems of social network-based recommendation. For example, the general idea of performing random walks on graphs with attributes on nodes can be extended to problems from other domains, e.g. in crime networks to suggest criminal suspects [106]. The input data for the CrimeWalker model proposed in [106] is the crime data consisting of the history of crimes and the criminals involved in each crime. We used the crime data to construct a co-crime network among criminals. Then given a set of arrested criminals in a crime, CrimeWalker performs random walks similar to those in TrustWalker to suggest new suspects for the crime.

Generally, random walks such as those in TrustWalker compute a similarity measure between any pair of users in graph with attributes on nodes. Moreover, note that social rating networks can be considered as a combination of a graph among users and a bipartite graph among users and items. Social rating networks can be generalized to include multiple entities (i.e., user, item, etc.) and the graphs among them. Predicting a rating for a user can be considered as computing the affinity of a user to an item [40] [118]. In the general form, random walk methods can be used to compute the affinity of one entity to another entity. These entities could be either from the same type, e.g., in link prediction, or from different types, e.g, in rating prediction.

The regularization idea proposed in SocialMF can generally be employed to address cold start problems. As discussed in Chapter 5 we can regularize the item factors to deal with cold start items. Generally any network among users or items can be employed for regularization to enhance the modeling in a recommender system. Also, the matrix factorization base approach can be extended to N-dimensional tensor factorization to learn the latent factors for more than two entity types [64]. For instance, we used tensor factorization in opinion mining to learn the latent factors of reviewers, raters, and products [82]. It should be noted that the model proposed in [82] can extended to exploit social regularization to regularize the latent factors for reviewers and raters. Tensor factorization has also been employed for context-aware recommendation [55] and social tag recommendation [95].

Finally, the idea proposed in GSBM [54] may be applicable in protein-protein interaction (PPI) networks. Protein-protein interactions occur when two or more proteins bind together, often to carry out their biological function. Every protein interacts with specific proteins to perform a particular function (corresponding to different groups in GSBM). Interactions in PPI networks can be modeled by the ideas proposed in GSBM. Basically, every protein would have a mixed membership functionality assignment.

7.3 Significance of the experimental results

Assessing the significance of data mining results is an important step in the knowledge discovery process. While results might appear interesting at a first glance, they can often be explained by already known characteristics of the data. Randomization is an established technique for significance testing, and methods to assess data mining results on vector data or network data were proposed [34] [39]. Basically, large number of randomized data sets are generated and a experiments are performed on every randomized data set. The experimental results for the randomized data sets are compared to the results on the original data set to test the significance of the results on the original data set.

In this section we assess the significance of the experimental results for the SocialMF model presented in this dissertation[3]. The randomization method is based on the model proposed in [35] that randomizes both ratings data and the social network while keeping some fundamental characteristics of the social rating network. Basically, the randomization model proposed in [35] preserves correlation information between the ratings and social relation, i.e. social influence, homophily, etc. In other words, the novelty of the proposed model in [35] lays in taking the dependency of

[3]Note that we picked SocialMF over TrustWalker and GSBM since it achieves best RMSE results.

ratings and social relations into account. We introduced a swap randomization technique that exploits an adaptive Metropolis sampling [6] [15] and interweaves attribute randomization and graph randomization steps [35].

In the following, we report our experimental results for the significance of results in SocialMF on both Flixster and Epinions. The significance of the RMSE results on the SRN is determined by the (one-tailed) empirical p-value[4]. If the p-value is sufficiently small, the null hypothesis is rejected, i.e. we can assume that the obtained results are not a consequence of the preserved background knowledge. In other words, a small p-value in a data set means that the result of SocialMF on that data set is significant.

Table 7.2: RMSE and significance test results for SocialMF

Data set	RMSE orig.	RMSE random.	p-value
Epinions	1.153	1.214	0.01
Flixster	0.875	0.945	0.01

Table 7.2 presents the results of significance test for SocialMF on both Flixster and Epinions[5]. In both data sets, the average RMSE of the models learned for the randomized data sets is substantially higher than the RMSE for the original data set. The p-values are smaller or equal to 0.01, i.e., with probability of at least 99% the results achieved by SocialMF are not attributed to the characteristics of the data sets preserved in the randomization [35].

7.4 How many ratings are enough for recommendation?

In this thesis, we have shown that exploiting social networks can improves the quality of recommendation, particularly for cold start users. The RMSE gain for cold start users is substantially higher for cold start users compared to all users. An important question to be answered in social network-based recommendation is that whether we always need to use the help of social networks for high quality recommendation. As discussed before, if a user has already expressed enough ratings, the gain achieved by social network-based recommendation is not substantial.

[4] http://en.wikipedia.org/wiki/P-value

[5] Note that both the swap randomization of the data sets and the learning of many models (one for each randomized data set) are computationally expensive. Therefore, we sampled the data sets in order to be able to efficiently learn models for several randomized data sets.

In this section, we perform more fine grained analysis to compare the results of recommendation models on users with different numbers of ratings. Figures 7.1 and 7.2 compare the RMSE of SocialMF [49] and BaseMF [99] for users with different number of ratings per user. As shown in Figure 7.1, if a user has more than 9 ratings in Epinions, the RMSE difference of the social network-based approach (SocialMF) and the baseline matrix factorization based method (BaseMF) is less than 5%. Similarly, the RMSE gain of SocialMF over BaseMF for users with more than 12 ratings in Flixster is less than 5%. Note that the gain for users with only one rating is 24% in Epinions and 19% in Flixster. The experimental results presented in Figures 7.1 and 7.2 confirm that the more ratings a user has, the less dependent the recommender system will be on the social network to model this user's behavior.

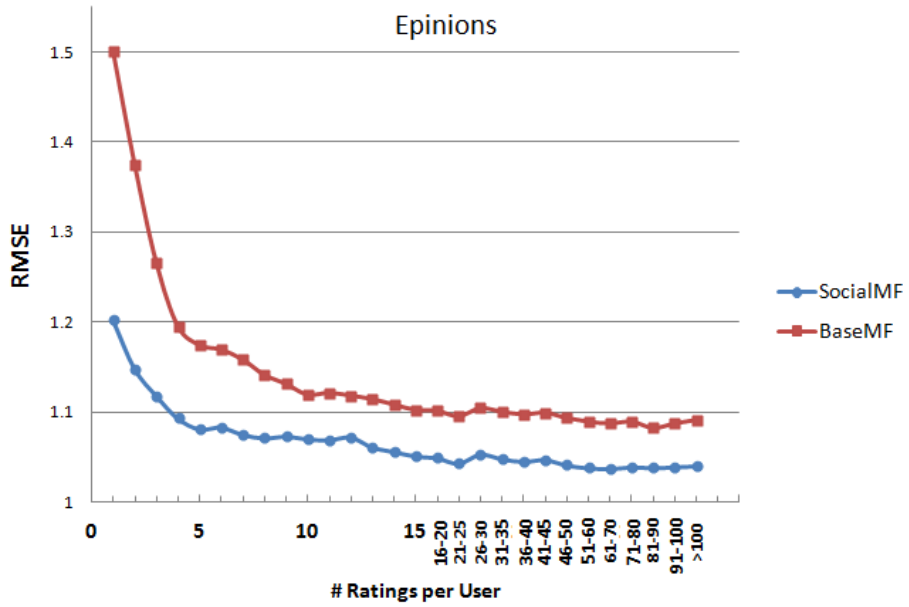


Figure 7.1: Distribution of RMSE of SocialMF for users with different number of ratings in Epinions

