Recommendation with contextual information

A Thesis

Submitted to the Faculty

of

Drexel University

by

Jia Huang

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

May 2014

Acknowledgements

First of all, I hope to express my sincere gratitude and appreciation to my supervisor, Prof. Xiaohua (Tony) Hu for his guidance and support over the past five years till the completion of this thesis. Under his guidance, I have not only gained much on domain knowledge and research methods, but also learnt from him a positive attitude towards research and life. I will always be grateful for his encouragement and support, and will keep such enthusiasm towards my future career and life.

My special thanks go to Dr. Christopher Yang. I am grateful for his supervision and valuable advice at the initial stage of my PhD study and research.

I am grateful for my thesis committee members, Prof. Yuan An, Prof. Weimao Ke Prof. Lisa Ulmer, and Prof. Erjia Yan, for their helpful comments and suggestions throughout the process from my proposal towards the completion of this thesis. Dr. Jiexun Li also deserves my thanks for his suggestions and comments on my coursework and thesis proposal. Thanks to Prof. Michael Atwood, Prof. Gregory W. Hislop, Prof. Katherine W. McCain, Prof. Michelle L. Rogers, and Prof. Susan Wiedenbeck for their efforts in getting me access to research methods and projects.

It is my honor to thank Xin Chen, Caimei Lu and Lifan Guo, who were senior lab mates under the supervision of Prof Tony Hu. I have received so much help from them on research, career development and life. I will give my special thanks to Caimei Lu, especially in the first two years of my PhD life. From there, I gradually picked up programming skills and research experiences. I would also like to thank Hao Ding, Wanying Ding, Xuemei Gong, Yue Shang, Xiaoli Song, and Mi Zhang for their efforts and constructive discussions in conducting the research work in this thesis. I want to thank

Zunyan Xiong, my three-year roommate. It is a nice journey to have you as a companion for our PhD lives. I thank my labmates in CVDI, Xingpeng Jiang, Mengwen Liu, Weiwei Xu and Ling Yuan. It is a nice experience to have weekly discussions with you.

I also appreciate the help from my friends I met at our College at Drexel. I would like to thank my classmates Thomas Heverin, Emad Khazraee, Diana Kusunoki, Adam Townes, Heather Willever-Farr, and Michael Zarro. I will always enjoy our course discussions and social activates at Landmark. My thanks to fellow students Zhan Zhang, Lu Xiao, Haodong Yang, Ling Jiang and many others. I would thank all of you again for all the happiness we share in our PhD lives.

Last but not least, I want to thank my parents. They always support me whenever I face difficulties in my life. Without their warm love, this thesis would never have been completed.

Table of Contents

List of Tables

List of Figures

Abstract

Recommendation with contextual information

Jia Huang

Advisor: Xiaohua Tony Hu, Ph.D.

Information retrieval (IR) systems have tremendously broaden users' access to information. However, users need to select their needs from trillions of information indexed daily. Due to the "semantic gap" between queries and indexed terms in IR system, whether users can satisfy their needs depends on whether they use the correct terms as queries. Recommender systems, have become a counterpart of information retrieval systems such as Google. They do not require users to specify their information needs in advance. They model users' preferences based on their history data, and automatically recommend items which satisfy their needs. In this way, both semantic gap and information overload can be alleviated. Collaborative filtering is the most popular recommendation algorithm used in academia and industry. However, it suffers from the cold start problem, where it cannot recommend new items to users or give recommendation to new users. Its recommendations also bias towards popular items.

The goal of this thesis is to develop recommendation algorithms which can solve the item-side and user-side cold-start problems. We incorporate item content to solve the item-side cold start problem, and incorporate user similarity networks or social networks to solve the user-side cold-start problem. Our contributions are listed as follows. First, we develop a content-based movie recommendation algorithm which incorporates user, actor, and movie into the same model. Second, we develop a content-based music recommendation algorithm. The core component is a hierarchical dynamic programming

algorithm to compute music similarities. The resulting music similarity matrix is then employed to make recommendations. Third, we apply social recommendation algorithms on Douban dataset. We look into a "Twitter-like" user-follower social network to see if it can improve recommendation performance. Finally, we validate the existence of social influence. We also analyze the social influence by investigating the correlation between one's influence and centrality, and by investigating how recommendation effectiveness changes with the number of recommendations.

Chapter 1. Introduction

1.1 Overview

Recommender systems, have become a counterpart of information retrieval systems such as Google (Figure 1.1). In traditional information retrieval systems, users first enter queries into search engines, and filter relevant ones from the returned results. However, it is difficult for users to form queries due to the semantic gap between queries and the indexing terms used by IR systems. Besides, information overload makes it impossible for users to select the information they need. Recommender systems do not require users to specify their information needs in advance. They model users' preferences based on their history data, and automatically recommend items which satisfy their needs. In this way, both semantic gap and information overload can be alleviated. Compared to search engines, recommender systems are especially useful when users do not have a clear goal [123].