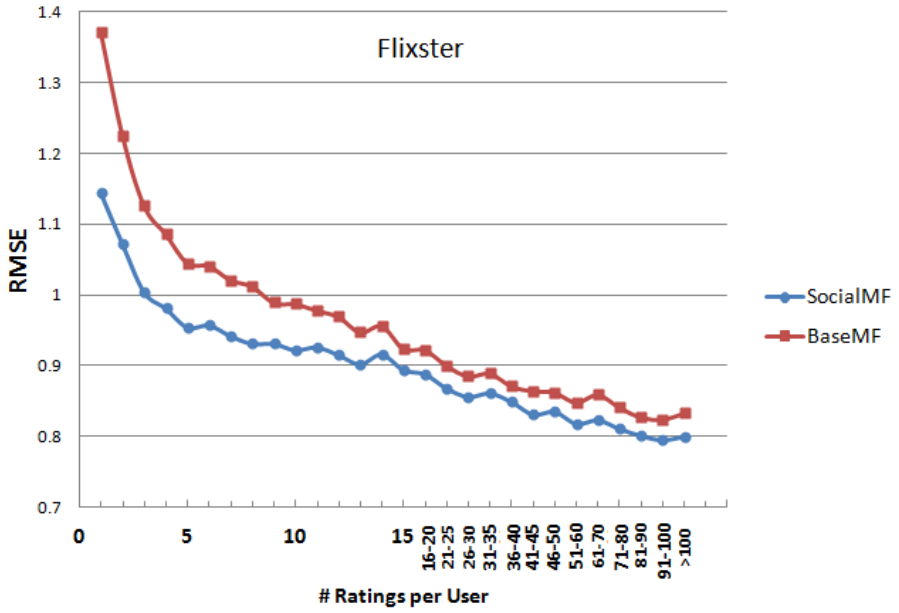


Figure 7.2: Distribution of RMSE of SocialMF for users with different number of ratings in Flixster

Chapter 8

Conclusion

Recommender systems have emerged as tools of choice to select the online information relevant to a given user. Collaborative filtering and matrix factorization are the most popular approaches to building recommender systems and have been successfully employed in many applications. However, these methods perform poorly for cold start users.

With the advent of online social networks, exploiting the information hidden in the social network to predict the behavior of users has attracted a lot of attention. Therefore, social network-based approaches for recommendation have emerged. Social network-based recommendation is particularly helpful for cold start users. The input data for recommendation in social networks is a social rating network, i.e., a social network together with the users' item ratings

Exploiting social networks in recommendation works because of the effects of selection and social influence. Social selection means that people tend to create social relations to people with similar attributes. Social influence implies that people adopt the rating behavior of their friends. Before proposing recommendation models in this dissertation, in Chapter 3, we first modeled the temporal dynamics of user behavior in social rating networks to capture these effects and to better understand the underlying mechanisms of user behavior in a social network.

In this thesis, we proposed several probabilistic methods for recommendation in social networks, including memory-based and model-based approaches. We proposed a memory-based approach, TrustWalker [47], in Chapter 4. TrustWalker is a random walk based method that combines social network-based and similarity-based approaches for recommendation. TrustWalker performs random walks on the social rating network to compute the estimated rating for a specific user on the target item. In each walk, TrustWalker considers not only ratings on the target item, but also those of similar items, with probability increasing with increasing length of the walk. Experimental results on

two real life data sets from Flixster.com and Epinions.com show that TrustWalker outperform state-of-the-art memory-based approaches for recommendation in social networks. We also extended the idea of TrustWalker to other recommendation problems such as top-N recommendation [48] and link prediction [51]. Experiments on Flixster and Epinions demonstrate that the proposed models outperform the comparison partners for top-N recommendation and link prediction. Note the the Flixster data set used in this thesis has been crawled and made publicly available for research purposed as part of this thesis.

As our first model-based approach, we proposed SocialItemMF [52], a comprehensive matrix factorization model consisting of two main components to deal with cold start users (SocialMF [49] [50]) and cold start items (ItemMF [52]). In SocialMF, the latent factors of users are regularized by the latent factors of their direct neighbors in the social network. As discussed in Chapter 5, dealing with cold start items is also very important. ItemMF has been proposed to address cold start items, in which the latent factor of an item is regularized by the latent factors of its neighbors in the item similarity graph. We combined the ideas employed in SocialMF and ItemMF to present a comprehensive model, SocialItemMF. In SocialItemMF, the latent factors for both users and items are regularized by the social network and item graph, which enables SocialItemMF to handle both cold start users and cold start items. Experimental results on Epinions and Flixster show that SocialItemMF outperforms existing models. In particular for cold start users and items, the accuracy gain for SocialItemMF compared to the state-of-the-art methods is substantial.

In Chapter 6, we proposed GSBM [54], a generalized stochastic blockmodel that models both social relations and the the rating behaviors. GSBM extends the idea proposed in stochastic block models and learns the mixed group membership assignments for both users and items in an SRN. GSBM fills the gap between clustering-based models and social network-based approaches for recommendation. GSBM is capable of predicting both types of user behavior, rating of items and the creation of links to other users. Compared to other model-based approaches for recommendation, GSBM is very comprehensive and able to perform multiple task simultaneously. However, experimental results show that although it achieves comparable high quality results on both link prediction and rating prediction, but GSBM does not outperform all the state-of-the-art models in rating prediction. It should be noted that methods outperforming GSBM in our experiments are TrustWalker and SocialMF that have been proposed in this thesis. In other words, GSBM outperforms all the existing models that have not been proposed in this dissertation.

In Chapter 7, we discussed some insightful questions regarding social network-based recommendation. We discussed about the characteristics of the data sets in which social network-based

recommendation potentially works and presented a statistical test to check whether social networks can add value to a recommender system. We also compared the different models in this dissertation and discussed which model to pick for different cases. We also applied a data randomization method to verify the significance of some of the experimental results in this dissertation. Finally, we experimentally evaluated the importance of social networks for cold start users, addressing the question of how many ratings are required for a user so that social network-based recommendation would not be necessary for him.

8.1 Future Work

This research suggests interesting directions for future work. In this section, we briefly discuss such directions for future research in the field of social network-based recommendation.

Enhancing the proposed models There are some enhancements that can improve the proposed models in this thesis. For instance, in SocialMF the hyper-parameters are not learned. If we use EM for learning, we will be able to learn the hyper-parameters in the M-step. Also, matrix factorization based models are potentially parallizable. Another improvement on SocialItemMF could be working on implementing a parallel version of SocialItemMF. Moreover, extending the idea in SocialMF to address the link prediction problem is also interesting. Finally, experimental results showed that GSBM does not outperform SocialMF or TrustWalker. We mentioned that one explanation could be the fact that GSBM is optimizing not only for rating prediction but also for link prediction which might lead to a tradeoff on the quality of recommendation. Working on the stochastic block model itself and how to model different components of GSBM could also lead to a better modeling of user behavior and therefore better results.

Distributed SRNs In this thesis and in the related work, the ratings are assumed to be stored in a centralized repository. However, applications such as mobile social networks require a distributed recommender, and a random walk model is a promising approach for such scenarios.

Context aware trust The trust concept we considered in this dissertation is context independent. However, people may trust other people in some context while they do not trust those people in other contexts. Investigating context-based trust models for recommendations is also an interesting direction for future work.