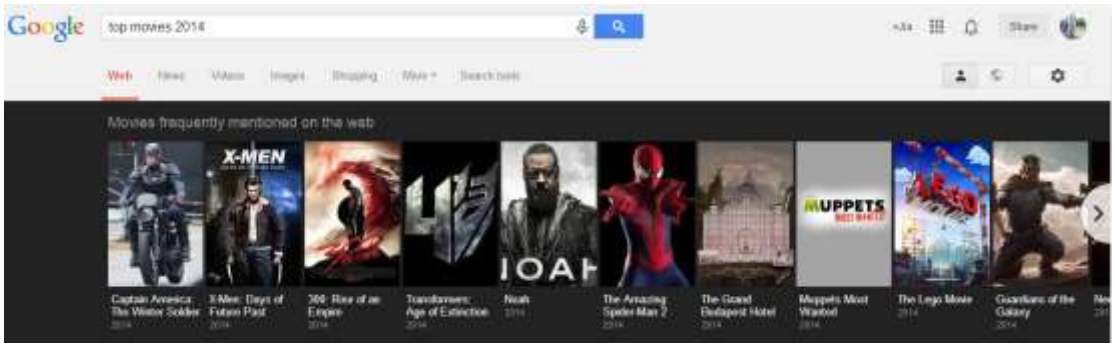


Figure 1.1 Movie "recommendation" in Google

Recommender system appears as an "independent research problem" in the mid of 1990s [2]. The GroupLens research group in Minnesota University was one of the pioneers in this area. They developed and studied MovieLens, a movie recommendation system [93].

The recommendation problem was defined as "predicting ratings of unseen items of a user given observed ratings and making recommendations based on the ratings".

In general, recommendation algorithms are classified into Collaborative Filtering (CF) and Content-based filtering [2]. Content-based filtering assumes that users prefer items which have similar content with those they liked before. For instance, a book's content information can include main text and metadata such as author, publisher, publication date, price and description. Collaborative filtering (CF) assumes that users who have similar preferences in the past will share interests in the future. Given a user, it recommends items preferred by similar users. Compared to content-based filtering, collaborative filtering is more widely used since it does not need item content. Collaborative filtering can be further divided into memory-based approach and model-based approach [17]. Memory-based CF algorithms compute user or item similarities and perform recommendations based on these similarities. Due to its simplicity, memory-based approach is adopted by real systems [93][74]. Model-based algorithms train a model based on the user-item ratings [54][55][56][106][124][128][99]. Algorithms including clustering model [124], Bayesian model [128] and aspect model [56] fall into this category.

Although collaborative filtering has become popular in both academia and industry, it suffers from the cold-start problem. The cold-start problem has two variations, item-side cold-start and user-side cold-start. Item-side cold-start denotes that CF cannot recommend new items to users. User-side cold-start denotes that CF is not capable to give recommendation to new users. Besides, most model-based CF performs poorly on users with few ratings [54][55][56].

Content-based filtering is the solution to item-side data sparseness and cold-start

problem. Graphical models and topic models have achieved their success in modeling text and recommending textual items such as document or paper [90][119]. However, for items without text or whose textual contents are difficult to obtain, we see few content-based models applied to recommending them. In [102], Schein et al. applied the two-way aspect model to movie recommendation by incorporating director and actor information. Actors were included into the model based on the assumption that casts of actors can be used as "surrogates" for movies. In music recommendation, via employing acoustic content features such as timbre and rhythm [59], metadata annotated by experts (e.g. Pandora[1]) or user created tags [117], content-based approach can compute similarities between any music pairs [23]. Given a user, when a new song enters into the system, one can compute the similarities between this song and those in her past collection, and recommend those with largest similarities.

Social recommendation incorporates users' social connections to solving the user-side data sparseness and cold start problem. User networks can be formed by people who share similar demographic information [58], people who are trustworthy [78], or real life friends [79]. It is based on the intuition that users' preferences are affected by those of their social connections. User profiles can be used to compute their demographic similarity [58]. However, such information was difficult to obtain due to privacy issues. Algorithms which incorporated trust networks have shown improvement over CF [78]. Moreover, social recommendation is useful because recommendations generated by recommender systems "cannot completely substitute personalized recommendations that people receive from friends" [70].

[1] Pandora.com. http://www.pandora.com/

1.2 Limitations and Open Problems

1.2.1Topic-model based movie recommendation

Under the problem of movie recommendation, content can include movies' metadata, description and even reviews. Since movie description and reviews may not be available and need to be preprocessed, metadata is the most easily accessible content information. Among metadata, casting information is an important factor in helping people decide which movie to watch, and how they rate them. Although [52] has included actor or director into their two-way aspect model, actor and movie do not exist in the same model.

1.2.2 Hierarchical Music Recommendation

Content-based music recommendation algorithms vary in the features and the similarity function. Various criteria have been used to evaluate similarities, including metadata (title, artist, country), genre and expert tags, symbolic (melody, harmony, structure), perceptual (energy, texture, beat), and even cognitive (experience, reference) [59]. However, previous studies either overlook the hierarchical structure within the audio features, only consider one aspect of acoustic features.

1.2.3 Non-availability of Trust scores

The limitation of social recommendation with trust network is that they only exist in systems where users can explicitly give trust scores. In many real world recommender systems, explicit trust scores are unavailable. Instead, they allow users to follow others or add others into her social circle. Few studies have incorporated social networks into the recommendation algorithms [31] [63]. Besides social networks, user similarity network

can also be constructed to help improve recommendation performance.

1.2.4 Social influence in Recommender systems

A more critical problem with previous social recommendation algorithms is that models were predefined based on heuristics. For instance, in Ma et al.'s study [77] the heuristic was that the social network and rating matrix should share the same user latent factors. Based on this heuristic, user trust network was simultaneously decomposed with the rating matrix. In their later study [79], the heuristic was that a user's preference vector should be as close as possible to her friends' preference vectors. Based on this heuristic, social networks was added as a regularization term to the objective function of the matrix factorization model. Although these heuristics were reasonable and the predefined models outperformed the baselines, it was unclear why these models worked. We believe that only after understanding the contexts in which social networks work in recommender systems, we can develop more appropriate social recommendation algorithms. This requires the study of social influence in recommender systems.

Social influence denotes that "one person performing an action causes people connected to her to do the same [22]. It has been employed in various applications such as recommender systems [5][91][104][112][113], viral marketing [33] [63] [94], information diffusion [8] [26] [83] [94] [119] [122] [126], and etc.

1.2.4.1 Existence of social influence in Recommender systems

Previous social recommendation studies and most social influence studies assumed that social influence was the only reason for the information diffusion. However, according to Anagnostopoulos's study [4], social influence is just one source of social correlation. Before employing social influence into recommendation algorithm, we need to first

validate the existence of social influence.

1.2.4.2Impact of social influence in Recommender systems

Once social influence was validated as the source of information diffusion, it can be used in applications. Although social influence studies have been widely adopted in aforementioned areas, most of them focus on influence maximization, where the goal is to find a set of seed users to target so as to maximize the influence spread. To our knowledge, we have seen few studies [127] which apply social influence analysis to developing recommendation algorithms. In a recommender system, we say that a user is activated if he reads a book or listens to a music. Given a user and her social connections' actions on an item in the past, we can build a model of social influence, and apply the model to predicting whether a user will adopt the item in the future. If the prediction is that the user will adopt it, we can recommend the item to the user.

1.3 Thesis Goals

The ultimate goal of this thesis is to develop recommendation algorithms which can solve the item-side and user-side cold-start problems. We realize it is not feasible to solve all the issues related to the cold-start problem, so we aim at addressing the core problems listed in the last section.

1 First, we will develop a content-based movie recommendation algorithm which incorporates user, actor or director, and movie into the same model.

2 Second, we will develop a content-based music recommendation algorithm. The core of this algorithm is a music similarity measurement which consider the hierarchical structure of content features.

3 Third, we will look into a "Twitter-like" user-follower social network and see if it

can improve recommendation performance.

4 Finally, we will validate the existence of social influence in our dataset and build a social influence model. We will then use the learnt model to item recommendation.

1.4 Thesis Contributions

Based on the goals above, this thesis make the following contributions. We summarize them in Figure 1.2.

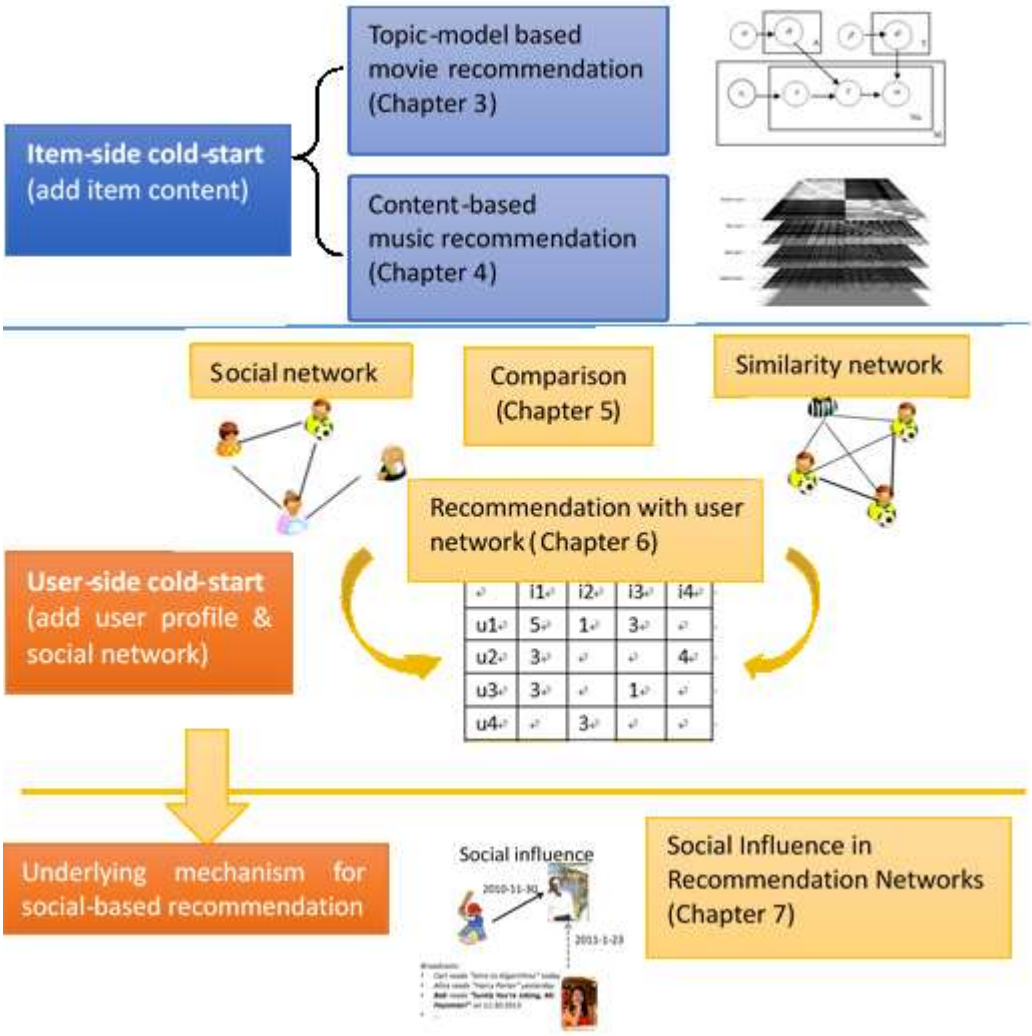


Figure 1.2 Summary of our contributions

1.4.1 Topic-model based movie recommendation

We try two approaches to extend the Author-Topic model [96] and apply them to movie recommendation. The first approach is to add user and item bias to the Author-Topic Model. In the second approach, we add another regularizer onto the matrix factorization model. We showed that topic models, in general, can be used in recommendation tasks. In implicit rating prediction task, they have comparable performances with the matrix factorization model. In explicit rating prediction task, they outperformed MF.