Temporal Recommendation Ratings expressed by users have a temporal dimensions. Basically, there is a difference between the ratings that have been expressed recently and the ratings expressed long time ago. Temporal aspects of ratings have been investigated for traditional recommender systems [59]. Incorporating the temporal dynamics of ratings in social network-based recommendation is a potential direction for future work that has not been investigated yet.

Multiple social networks Existing models for recommendation and personalization in social networks assume that a single social network is given as the input, and the social relations are explicitly expressed by users. However, in practice users are reluctant to reveal their social relations, mainly due to the additional effort required to express and maintain these relations, and also due to privacy concerns. On the other hand, users interact with each other extensively on social media sites, e.g., by commenting on a news article or replying to another users comment. These interactions lead to the formation of implicit social networks. Multiple implicit social networks can be inferred based on various interactions among users in different contexts. Employing multiple social networks in the recommendation process is a very interesting direction to extend this thesis.

Negative Trust Social relations in this thesis and most related work are considered to imply a positive correlation between the two corresponding users. However, in some online social networks (including Epinions.com) user can expressed a negative social relation or so-called distrust to other users. Handling negative social relations is very different from handling positive social relations [38]. Taking distrust into account for social network-based recommendation has received very little attention [70] and is a potentially interesting direction for future work.

Online learning Model-based approaches for recommendation require a training phase to learn the model parameters. Training is expensive and can not be done very frequently, especially if the data sets are very large. Continuous streams of rating data are arriving at the system. Since the training is far less frequent than the arrival of the new rating data, it takes a while for newly arrived ratings to affect the user/item modeling of the system. To address this issue, online learning [111] [22] has emerged that is able to learn the latent factors incrementally and one instance at a time. Working on online learning methods for the proposed model in this dissertation is very useful and interesting.

Appendix A

Proof of Convergence of $\mathbf{P_i^*}$ and Variance in TrustWalker

In this appendix we provide proofs for convergence of $\mathbf{P_i^*}$ and variance of the results of different random walks in the TrustWalker model presented in Chapter 4.

A.1 Convergence of $\mathbf{P_i^*}$

If we consider a square matrix $\mathbf{A}$, then the geometric series on $\mathbf{A}$ can be defined as $\sum_{k=1}^{\infty} \mathbf{A}^k$.

Lemma 1 $(\mathbf{I} - \mathbf{A})(\mathbf{A} + \mathbf{A^2} + \mathbf{A^3} + ... + \mathbf{A^k}) = (\mathbf{I} - \mathbf{A^k})$ If all λ_i values for $\mathbf{A}$ are nonequal to 1, then $(\mathbf{I} - \mathbf{A})$ is invertible, and we'll have:

$$(\mathbf{A} + \mathbf{A^2} + \mathbf{A^3} + ... + \mathbf{A^k}) = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{A} - \mathbf{A^k}). \tag{A.1}$$

Theorem 1 If $|\lambda_i| < 1$ for each eigenvalue $|\lambda_i|$ of $\mathbf{A}$, the geometric series generated by $\mathbf{A}$ converges. Then, we will have:

$$\sum_{k=1}^{\infty} \mathbf{A}^k = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{A} \tag{A.2}$$

Proof According to eigen decomposition of matrices, we can decompose the matrix as $\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$, where $\mathbf{D}$ is a diagonal matrix formed from the eigenvalues of $\mathbf{A}$, and the columns of $\mathbf{V}$ are the corresponding eigenvectors of $\mathbf{A}$. If we expand this decomposition, we will get:

$$\mathbf{A} = \sum_{i=1}^{N} \lambda_i \mathbf{V}_i$$

$$\mathbf{A}^k = \sum_{i=1}^{N} \lambda_i^k \mathbf{V}_i^k$$

In the above equation, each $\mathbf{V}_i$ is a matrix built from eigenvector corresponding λ_i. $\mathbf{V}_i$ has the property that $\mathbf{V}_i^k = \mathbf{V}_i$. As a result,

$$\mathbf{A}^k = \sum_{i=1}^{N} \lambda_i^k \mathbf{V}_i$$

Since we have $|\lambda_i| < 1$, we will have:

$$\lim_{k\to\infty} \mathbf{A}^k = \lim_{k\to\infty} \sum_{i=1}^{N} \lambda_i^k \mathbf{V}_i^k = \lim_{k\to\infty} \sum_{i=1}^{N} 0 \times \mathbf{V}_i = \mathbf{0}$$

Using lemma 1, we will get:

$$\sum_{k=1}^{\infty} \mathbf{A}^k = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{A} - \mathbf{A^k}) = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{A}$$

Lemma 2 Assume $\mathbf{P}$ is a probability matrix. If we multiply at least one row of the matrix by a positive real number $\alpha < 1$, then $|\lambda_i| < 1$ for all eigenvalues of the $\mathbf{P}$.

Proof According to Perron-Frobenius theorem[1], if $\mathbf{P}$ is a probability matrix, then $|\lambda_i| \leq 1$ for all eigenvalues of the $\mathbf{P}$. We can easily check that if we multiply one row of the matrix by $\alpha < 1$, then $|\lambda_i| \neq 1$.

Theorem 2 If we do not consider step k as a factor in $\phi_{u,i}$, then $\mathbf{P_i^*}$ will converge and the expression representing $\mathbf{P_i^*}$ can be simplified.

Proof Since we are not considering k, all matrices $\mathbf{\Phi_{k,i}}$ are the same and we will refer to them by $\mathbf{\Phi_i}$. Now, we can rewrite Equation (4.15) as follows:

$$\mathbf{P_i^*} = \sum_{k=1}^{\infty} (\mathbf{\Phi_i P})^k \tag{A.3}$$

Since multiplying $\mathbf{\Phi_i}$ into $\mathbf{P}$ will multiply at least one of the rows of $\mathbf{P}$ by a positive real number $\alpha < 1$, using lemma 2, we will have $|\lambda_i| < 1$ for all eigenvalues of $\mathbf{\Phi_i P}$. Now, using theorem 1, we will get:

$$\mathbf{P_i^*} = \sum_{k=1}^{\infty} (\mathbf{\Phi_i P})^k = (\mathbf{I} - \mathbf{\Phi_i P})^{-1}\mathbf{\Phi_i P} \tag{A.4}$$

[1] http://en.wikipedia.org/wiki/Perron-Frobenius_theorem

If we consider step k in $\mathbf{\Phi_{k,i}}$, the proof for convergence of $\mathbf{P}_\mathbf{i}^*$ would be similar. However, since matrices $\mathbf{\Phi_{k,i}}$ are different for different values of k, we can not simplify $\mathbf{P}_\mathbf{i}^*$. Therefore, unless we want to ignore the factor k, we can not have a closed form solution, and as we see in the experimental results section, the step k is influential in the quality of predictions.

A.2 Convergence of Variance

Lemma 3 The mean of the results of different random walks will converge to a constant value.

Proof If we show the result of N^{th} random walk by r_N, and the mean of results up to N^{th} random walk as $\bar{r}_N$, then we will have:

$$\bar{r}_{N+1} = \frac{(N \times \bar{r}_N) + r_{N+1}}{N+1}$$

Since the result of each random walk is a rating, and ratings are in finite ranges, we will have:

$$\lim_{N\to\infty} \bar{r}_{N+1} - \bar{r}_N = \lim_{N\to\infty} \frac{(N \times \bar{r}_N) + r_{N+1}}{N+1} - \bar{r}_N = 0;$$

So, the average of ratings will converge to a constant $\bar{r}$. Note that if ratings are in infinite range, then we can not prove this, but typically the ratings are in finite range

Lemma 4 The variance in the results of random walk will converge to a constant value. In other words, the rate of change in variance will converge to zero.

Proof We prove that the variance of the results up to N random walks, denoted by σ_N, converges as follows:

$$\lim_{N\to\infty} \sigma_{N+1}^2 - \sigma_N^2 = \lim_{N\to\infty} \frac{\sum_{k=1}^{N+1} (r_k - \bar{r}_{N+1})^2}{N+1} - \frac{\sum_{k=1}^{N} (r_k - \bar{r}_N)^2}{N}$$
$$= \lim_{N\to\infty} \frac{N \times \sigma_N^2 + (r_{N+1} - \bar{r})^2}{N+1} - \sigma_N^2 = 0 \quad \text{(A.5)}$$

Appendix B

Posterior Inference and Parameter Estimation in GSBM

In this appendix, we provide the details of the learning phase for the GSBM[54] model introduced in Chapter 6. We use variational-EM to learn the parameters of the proposed model. The E-step is a variational inference on the latent variables. After learning the variational parameters for the latent variables, we perform the M-step similar to the conventional EM algorithm.

B.1 Variational Inference in GSBM

For each of the rating and social interactions processes, the likelihood is computed as follows:

$$p(T, Z_{\rightarrow}, Z_{\leftarrow} | B_T, \overrightarrow{\Pi}_{1:N}, \rho_T) = \prod_u \prod_v P(t_{u,v} | \overrightarrow{z}_{u\rightarrow v}, \overrightarrow{z}_{u\leftarrow v}, B_T, \rho_T) P(\overrightarrow{z}_{u\rightarrow v} | \overrightarrow{\Pi}_u) P(\overrightarrow{z}_{u\leftarrow v} | \overrightarrow{\Pi}_v) \quad \text{(B.1)}$$

and

$$p(R, Y, X_{\rightarrow}, X_{\leftarrow} | B_R, B_Y, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}, \rho_R) = \prod_u \prod_i p(\overrightarrow{R}_{u,i} | \overrightarrow{x}_{u\rightarrow i}, \overrightarrow{x}_{u\leftarrow i}, B_R, Y_{u,i}) p(Y_{u,i} | \overrightarrow{x}_{u\rightarrow i}, \overrightarrow{x}_{u\leftarrow i}, B_Y, \rho_R) \times p(\overrightarrow{x}_{u\rightarrow i} | \overrightarrow{\Pi}_u) p(\overrightarrow{x}_{u\leftarrow i} | \overrightarrow{\Delta}_i) \quad \text{(B.2)}$$

Now, the likelihood of the complete data is as follows:

$$\begin{aligned} p(T, R, Y, Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow}, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M} | \alpha, \beta, B_T, B_R, B_Y, \rho_T, \rho_R) \\ = p(T, Z_{\rightarrow}, Z_{\leftarrow} | B_T, \overrightarrow{\Pi}_{1:N}, \rho_T) p(R, Y, X_{\rightarrow}, X_{\leftarrow} | B_R, B_Y \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}, \rho_R) \\ \times \prod_u p(\overrightarrow{\Pi}_u | \overrightarrow{\alpha}) \prod_i p(\overrightarrow{\Delta}_i | \overrightarrow{\beta}) \end{aligned} \quad \text{(B.3)}$$

Using Jensen's inequality, we have:

$$\begin{aligned} \log p(T, R, Y, \alpha, \beta, B_T, B_R, B_Y, \rho_T, \rho_R) \geq \\ \mathbb{E}_q \big[\log p(T, R, Y, Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow}, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M} | \alpha, \beta, B_T, B_R, B_Y, \rho_T, \rho_R) \big] \\ - \mathbb{E}_q \big[\log q(Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow}, \overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}) \big] \end{aligned} \quad \text{(B.4)}$$

The right hand side of equation 4 is the lower bound for log likelihood and we denote it by $\mathcal{L}$ in the rest of this section.

We introduce a distribution of the latent variables q depending on a set of free variational parameters. We specify q in a fully-factorized way as follows:

$$\begin{aligned} q(\overrightarrow{\Pi}_{1:N}, \overrightarrow{\Delta}_{1:M}, Z_{\rightarrow}, Z_{\leftarrow}, X_{\rightarrow}, X_{\leftarrow} | \overrightarrow{\gamma}_{1:N}, \overrightarrow{\delta}_{1:M}, \Phi_{\rightarrow}, \Phi_{\leftarrow}, \Omega_{\rightarrow}, \Omega_{\leftarrow}) = \\ \prod_u q(\overrightarrow{\Pi}_u | \overrightarrow{\gamma}_u) \prod_u \prod_v q(\overrightarrow{z}_{u \rightarrow v} | \overrightarrow{\phi}_{u \rightarrow v}) q(\overrightarrow{z}_{u \leftarrow v} | \overrightarrow{\phi}_{u \leftarrow v}) \\ \prod_i q(\overrightarrow{\Delta}_i | \overrightarrow{\delta}_i) \prod_u \prod_i q(\overrightarrow{x}_{u \rightarrow i} | \overrightarrow{\omega}_{u \rightarrow i}) q(\overrightarrow{x}_{u \leftarrow i} | \overrightarrow{\omega}_{u \leftarrow i}) \end{aligned} \quad \text{(B.5)}$$

The lower bound for the log likelihood can be expanded as follows:

$$\begin{aligned}
\mathcal{L} = \mathbb{E}_q\Big[\sum_u\sum_v \ln p(t_{u,v}|\overrightarrow{z}_{u\to v}\overrightarrow{z}_{u\leftarrow v}, B_T, \rho_T)\Big] \\
+ \mathbb{E}_q\Big[\sum_u\sum_v \ln p(\overrightarrow{z}_{u\to v}|\overrightarrow{\Pi}_u)\Big] + \mathbb{E}_q\Big[\sum_u\sum_v \ln p(\overrightarrow{z}_{u\leftarrow v}|\overrightarrow{\Pi}_v)\Big] \\
+ \mathbb{E}_q\Big[\sum_u\sum_i \ln p(\overrightarrow{R_{u,i}}|\overrightarrow{x}_{u\to i}\overrightarrow{x}_{u\leftarrow i}, B_R)\Big] \\
+ \mathbb{E}_q\Big[\sum_u\sum_i \ln p(Y_{u,i}|\overrightarrow{x}_{u\to i}\overrightarrow{x}_{u\leftarrow i}, B_Y, \rho_R)\Big] \\
+ \mathbb{E}_q\Big[\sum_u\sum_i \ln p(\overrightarrow{x}_{u\to i}|\overrightarrow{\Pi}_u)\Big] + \mathbb{E}_q\Big[\sum_u\sum_i \ln p(\overrightarrow{x}_{u\leftarrow i}|\overrightarrow{\Delta}_i)\Big] \\
+ \mathbb{E}_q\Big[\sum_u \ln p(\overrightarrow{\Pi}_u|\overrightarrow{\alpha})\Big] + \mathbb{E}_q\Big[\sum_i \ln p(\overrightarrow{\Delta}_i|\overrightarrow{\beta})\Big] \\
- \mathbb{E}_q\Big[\sum_u \ln q(\overrightarrow{\Pi}_u|\overrightarrow{\gamma}_u)\Big] - \mathbb{E}_q\Big[\sum_u\sum_v \ln q(\overrightarrow{z}_{u\to v}|\overrightarrow{\phi}_{u\to v})\Big] - \mathbb{E}_q\Big[\sum_u\sum_v \ln q(\overrightarrow{z}_{u\leftarrow v}|\overrightarrow{\phi}_{u\leftarrow v})\Big] \\
- \mathbb{E}_q\Big[\sum_i \ln q(\overrightarrow{\Delta}_i|\overrightarrow{\delta}_i)\Big] - \mathbb{E}_q\Big[\sum_u\sum_i \ln q(\overrightarrow{x}_{u\to i}|\overrightarrow{\omega}_{u\to i})\Big] - \mathbb{E}_q\Big[\sum_u\sum_i \ln q(\overrightarrow{x}_{u\leftarrow i}|\overrightarrow{\omega}_{u\leftarrow i})\Big]
\end{aligned} \tag{B.6}$$