1.4.2 Hierarchical Music Recommendation

We build a hierarchical dynamic programming algorithm, Hierarchical Sequence Alignment (HSA) to measure music similarities. The resulting music similarity matrix is then employed to make recommendations. We focus on investigating how performances of music recommendation algorithms vary on songs with different popularities. We first introduced our dataset and how we preprocessed the data. Results show that CF does perform poorly on cold songs as we expected. Besides, single layer alignment algorithm BTMLYR has comparable performance on cold songs, indicating that this algorithm is able to give recommendations with both novelty and relevance. However, to the contrary of our expectation, Hierarchical sequence alignment algorithm HSA also performs between on popular songs than cold songs. Finally, single layer alignment algorithm BTMLYR outperforms HSA.

1.4.3 Non-availability of Trust scores

Unlike Epinions [2], most recommender systems do not provide trust scores. Accordingly, we crawl a dataset from Douban[3], one of the largest recommender systems in China. We apply existing social recommendation algorithm on Douban dataset and compare them with the state-of-the-art matrix factorization model. We also compare the result on Douban with previous findings from Epinions. We then substitute the social network with user similarity network to see which type of network can help improve recommendation performance better. When incorporating user similarity network, we found out that the Social-regularization Model performs best. In particular, it outperforms Social-fusion model. The reason may be that in Social-fusion model, the factorization of both the rating matrix and the social network are constrained by sharing the same user feature matrix. Such constraint makes it difficult to learning the parameters. Moreover, the Social-fusion model is much sensitive to parameter tunings than the Social-regularization model.

1.4.4 Social influence in Recommender systems

We follow Anagnostopoulos et al.'s [4] approach to identify social influence from two other sources, homophily and confounding. We analyze the information passing in an online recommendation system. Our dataset consists of a "read" network between users and books, and a user-follower network. We first investigate in general, if one's recommendations have impacts on her followers' decisions. We then analyze the correlation between one's influence and her network centrality. Finally, we investigate how

[2] Epinions. http://www.epinions.com/

[3] Douban. http://www.douban.com/

recommendation effectiveness changes with the number of recommendations. This investigation is taken from both senders' and receivers' perspectives. Results show that a user does have influence over her followers' decisions. Such influence is not correlated with her centrality. The more a book is recommended, the more likely that one will accept it. However, there is a saturate point beyond which more recommendations will have no more impact. On the other hand, the more recommendations one makes, the more likely that her recommendations will be accepted. This trend has no saturate point.

1.5 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we review related work in item recommendation. Chapter 3 and Chapter 4 both target the item-side cold start problem. Chapter 3 addresses the first goal, which is to develop a content-based movie recommendation algorithm by incorporating user, movie and casting information. In Chapter 4, we address the second goal, which is to develop a content-based music recommendation algorithm. This algorithm considers the hierarchical structure of acoustic features when calculating music similarities. Chapter 5 and Chapter 6 target the user-side cold start problem. In Chapter 5, we compare the properties of similarity network and familiarity network (social network). We then compare several social-based recommendation models on our dataset. In Chapter 7, we validated the existence of social influence and analyze the information passing patterns in our dataset. Finally, Chapter 8 concludes this thesis and present future directions.

Chapter 2. Background and related work

Recommendation is a broad topic covering studies from tag recommendation [75], image annotation [27], question and answering [46] and etc. In this thesis, we only focus on item recommendation. The core technique is to predict and present items (including movies, music or books) that users may like.

In general, recommendation algorithms can be divided into Collaborative Filtering (CF) and Content-based filtering [2]. CF assumes that users who have similar preferences in the past will share interests in the future. Content-based filtering assumes that users prefer items sharing similar content with those they liked before.

This chapter reviews the main approaches that have been applied to item recommendation. We will first introduce collaborative filtering. As mentioned in the Introduction, collaborative filtering suffers from both item-side and user-side cold-start problems. Hence, we then present content-based models which involve contextual information to solve problems. We first introduce approaches which employ item content to solve item-side problem in section 2.2 and 2.3. In section 2.2, we present various topic models on movie recommendation. In section 2.3, we present studies which use acoustic features on music recommendation. Then, in section 2.4, we show studies which incorporate user demographic information or social network to solve the user-side cold start problem. Finally, in section 2.5, social influence in recommendation network is summarized, which facilitates better design of social-based recommenders.

2.1 Collaborative filtering

Based on [17], Collaborative filtering is grouped into memory-based CF and model-based CF.

2.1.1 Memory-based CF

Memory based CF is one of the most widely applied recommendation approaches in the early age of the development of recommender systems [17] [50] [100] [74]. It can be further divided into user-based methods and item-based methods. User-based method recommends items preferred by the most similar users [17] [50]. Item-based method recommends items most similar to the active user's previous preferred items [100]. It has been successfully applied in Amazon's recommendation engine [74]. The core step of memory-based approach is to compute the similarities between users or items. PCC (Pearson Correlation Coefficient) [93] and VSS (Vector Space Similarity) [17] are the two major similarity measurements. For user-based CF, PCC computes two user's similarity based on items they both rated. The PCC similarities range from [-1, 1].

$$sim(a,b) = \frac{\sum_{i \in I(a) \cap I(b)} (r_{ai} - \bar{r}_a) \cdot (r_{bi} - \bar{r}_b)}{\sqrt{\sum_{i \in I(a) \cap I(b)} (r_{ai} - \bar{r}_a)^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(b)} (r_{bi} - \bar{r}_b)^2}} \quad (2.1)$$

VSS computes two users' similarity by measuring the cosine similarity of their vector, which are their ratings of items.