Now, we compute the details of each segment of the lower bound one by one in the following equations.

$$\begin{aligned}
&\mathbb{E}_q\Big[\sum_u\sum_v \ln p(t_{u,v}|\overrightarrow{z}_{u\to v}\overrightarrow{z}_{u\leftarrow v}, B_T, \rho_T)\Big] = \\
&\mathbb{E}_q\Big[\sum_u\sum_v\sum_{i=1}^{K_1}\sum_{j=1}^{K_1}\big[z_{u\to v_i}z_{u\leftarrow v_j}\big(t_{u,v}\ln(1-\rho_T)B_{T_{i,j}} + (1-t_{u,v})\ln(1-(1-\rho_T)B_{T_{i,j}})\big)\big]\Big] = \\
&\sum_u\sum_v\sum_{i=1}^{K_1}\sum_{j=1}^{K_1}\big[\phi_{u\to v_i}\phi_{u\leftarrow v_j}\big(t_{u,v}\ln(1-\rho_T)B_{T_{i,j}} + (1-t_{u,v})\ln(1-(1-\rho_T)B_{T_{i,j}})\big)\big]
\end{aligned} \tag{B.7}$$

Similarly, we have

$$\begin{aligned}
&\mathbb{E}_q\Big[\sum_u\sum_i \ln p(Y_{u,i}|\overrightarrow{x}_{u\to i}\overrightarrow{x}_{u\leftarrow i}, B_Y, \rho_R)\Big] = \\
&\sum_u\sum_i\sum_{m=1}^{K_1}\sum_{n=1}^{K_2}\big[\omega_{u\to i_m}\omega_{u\leftarrow i_n}\big(Y_{u,i}\ln(1-\rho_R)B_{Y_{m,n}} + (1-Y_{u,i})\ln(1-(1-\rho_R)B_{Y_{m,n}})\big)\big]
\end{aligned} \tag{B.8}$$

The expected values of membership indicators are computed as follows:

$$\mathbb{E}_q\Big[\sum_u\sum_v \ln p(\overrightarrow{z}_{u\to v}|\overrightarrow{\Pi}_u)\Big] = \mathbb{E}_q\Big[\sum_u\sum_v \ln\Big(\prod_{l=1}^{K_1}\Pi_{u,l}^{z_{u\to v_l}}\Big)\Big]$$
$$= \mathbb{E}_q\Big[\sum_u\sum_v\sum_{l=1}^{K_1}\big(z_{u\to v_l}\ln\Pi_{u,l}\big)\Big] = \sum_u\sum_v\sum_{l=1}^{K_1}\Big[\phi_{u\to v_l}\big(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t})\big)\Big] \quad \text{(B.9)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_v \ln p(\overrightarrow{z}_{u\leftarrow v}|\overrightarrow{\Pi}_v)\Big] = \sum_u\sum_v\sum_{l=1}^{K_1}\Big[\phi_{u\leftarrow v_l}\big(\psi(\gamma_{v,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{v,t})\big)\Big] \quad \text{(B.10)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_i \ln p(\overrightarrow{x}_{u\to i}|\overrightarrow{\Pi}_u)\Big] = \sum_u\sum_i\sum_{l=1}^{K_1}\Big[\omega_{u\to i_l}\big(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t})\big)\Big] \quad \text{(B.11)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_i \ln p(\overrightarrow{x}_{u\leftarrow i}|\overrightarrow{\Delta}_i)\Big] = \sum_u\sum_i\sum_{l=1}^{K_2}\Big[\omega_{u\leftarrow i_l}\big(\psi(\delta_{i,l}) - \psi(\sum_{t=1}^{K_2}\delta_{i,t})\big)\Big] \quad \text{(B.12)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_i \ln p(R_{u,i}|\overrightarrow{x}_{u\to i}\overrightarrow{x}_{u\leftarrow i}, B_R, Y_{u,i})\Big] =$$
$$\mathbb{E}_q\Bigg[\sum_u\sum_i \ln\Big[\Big(\prod_{m=1}^{K_1}\prod_{n=1}^{K_2}\big[\prod_{l=1}^{5}(B_{R_{m,n,l}})^{R_{u,i_l}}\big]^{x_{u\to i_m}x_{u\leftarrow i_n}}\Big)^{Y_{u,i}}\Big]\Bigg]$$
$$= \mathbb{E}_q\Bigg[\sum_u\sum_i\Big[Y_{u,i}\sum_{m=1}^{K_1}\sum_{n=1}^{K_2}\big[x_{u\to i_m}x_{u\leftarrow i,n}\big(\sum_{l=1}^{5}R_{u,i,l}\ln B_{R_{m,n,l}}\big)\big]\Big]\Bigg]$$
$$= \sum_u\sum_i\Bigg[Y_{u,i}\sum_{m=1}^{K_1}\sum_{n=1}^{K_2}\Big[\omega_{u\to i_m}\omega_{u\leftarrow i_n}\big(\sum_{l=1}^{5}R_{u,i,l}\ln B_{R_{m,n,l}}\big)\Big]\Bigg] \quad \text{(B.13)}$$

$$\mathbb{E}_q\Big[\sum_u \ln p(\overrightarrow{\Pi}_u|\overrightarrow{\alpha})\Big] =$$
$$\sum_u\Big[\ln\Gamma(\sum_{l=1}^{K_1}\alpha_l) - \sum_{l=1}^{K_1}\ln\Gamma(\alpha_l) + \sum_{l=1}^{K_1}\big[(\alpha_l-1)\big(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t})\big)\big]\Big] \quad \text{(B.14)}$$

$$\mathbb{E}_q\Big[\sum_i \ln p(\overrightarrow{\Delta}_i|\overrightarrow{\beta})\Big] = \sum_i \Big[\ln\Gamma(\sum_{l=1}^{K_2}\beta_l) - \sum_{l=1}^{K_2}\ln\Gamma(\beta_l) + \sum_{l=1}^{K_2}\big[(\beta_l-1)\big(\psi(\delta_{u,l}) - \psi(\sum_{t=1}^{K_2}\delta_{u,t})\big)\big]\Big] \quad \text{(B.15)}$$

$$\mathbb{E}_q\Big[\sum_u \ln q(\overrightarrow{\Pi}_u|\overrightarrow{\gamma}_u)\Big] = \sum_u \Big[\ln\Gamma(\sum_{l=1}^{K_1}\gamma_{u,l}) - \sum_{l=1}^{K_1}\ln\Gamma(\gamma_{u,l}) + \sum_{l=1}^{K_1}\big[(\gamma_{u,l}-1)\big(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t})\big)\big]\Big] \quad \text{(B.16)}$$