$$sim(a,b) = cos(\vec{a}, \vec{b}) \frac{\sum_{i \in I(a) \cap I(b)} r_{ai} \cdot r_{bi}}{\sqrt{\sum_{i \in I(a) \cap I(b)} r_{ai}^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(b)} r_{bi}^2}} \quad (2.2)$$

PCC and VSS become identical when both users' ratings have a mean of zero. After computing the similarities, the predicted rating for user u to item i will be an aggregation of these similar users' ratings on$i$. The aggregations can be done in different ways [2]. (2.3) gives one of the ways.

$$r_{ui} = \bar{r}_u + \frac{1}{\sum_{u' \in N_u} |Sim(u,u')|} \sum_{u' \in N_u} Sim(u,u') \cdot (r_{u'i} - \bar{r}_{u'}) \quad (2.3)$$

Albeit the fact that these two similarity measures are simple to implement, they have their limitations. For Pearson correlation, first, it does not consider the number of items

that two users both rated. Two users who rate thousands of different items but only share ten ratings, could have an inaccurate similarity of one. Second, if two users overlap on only one item, no correlation can be computed. Finally, the correlation is also undefined if either series of preference values are all identical. For VSS, the similarity of two users who rate only one item in common will be one.

Item-based CF works similarly as user-based CF. It computes item similarities instead of user similarities. The predicted rating for user u to item i will be an aggregation of u's ratings on those items similar to i.

2.1.2 Model-based CF

Memory-based CF predicts ratings by processing the user-item matrix directly. It has no training phase and thus easy to implement. However, memory-based approach has problems. First, due to data sparseness and the aforementioned limitations in similarity measures, it does not have good accuracy. Second, it does not scale well since all pair-wise similarities need to be computed. Finally, because it does not have a learning stage, it only tailors a specific system and no new knowledge is gained.

Model-based CF is proposed to solve these problems. Instead of giving recommendations based on the matrix directly, model-based CF first trains a model from the rating matrix, then given predictions based on the trained model [40] [80] [121]. Thus, model-based CF can be applied to different datasets. Popular models include matrix factorization models [11] [40] [92] [113], probabilistic models [28] [52] [61] [81] [106] and hybrid models [65]. These models share the assumption that the ratings can be explained by some latent variables.

The MF models share the same idea with Singular Value Decomposition (SVD) [34].

The basic assumption of latent factor models is that users and items are associated with a factor vector. Users' interests in items (e.g. shown in their ratings) depend solely on the interactions between users and items. Basically, the user-item matrix R, is decomposed into the product of a user feature matrix U, and a item feature matrix V, $R = UV^T$. The meaning is that each user's ratings on an item, can be seen as a linear combination of his preference on each latent features. However, SVD is prone to overfit. Regularized MF is proposed to reduce overfitting, in which regularizations on the norms of U and V are added onto the MF model [92].

Take regularized MF for example. The objective function is to minimize the sum-squared error between the predicted ratings and the original ratings, plus the regularization terms (2.4). I_{ij} is the indicator function, which equals to one when user i rated item j. U_i and V_j are low dimensional vectors which represent preferences of users and items.

$$\min_{U,V} L(R,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 \quad (2.4)$$

ALS (Alternative Least Square) or Stochastic Gradient descent can be used to learn the user and item feature matrices [67]. It is suggested to use SGD over ALS when the data is sparse. The reason is that stochastic gradient descent loops through each rating case, while ALS alternatively fixes all user feature vectors to learn each item vector, and vice versa.

Though matrix factorization methods are the state-of-art, they suffer from the problem that the latent features are not interpretable, since the common factors are used to interpret both users and items. Probabilistic approaches can ease this problem by introducing more hidden variables in priori [52] [54] [55] [56]. The two-sided clustering model assumes that each user should belong to exactly one group and so does each item [52] [71] [72] [73].

The aspect model introduced one set of latent variables$Z$, which implicitly cluster both items and users. Experiments showed that the performance of aspect model is substantially better than the two-sided clustering model [52]. The FMM model [106] is motivated by the two-sided clustering model and the aspect model. It introduces two groups of latent variables to cluster items and users separately. Besides, it introduced an additional hidden variable "preference", to account for the variance in the rating patterns among different users with similar interests. Other hybrid approaches include merging the MF model with the memory-based (neighborhood) model, which performed well on the Netflix data [65].

Later, MF and probabilistic models are combined in a systematic way [98][99]. Salakhutdinov et al. proposed a Probabilistic Matrix Factorization (PMF) model in which the distribution of each rating r_{ui} is Gaussian [98]. The mean of each rating r_{ui} is the inner product of user u's vector and item i's vector. All the Gaussian distributions share the same variance. Besides, each user and item feature vector has a prior of zero-mean spherical Gaussian distribution. All user vectors share the same variance, and all item vectors share the same variance. PMF is mathematically equivalent to the regularized MF model. However, PMF has problem for users with few ratings. For these users, their feature vectors will be close to the prior mean, thus the predicted ratings for them will be close to the average rating of all users. In the same paper, they also presented a Constrained Probabilistic Matrix Factorization model. The rationale behind constrained PMF is that "users who have seen the same movies will have the same prior distributions for their feature vectors" [98]. In [99], the Gaussian priors were replaced by the Gaussian-Wishart priors.

Based on the pLSA model [51], Hoffmann proposed the Gaussian-pLSA model for

the recommendation task [56]. It differs from PMF in that, the distribution of each rating given a (community, item) pair is assumed to be Gaussian. Moreover, instead of sharing the same variance among all ratings, each rating given a (community, item) pair also has a unique variance. Intuitively, the assumptions in Gaussian-pLSA model are more reasonable. The rationale is that users in the same community share common interests, so it is more likely that they have a similar preference over the same item.