$$\mathbb{E}_q\Big[\sum_i \ln q(\overrightarrow{\Delta}_i|\overrightarrow{\delta}_i)\Big] = \sum_i \Big[\ln\Gamma(\sum_{l=1}^{K_2}\delta_{i,l}) - \sum_{l=1}^{K_2}\ln\Gamma(\delta_{i,l}) + \sum_{l=1}^{K_2}\big[(\delta_{i,l}-1)\big(\psi(\delta_{i,l}) - \psi(\sum_{t=1}^{K_2}\delta_{i,t})\big)\big]\Big] \quad \text{(B.17)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_v \ln q(\overrightarrow{z}_{u\to v}|\overrightarrow{\phi}_{u\to v})\Big] =$$

$$\mathbb{E}_q\Big[\sum_u\sum_v \ln\prod_{l=1}^{K_1}(\phi_{u\to v_l})^{z_{u\to v_l}}\Big] = \mathbb{E}_q\Big[\sum_u\sum_v\sum_{l=1}^{K_1}\big(z_{u\to v_l}\ln\phi_{u\to v_l}\big)\Big]$$

$$= \sum_u\sum_v\sum_{l=1}^{K_1}\big(\phi_{u\to v_l}\ln\phi_{u\to v_l}\big) \quad \text{(B.18)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_v \ln q(\overrightarrow{z}_{u\leftarrow v}|\overrightarrow{\phi}_{u\leftarrow v})\Big] = \sum_u\sum_v\sum_{l=1}^{K_1}\big(\phi_{u\leftarrow v_l}\ln\phi_{u\leftarrow v_l}\big) \quad \text{(B.19)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_i \ln q(\overrightarrow{x}_{u\to i}|\overrightarrow{\omega}_{u\to i})\Big] = \sum_u\sum_i\sum_{l=1}^{K_2}\big(\omega_{u\to i_l}\ln\omega_{u\to i_l}\big) \quad \text{(B.20)}$$

$$\mathbb{E}_q\Big[\sum_u\sum_i \ln q(\overrightarrow{x}_{u\leftarrow i}|\overrightarrow{\omega}_{u\leftarrow i})\Big] = \sum_u\sum_i\sum_{l=1}^{K_2}\big(\omega_{u\leftarrow i_l}\ln\omega_{u\leftarrow i_l}\big) \quad \text{(B.21)}$$

To find the optimum variational parameters, we compute the derivatives of the lower bound function with respect to all variational parameters and set them to zero.

$$\frac{\partial \mathcal{L}}{\partial \phi_{u\to v_l}} = \sum_{t=1}^{K_1} \left[\phi_{u\leftarrow v_t}\left(t_{u,v}\ln\left((1-\rho_T)B_{T_{l,t}}\right) + (1-t_{u,v})\ln(1-(1-\rho_T)B_{T_{l,t}})\right)\right] - \ln\phi_{u\to v_l} + (\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t}) \quad \text{(B.22)}$$

Setting the derivative to zero, we have:

$$\phi_{u\to v_l} \propto \exp\left(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t})\cdot\prod_{t=1}^{K_1}\left[((1-\rho_T)B_{T_{l,t}})^{t_{u,v}\phi_{u\leftarrow v_t}}(1-(1-\rho_T)B_{T_{l,t}})^{(1-t_{u,v})\phi_{u\leftarrow v_t}}\right]\right. \quad \text{(B.23)}$$

Similarly,

$$\phi_{u\leftarrow v_l} \propto \exp\left(\psi(\gamma_{v,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{v,t})\cdot\prod_{t=1}^{K_1}\left[((1-\rho_T)B_{T_{l,t}})^{t_{u,v}\phi_{u\to v_t}}(1-(1-\rho_T)B_{T_{l,t}})^{(1-t_{u,v})\phi_{u\to v_t}}\right]\right. \quad \text{(B.24)}$$

For variational parameters creating a rating, we have:

$$\frac{\partial \mathcal{L}}{\partial \omega_{u\to i_l}} = \sum_{n=1}^{K_2}\left[\omega_{u\leftarrow i_n}\left(\sum_{t=1}^{5} R_{u,i,t}\ln B_{R_{l,n,t}}\right)\right] + \sum_{n=1}^{K_2}\left[\omega_{u\leftarrow i_n}\left(Y_{u,i}\ln(1-\rho_R)B_{Y_{m,n}} + (1-Y_{u,i})\ln(1-(1-\rho_R)B_{Y_{m,n}})\right)\right] - \ln\omega_{u\to i_l} + (\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_1}\gamma_{u,t}) \quad \text{(B.25)}$$

Hence,

$$\omega_{u\to i_l} \propto \exp\left(\psi(\gamma_{u,l}) - \psi(\sum_{t=1}^{K_2}\gamma_{u,t})\cdot\prod_{n=1}^{K_2}\left[\prod_{t=1}^{5} {B_{R_{l,n,t}}}^{R_{u,i,t}}\right]^{\omega_{u\leftarrow i_n}}\right. \times \prod_{n=1}^{K_2}\left[((1-\rho_R)B_{Y_{l,n}})^{Y_{u,i}\omega_{u\leftarrow i_n}}(1-(1-\rho_R)B_{Y_{l,n}})^{(1-Y_{u,i})\omega_{u\leftarrow i_n}}\right] \quad \text{(B.26)}$$

Similarly,

$$\omega_{u\leftarrow i_l} \propto \exp\left(\psi(\delta_{i,l}) - \psi(\sum_{t=1}^{K_2}\delta_{i,t})\cdot \prod_{m=1}^{K_1}\left[\prod_{t=1}^{5} B_{R_{m,l,t}}{}^{R_{u,i,t}}\right]^{\omega_{u\to i_m}}\right.$$
$$\left.\times \prod_{m=1}^{K_1}\left[((1-\rho_R)B_{Y_{l,m}})^{Y_{u,i}\omega_{u\to i_m}}(1-(1-\rho_R)B_{Y_{l,m}})^{(1-Y_{u,i})\omega_{u\to i_m}}\right]\right. \quad \text{(B.27)}$$

The priors for mixed membership assignments can also be computed as follows:

$$\frac{\partial \mathcal{L}}{\partial \gamma_{u,l}} = \psi'(\gamma_{u,l})\Big[\alpha_l - \gamma_{u,l} + \sum_v \phi_{u\to v_l} + \sum_v \phi_{v\leftarrow u_l} + \sum_i \omega_{u\to i_l}\Big]$$
$$- \big[\psi'(\sum_{s=1}^{K_1}\gamma_{u,s})\big]\Big[\sum_{t=1}^{K_1}\Big(\alpha_t - \gamma_{u,t} + \sum_v \phi_{u\to v_t} + \sum_v \phi_{v\leftarrow u_t} + \sum_i \omega_{u\to i_t}\Big)\Big] \quad \text{(B.28)}$$

Therefore,

$$\gamma_{u,l} = \alpha_l + \sum_v \phi_{u\to v_l} + \sum_v \phi_{v\leftarrow u_l} + \sum_i \omega_{u\to i_l} \quad \text{(B.29)}$$

$$\frac{\partial \mathcal{L}}{\partial \delta_{i,l}} = \psi'(\delta_{i,l})\Big[\beta_l - \delta_{i,l} + \sum_u \omega_{u\leftarrow i_l}\Big]$$
$$- \big[\psi'(\sum_{s=1}^{K_2}\delta_{i,s})\big]\Big[\sum_{t=1}^{K_2}\Big(\beta_t - \delta_{i,t} + \sum_u \omega_{u\leftarrow i_t}\Big)\Big] \quad \text{(B.30)}$$

Therefore,

$$\delta_{i,l} = \beta_l + \sum_u \omega_{u\leftarrow i_l} \quad \text{(B.31)}$$

B.2 Parameter Estimation in GSBM

We compute the empirical Bayes estimates of the model hyper parameters $\{\overrightarrow{\alpha}, \overrightarrow{\beta}, B_T, B_R, \rho_T, \rho_R\}$ with a variational expectation-maximization (EM) algorithm.

A closed form solution for the approximate maximum likelihood estimate of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ does not exist [3]. We use a linear time Newton-Raphson method, where the gradient and Hessian are

$$\frac{\partial \mathcal{L}}{\partial \alpha_l} = N \times \Big(\Psi(\sum_{t=1}^{K_1} \alpha_t) - \Psi(\alpha_l)\Big) + \sum_u \Big(\Psi(\gamma_{u,l}) - \Psi(\sum_{t=1}^{K_1} \gamma_{u,t})\Big) \tag{B.32}$$

$$\frac{\partial^2 \mathcal{L}}{\partial \alpha_l \partial \alpha_t} = N \times \Big(\Psi'(\sum_{s=1}^{K_1} \alpha_s) - \mathbb{I}_{(l=t)}.\Psi'(\alpha_l)\Big) \tag{B.33}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_l} = M \times \Big(\Psi(\sum_{t=1}^{K_2} \beta_t) - \Psi(\beta_l)\Big) + \sum_i \Big(\Psi(\delta_{i,l}) - \Psi(\sum_{t=1}^{K_2} \delta_{i,t})\Big) \tag{B.34}$$

$$\frac{\partial^2 \mathcal{L}}{\partial \beta_l \partial \beta_t} = M \times \Big(\Psi'(\sum_{s=1}^{K_2} \beta_s) - \mathbb{I}_{(l=t)}.\Psi'(\beta_l)\Big) \tag{B.35}$$

The approximate MLE of B_T and B_R are:

$$B_{T_{i,j}} = \frac{\sum_u \sum_v t_{u,v} \phi_{u \to v_i} \phi_{u \leftarrow v_j}}{(1-\rho_T) \sum_u \sum_v \phi_{u \to v_i} \phi_{u \leftarrow v_j}} \tag{B.36}$$

$$B_{Y_{m,n}} = \frac{\sum_u \sum_i Y_{u,i} \omega_{u \to i_m} \omega_{u \leftarrow i_n}}{(1-\rho_R) \sum_u \sum_i \omega_{u \to i_m} \phi_{u \leftarrow i_n}} \tag{B.37}$$

$$B_{R_{m,n,l}} = \frac{\sum_u \sum_i Y_{u,i} (\omega_{u \to i_m} \omega_{u \leftarrow i_n} R_{u,i,l})}{\sum_u \sum_i Y_{u,i} (\omega_{u \to i_m} \omega_{u \leftarrow i_n})} \tag{B.38}$$

Also, for the sparsity parameter, we have:

$$1 - \rho_T = \frac{\sum_u \sum_v \sum_i \sum_j \phi_{u \to v_i} \phi_{u \leftarrow v_j} t_{u,v}}{\sum_u \sum_v \sum_i \sum_j \phi_{u \to v_i} \phi_{u \leftarrow v_j} B_{T_{i,j}}} \tag{B.39}$$

$$1 - \rho_R = \frac{\sum_u \sum_i \sum_m \sum_n \omega_{u \to i_m} \omega_{u \leftarrow i_n} Y_{u,i}}{\sum_u \sum_i \sum_m \sum_n \omega_{u \to i_n} \omega_{u \leftarrow i_n} B_{Y_{m,n}}} \tag{B.40}$$

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–492, 1994.

[3] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

[4] Aris Anagnostopoulos, Ravi Kumar, and Mohammad Mahdian. Influence and correlatino in social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 7–15, 2008.

[5] Reid Andersen, Christian Borgs, Jennifer Chayes, Uriel Feige, Abraham Flaxman, Adam Kalai, Vahab Mirrokni, and Moshe Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *WWW'08: 17th International World Wide Web Conference*, pages 199–208, 2008.

[6] Yves Atchade, Gersende Fort, Eric Moulines, and Pierre Priouret. *Adaptive Markov chain Monte Carlo : Theory and Methods*, chapter Bayesian Time Series Models, pages 32–51. Cambridge University Press, 2011.

[7] Paolo Avesani, Paolo Massa, and Roberto Tiella. Moleskiing.it: a trust-aware recommender system for ski mountaineering. *International Journal for Infonomics*, 2005.

[8] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD'06*, pages 44–54, 2006.

[9] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1st edition, 1999.

[10] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[11] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of ACM*, 40(3):66–72, 1997.

[12] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: using social and content-based information in recommendation. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 714–720, 1998.

[13] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: two sides of the same coin? *Communications of ACM*, 35(12):29–38, 1992.

[14] Robert M. Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD'07: The 13th ACM SIGKDD conference on Knowledge Discovery andData Mining*, pages 95–104, 2007.

[15] Julian Besag and Peter Clifford. Generalized monte carlo significance tests. *Biometrika*, 76(4):633–642, 1989.

[16] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, 1998.

[17] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2 edition, 2007.

[18] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[19] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Annual Conference on Uncertainty in Artificial Intelligence (UAI98)*, pages 43–52, 1998.

[20] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998.

[21] Vincent Buskens. *Social Networks and Trust*. Springer, 2002.

[22] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[23] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th Annual World Wide Web Conference (WWW'09)*, pages 721–730, April 2009.

[24] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.

[25] James Coleman. *Foundations of Social Theory*. Harvard University Press, 1990.

[26] David Crandall, Dan Cosley, Daniel Huttenlocher, Jon Kleinberg, and Siddharth Suri. Feedback effects between similarity and social influence in online communities. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 160–168, 2008.

[27] Easley David and Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[28] Joaquin Delgado and Naohiro Ishii. Memory-based weighted-majority prediction for recommender systems. In *ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[29] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[30] Mukund Deshpande and George Karypis. Item based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, 2004.

[31] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[32] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(3):355–369, 2007.

[33] Noah E. Friedkin. *A Structural Theory of Social Influence*. Cambridge University Press, 1998.

[34] Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(3):1–32, 2007.

[35] Stephan Gnnemann, Phuong Dao, Mohsen Jamali, and Martin Ester. Assessing the significance of data mining results on graphs with feature vectors. In *Submitted to the International Conference on Data Mining*, ICDM'12, 2012.

[36] Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland College Park, 2005.

[37] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[38] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW'04: 13th International World Wide Web Conference*, pages 403–412, 2004.

[39] Sami Hanhijarvi, Gemma C. Garriga, and Kai Puolamaki. In *Proceedings of the 9th SIAM International Conference on Data Mining*, SDM'09.

[40] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, 2000.

[41] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[42] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, 1995.

[43] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, 2003.

[44] Paul Holland and Samuel Leinhardt. Some evidence on the transitivity of positive interpersonal sentiment. *American Journal of Sociology*, pages 1205–1209, 1972.

[45] Petter Holme and Mark E.J. Newman. Nonequilibrium phase transition in the coevolution of networks and opinions. *Physical Review E*, 74, 2006.

[46] David R. Hunter, Steven M. Goodreau, and Mark S. Handcock. Goodness of fit of social network models. *Journal of the American Statistical Association*, 103(481):248–258, 2008.