2.2 Cold-start movie recommendation

There already exist some collaborative filtering models applied to solving the item-side cold start problem. In [88], a boosting model was proposed, which injected several filter bots into the collaborative filtering model. Filter bots included critic ratings and feature bots, where features included genre and casting information. Results showed that filter bots can help improve recommendation in cold-start settings.

However, a much larger number of content-based models have been developed to solve the item-side cold-start problem. Among them, [90] was one of the earliest work. In this study, a three-way aspect model was proposed, which combined collaborative and content-based recommendations. It extends Hofmann's two-way aspect model [56] by adding a third dimension, item content. The model assumed that each user has his or her interested topics. These topics "generate" both items themselves and their content. Results showed that content information can help improve performance of simple collaborative approaches such as k-nearest neighbors. However, this model is applied to paper recommendation instead of movie recommendation.

In their later study [102], Schein et al. applied the two-way aspect model to movie recommendation by incorporating director and actor information. Based on the assumption

that “casts of actors can act as surrogates for movies”, the original user-movie matrix is transformed into a user-actor matrix. Each observation (u, m) is used to generate a set of observations (u, a_i) where a_i is an actor of m. Given a user, a movie is recommended if it is similar to those movies he has already rated. The model performance was evaluated on two tasks. In the Implicit Rating Prediction task, the model only predicted whether a user will rate a movie or not. In the rating imputation task, given that a user watched movie m, the model predicted the probability that he would rate it higher than four. Results showed that the aspect model outperformed naïve Bayes in implicit rating task, but performed worse than the user average rating in the rating imputation task.

A hybrid model was proposed in [101]. It first performed user-based collaborative filtering. Then, content-based filtering was performed based on the results. It added the predicted rating from user-based CF to the scores of each content feature of the movie. For instance, if the predicted rating was five, then the scores of each actor, director and the genre of the movie were added by five. Finally, the average score of each content element was calculated. Results showed that, in the scenario of general recommendations, the system did not have significant improvement over baselines. However, in recommending new movies, the system outperformed pure content-based method.

Besides probabilistic models, other models have also been used to incorporate actor information. A recommendation algorithm named CONTEXTWALK was proposed which was based on Markov random network [14]. It included tags, movie genres and actors. It was flexible and could include other contextual information. However, the authors did not empirically test their model on any dataset.

Other models include regression models [89] and Boltzmann machines [45]. In [89],

each user was represented by a vector of dimension C. Each item was represented by a vector of dimension D. The predicted rating was the weighted outer product of the active user vector and item vector. For the loss function, this paper used pairwise loss instead of least-square loss. This was based on the assumption that, if a user rated two items similarly, then the similarities between these two items with the user should also be close. Results showed that the proposed pairwise model outperformed other models in all three cold-start settings. In [45], the Tied Boltzmann machine was proposed. In this model, only actors were included as item features. Results showed that tied Boltzmann model outperformed the user-actor aspect model in both runtime and accuracy. Besides, the tied Boltzmann machine does not have overfitting problem whereas the user-actor aspect model is easy to overfit.

2.3 Hierarchical Music Recommendation

Music recommendation algorithms are classified into demographic filtering, collaborative filtering, content-based filtering and hybrid models [24]. Demographic filtering assumes that the type of user and her music preference are related, and tries to identify such relationship. It clusters users into groups based on users' profiles including personal information (age, marital status, gender, etc.), geographic information and psychographic information (interests, lifestyle, etc.) [24]. However, due to privacy issues, user profiles are usually not publicly available. Collaborative filtering is of one the most adopted recommendation approaches. It have been empirically tested that CF outperforms content-based approach in music recommendation [109]. However, CF suffers from the item-side cold start problem. New users with no ratings will not get recommendations, and new items with no ratings will not be recommended [87]. The second drawback of CF is

the gray sheep problem where it is difficult for CF to recommend music to "niche listeners" who have special tastes [29]. The reason is that the music they have listened to have little usage data. This problem is most prominent in music recommendation than in other domains. The last problem, denoted as "popularity bias" problem in [24], means that CF assigns large similarities between popular music with other music pieces. This leads to the situation that popular music is more probable to get recommended.

The content analysis of multimedia data including text, video and music has gained a lot of progress recently [84]. Content-based recommendations could potentially overcome both problems by calculating the similarities between new item or niche item with existing items in the system. Content-based recommenders can be divided into two groups, those based on metadata and those based on audio signals [108]. However, it was pointed out that metadata-based approaches cannot recommend niche items [24].

The core of content-based music recommendation is how we measure music similarity. Measuring similarities between two music pieces in audio format is a challenging task. The first challenge is that music features are neither orthogonal nor equally important. Hence, although the simplest approach to measuring music similarity is to first represent each song into a feature vector, then calculate Euclidean distance between vectors, such an approach is based on unrealistic assumptions of the orthogonality and equal importance of music features [108]. In [108], assuming that a good similarity metric should put songs with similar metadata together, the authors transformed music similarity measurement into a classification task. Given audio features and class labels of album, artist or blog, their algorithm learned a metric space which put similar songs close to each other under the K Nearest Neighbor (KNN) classifier. Experiments were done based on a dataset of 18

features per song, and class labels of 74 albums, 164 artists and 319 blogs. Results showed that Neighborhood component analysis (NCA) performed best on all three classification tasks. When noisy features were added, NCA and Relevant component analysis (RCA) performed best.