[47] Mohsen Jamali and Martin Ester. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 397–406, 2009.

[48] Mohsen Jamali and Martin Ester. Using a trust network to improve top-n recommendation. In *Proceedings of the third ACM conference on Recommender systems*, RecSys'09, pages 181–188, 2009.

[49] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 135–142, 2010.

[50] Mohsen Jamali and Martin Ester. A transitivity aware matrix factorization model for recommendation in social networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI'11, pages 2644–2649, 2011.

[51] Mohsen Jamali and Martin Ester. Random walk models for combining social network-based and similarity-based recommendation. *Submitted to IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2012.

[52] Mohsen Jamali and Martin Ester. Regularized matrix factorization for cold start recommendation. *Submitted to IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2012.

[53] Mohsen Jamali, Gholamreza Haffari, and Martin Ester. Modeling the temporal dynamics of social rating networks using bidirectional effects of social relations and rating patterns. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 527–536, 2011.

[54] Mohsen Jamali, Tianle Huang, and Martin Ester. A generalized stochastic block model for recommendation in social rating networks. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys'11, pages 53–60, 2011.

[55] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 79–86, 2010.

[56] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254, New York, NY, USA, 2001.

[57] Heung-Nam Kim, Ae-Ttie Ji, Hyun-Jun Kim, and Geun-Sik Jo. Error-based collaborative filtering algorithm for top-n recommendation. In *The Joint International Conferences on Asia-Pacific Web Conference and Web-Age Information Management (APWeb/WAIM)*, pages 594–605, Huang Shan, China, June 2007.

[58] Yehuda Koren. Factorization meets the neighborhood a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, 2008.

[59] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 447–456, 2009.

[60] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42:30–37, 2009.

[61] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 611–617, 2006.

[62] YoungOk Kwon. Improving top-n recommendation techniques using rating variance. In *RecSys'08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 307–310, 2008.

[63] Ken Lang. Newsweeder: Learning to filter netnews. In *12th International Conference on Machine Learning (ICML95)*, pages 331–339, 1995.

[64] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal Matrix Analyis and Applications*, 21(4):1253–1278, 2000.

[65] Paul Lazarsfeld and Robert Merton. Friendship as a social process: A substantive and methodological analysis. In M. Bergen, T. Abel, and C. Page, editors, *Freedom and Control in Modern Society*, pages 18–66. Van Nostrand, 1954.

[66] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 462–470, 2008.

[67] Levien and Aiken. Advogato's trust metric. online at http://advogato.org/trust-metric.html, 2002.

[68] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7:76–80, 2003.

[69] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR'09, pages 203–210, 2009.

[70] Hao Ma, Michael R. Lyu, and Irwin King. Learning to recommend with trust and distrust relationships. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 189–196, 2009.

[71] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 931–940, 2008.

[72] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 287–296, 2011.

[73] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *Seventeenth Annual Conference on Neural Information Processing Systems*, NIPS'03.

[74] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 17–24, 2007.

[75] Paolo Massa and Paolo Avesani. Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(1), 2007.

[76] Yutaka Matsuo and Hikaru Yamamoto. Community gravity: measuring bidirectional effects by trust and rating on online social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 751–760, 2009.

[77] Matthew R. McLaughlin and Jonathan L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th international ACM SIGIR conference on Information Retrieval*, SIGIR'04, pages 329–336, 2004.

[78] Miller Mcpherson, Lynn S. Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[79] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[80] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, 2003.

[81] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, 2007.

[82] Samaneh Moghaddam, Mohsen Jamali, and Martin Ester. Etf: extended tensor factorization model for personalizing prediction of review helpfulness. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 163–172, 2012.

[83] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, 2000.

[84] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In *JSAI'07: Proceedings of the 2007 conference on New frontiers in artificial intelligence*, pages 40–46, 2008.

[85] Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, 1998.

[86] Mark Newman, Albert-Laszlo Barabasi, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

[87] Mark E.J. Newman. Power laws, pareto distributions and zipfs law. *Contemporary Physics*, 46(5):323–351, 2005.

[88] John O'Donovan and Barry Smyth. Trust in recommender systems. In *IUI'05: 10th international conference on Intelligent user interfaces*, pages 167–174, San Diego, USA, 2005.

[89] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 653–658. ACM, 2004.

[90] Dmitry Y. Pavlov and David M. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *In Proceedings of Neural Information Processing Systems*, 2002.

[91] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification ofinteresting web sites. *Machine Learning*, 27(3):313–331, 1997.

[92] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[93] Clayton C. Peddy and Derek Armentrout. *Building Solutions with Microsoft Commerce Server 2002*. Microsoft Press, Redmond, WA, USA, 2003.

[94] Anatol Rapoport. Mathematical models of social interaction. In D. Luce D, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume 2. Wiley, New York, USA, 1963.

[95] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 727–736, 2009.

[96] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: ACM conference on Computer supported cooperative work*, 1994.

[97] Achim Rettinger, Matthias Nickles, and Volker Tresp. A statistical relational model for trust learning. In *AAMAS'08: 7th international joint conference on Autonomous agents and multiagent systems*, pages 763–770, 2008.

[98] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 61–70, 2002.

[99] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Twenty-First Annual Conference on Neural Information Processing Systems*, NIPS'07, 2007.

[100] J. J. Sandvig, Bamshad Mobasher, and Robin Burke. Robustness of collaborative recommendation based on association rule mining. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 105–112, 2007.

[101] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW'01: 10th International World Wide Web Conference*, pages 285–295, 2001.

[102] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international*

ACM SIGIR conference on Research and development in information retrieval, SIGIR '02, pages 253–260, 2002.

[103] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

[104] Tom A.B. Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100, 1997.

[105] Ian Soboroff, Charles Nicholas, and Charles K. Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI99 Workshop on Machine Learning for Information Filtering*, pages 86–91, 1999.

[106] Mohammad A. Tayebi, Mohsen Jamali, Martin Ester, Uwe Glässer, and Richard Frank. Crimewalker: a recommendation model for suspect investigation. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys'11, pages 173–180, 2011.

[107] Nava Tintarev. Explaining recommendations. In Cristina Conati, Kathleen McCoy, and Georgios Paliouras, editors, *User Modeling 2007*, volume 4511 of *Lecture Notes in Computer Science*, pages 470–474. Springer Berlin / Heidelberg, 2009.

[108] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 613–622, Washington, DC, USA, 2006.

[109] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*, 1998.

[110] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 47–56, 2009.

[111] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer Science, 2005.

[112] Frank Edward Walter, Stefano Battiston, and Frank Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[113] Chong Wang and David Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 448–456, 2011.

[114] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 537–546, 2011.

[115] Hilmi Yildirim and Mukkai S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys'08: ACM Conference on Recommender Systems*, pages 131–138, Switzerland, 2008.

[116] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. A unified framework for link recommendation using random walks. In *Proceedings of AONAM'10, the 2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 152–159, 2010.

[117] Cong Yu, Laks V. S. Lakshmanan, and Sihem Amer-Yahia. Recommendation diversification using explanations. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1299–1302, 2009.

[118] Elena Zheleva, Hossam Sharara, and Lise Getoor. Co-evolution of social and affiliation networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1007–1016, 2009.

[119] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '11, pages 315–324, 2011.

[120] Cai Nicolas Ziegler. *Towards Decentralized Recommender Systems*. PhD thesis, University of Freiburg, 2005.

[121] Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.

[122] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 22–32, 2005.

Index