However, existing content-based approaches have limitations. The authors in [87] pointed out that metadata based approaches often lead to uninteresting recommendations since they recommended music of similar artists or genres with users' listening history. Audio based approaches need to define an ad-hoc similarity metric. Such heuristic similarity metric may not be able to bridge the "semantic gap" between song characteristics related to user preference and corresponding audio signal, and hence makes audio-based approaches unable to generate accurate recommendations. This is also the reason why content-based recommenders could not beat CF. CF captures high level semantic similarities between music from the collective behaviors, while acoustic based approaches are not able to extract such similarities. A. Van den Oord et al. [87] believed that latent factors were able to bridge this gap. However, traditional latent factor models perform poorly when usage data is scarce. To solve this problem, this paper first applied latent factors from Weighted Matrix Factorization model (WMF) [57] to the available usage data. These latent factors were then used as ground truth to train a regression model to predict new item latent factors. Results showed that, the neural network model outperformed models based on bag-of-words representation. However, there was still a large gap between the performance of the proposed model and that of the latent factor model. They concluded that it was because many aspects that affected user preference cannot be obtained from audio signals only.

The second challenge in measuring music similarity lies in that various criteria can be used to evaluate similarities, including metadata (title, artist, country), genre and expert tags, symbolic aspects(tempo, melody, rhythm, timbre, harmony), perceptual aspects (energy, texture, beat), and even cognitive aspects (experience, reference) [59]. Each user may have her unique criteria for similarity. All the aforementioned content-based and hybrid music recommendation models only focus on one or few criteria [82] [35]. Moreover, they overlook the fact that music similarities can be measured on different scales/levels, and only look at one level [7]. To overcome these limitations, Tristan proposes a recursive method to compute the acoustic similarities from multiple aspects [59]. Similarities on each aspect is computed recursively through dynamic programming. Our content-based music recommendation algorithm is mostly inspired by this study.

2.4 Recommendation with social network

To find out if social network can help improve recommendation, we need to first analyze the relationship between one's social network and his interest-sharer network. The former is a network consisting of people the user knows. The latter is a network consisting of people who share similar interests, but are not necessarily known by the user. Our assumption here is that if these two networks are similar, then social networks may provide little extra information, thus has little use in the recommendation. If they are different, it can be useful for recommendation. Studies that analyzed the relationship between the user-item matrix and the trust (social) network among users" include [1] [58]. Adamic et al. studied the friendship network in an online community of Stanford University [1]. Results showed that users who share interests are more likely to be friends. They also found that a user's number of friends has a small but significant correlation with his number of interests.

Besides, weak ties often link two dissimilar users. In [58], a similarity network and a social network in a Chinese recommender system were analyzed and compared. The similarity network was constructed from the co-reading relations between two users. The social network was constructed by the follower/followee relationships. Results showed that both networks have the small world property. Besides, top users under different centrality measures are consistent for the co-reading networks, but not for the friendship network. Finally, the cosine similarities between the same user's vectors in the two types of networks were very small, indicating that there is little correlation between familiarity and similarity-based networks.

More recent studies differentiate friendships (familiarity) network with similarity network, and compared recommendations based on the two networks [69] [15] [107]. Groh et al. proposed a method to predict users' ratings to 82 clubs. They tested if users' rating behaviors are correlated with their friendships. Experimental results show that friends have more similar ratings than strangers. Besides, familiarity-based recommendation performs better than similarity-based recommendation, especially when the data is very sparse [44]. In [69], two recommendation algorithms based on the two relations were also compared. In [48], the field study showed that familiarity network was more effective to help recommendation than similarity network.

Different from previous studies, Ma et al. combined users' ratings with social networks to do recommendation [77]. They analyzed Epinion[4], a system for users to share knowledge and provide reviews. Each user has a trust list and a distrust list, which constitutes the trust social network. It assumed that a common user feature matrix existed

[4] Epinions. http://www.epinions.com/

between the trust network and the user-item matrix. Results showed that it has a high prediction quality and could scale well to large datasets. However, since in this system, users explicitly express their trust and distrust relations, their social network may affect their interests. Thus, their algorithm may not perform well in other systems without this feature. In their follow-up study [79], they incorporated social network information as additional regularization terms. Their first regularization model assumes that a user's taste should be close to the average of her friends' tastes. Their second regularization model assumes that the tastes of friends who are more similar to the target user should contribute more to his tastes. Analysis on Douban [5] showed that social network can help recommendation [79].

2.5Social Influence in Recommendation Networks

There are two lines of research in the existing social influence studies. One line of research focus on validating the existence of social influence. The other assumes social influence already exists and applies it to different applications.

2.5.1 Existence of social influence

Anagnostopoulos et al. believed that social influence is just one source of social correlation [4]. They proposed two tests to identify social influence from two other sources, homophily and confounding. Both tests were performed based on the linear threshold model. In the "shuffle test", they created a new graph by shuffling all the timestamps. In the "edge-reversal test", they created a new graph by reversing the directions of all the edges. In both tests, they ran logistic regression on the original graph and the new graph.

[5] Douban. http://www.douban.com/

If the values of the same model parameter were close in two graphs, the model exhibited no social influence.

2.5.2 Influence Maximization

In the data mining field, the influence in social networks was first studied as a viral marketing problem [33] [94]. Kempe et al. [63] later formulated this problem as an optimization problem named "Influence maximization", and focused their study on two models, the Linear Threshold model (LT) and Independent Cascade model. Linear Threshold model originated from Granovetter's study [43]. In this model, each user has a threshold and is influenced by her connectors with a weight. Given an inactive user at timestamp$t$, only when the total weights from her connectors exceeds her threshold, would she be activated in $t+1$. In Independent Cascade model, at each timestamp$t$, each user has only one opportunity to active her inactive neighbors. Under both models, the goal was to maximize the expected number of activated nodes in the end.

2.5.3 Impact of social influence in Recommender systems

One of the earliest studies on the social influence in recommendation network analyzed a website which gave discounts to customers who recommended products to others [70]. It studied the impact factors on recommendation effectiveness, the latency between recommendation and users' decisions, and how recommendation effectiveness varied with product categories. They showed several interesting results which are contradictory to the assumptions of popular diffusion models. First, recommendation effectiveness did not increase continuously, but increased and then saturated as more and more recommendations coming from friends. Second, users with large degree centralities had smaller recommendation effectiveness than users with small degree centralities. As to

product category, recommendations were more successful on technical or religious books. They concluded that viral marketing in online recommendation systems was not as epidemic as one might have expected.

In a more recent study [47], they analyzed the information passing process in online shopping based on the transactions and messages exchanged between a sample of Taobao users. Information pass rate was defined as the probability that a user's friend would buy a product given that this user bought it and told this friend. Results showed that the information pass rate in the social network is significantly larger than that in a rewired random network. Although they analyzed the relationship between the information pass rate and various factors such as message strength, time in a day and product price, they did not explicitly model the information pass rate in a thorough way.

In [42], the authors built a network cascade transmission model. In this model, the transmission probability depended only on the difference between times when two nodes became affected. In [127], the authors developed a probabilistic generative model, the social influenced selection (SIS) model, which integrated social influence, user behavior and item content. They applied this model to item recommendation and group recommendation. Result showed that the SIS model was effective for both recommendation tasks. They also showed that users rarely followed their friends' uncommon preferences. Moreover, the strength of influence between a user and one friend was not correlated with their similarity.

Chapter 3. Topic-Model based recommendation

Collaborative filtering has become the mainstream approach to item recommendation. However, it cannot recommend new items to users since they have no rating information. In this chapter, we propose a novel recommendation algorithm - Actor Topic Model. This model is originally designed for movie recommendation, but can also be applied to many other problems.

In the Actor Topic Model, each user is modeled as a document. All the movies she has watched are modeled as the "words" for her. Each actor of the movie is modeled as an author. We model the generative process for each movie as below. First, one actor is selected from the user's actor set. Then, a topic is sampled from this actor's topic distribution. A movie is then sampled from topic's movie distribution. After learning, we obtain the topic distribution of each actor and each movie. We then compute each user's topic distribution by averaging the topic distributions of his "watched" actors. The rating is predicted as the similarity between the topic distributions of the active user and movie.

Then, we extend the Matrix Factorization (MF) model by adding topic-related regularization terms, which are learnt from the Actor-Topic models. The assumption here is that a user (item)'s latent factor should be close to her (its) topic vector. We show that in implicit rating prediction task, topic models have comparable performances with MF, while in explicit rating prediction task, they outperform MF. Thus, topic models, in general, can be applied to recommendation tasks.

3.1 Introduction

Collaborative filtering has become the mainstream approach to item recommendation, including movie recommendation. It gains popularity since it can perform really well by

just utilizing users' ratings on movies. Thus, it is easy to implement and is domain independent. However, collaborative filtering requires the "rating profiles" of users and items to compute their similarities. In real-world systems, rating information on new items are not always available. Such problem, which is called "cold start" problem, has undermined the capability of traditional collaborative filtering methods.

Content-based approach can solve this item-side "cold-start" problem by utilizing new items' content information. Take movies for example. Content can include movies' metadata, description and even reviews. Even though descriptions and reviews might not be available and need to be preprocessed, movie metadata could always be easily accessible. Among metadata, actor and director information are important factors in helping people decide which movie to watch, and how they rate them.

The two-way aspect model and the two-sided clustering models are among the earliest graphical models applied to recommendation tasks [52]. The two-way aspect model extended the aspect model [53] to capture users' preferences. Similar with the aspect model, it is also a soft clustering model which models each user's preference as a combination of preference vectors over latent factors. The two-sided clustering model, on the other hand, is a hard-clustering model which partitioned each user and item into exactly one cluster. Results showed that the two-way aspect model significantly outperformed the two-sided clustering model in movie recommendation. Later, the three-way aspect model was proposed [90]. It extended the two-way aspect model [53] by adding a third dimension, item content. The model assumed that each user has her interested topics. These topics "generate" both items themselves and their content. However, this model is applied to paper recommendation instead of movie recommendation. In [103] and [102] two-way

aspect models and three-way aspect models were applied to movie recommendation using MovieLens dataset [102]. In [103], only actor information was incorporated into the two-way aspect model. Based on the assumption that "casts of actors can act as surrogates for movies", the original user-movie matrix is transformed into a user-actor matrix. Given a user, a movie is recommended if it is similar to those movies he has already rated. The model performance was evaluated on two tasks. In the Implicit Rating Prediction task, the model only predicted whether a user will rate a movie or not. In the rating imputation task, given that a user watched movie m, the model predicted the probability that he would rate it higher than four. Results showed that the aspect model outperformed naïve Bayes in implicit rating task, but performed worse than the user average rating in the rating imputation task. In [102], several additional two-way aspect model were tested. Three-aspect models were also built by including director information. Results showed that two-way models performed better than three models. They concluded that actors can provide enough information to recommend movies that no user has seen, while directors do not help. However, in the three-way aspect model with the best recommendation performance, users, actors and movies are assumed to be independent with each other given topics. This is not true in reality.

Topic models, such as Latent Dirichlet Allocation (LDA) [13] and pLSA [51], assume the existence of latent topics. They have achieved their success in modeling text and have been applied to paper recommendation [119]. Author-Topic Model extended LDA [13] by including authorship information. Its generative process is as follows: (1) Build a topic distribution for each author of a document. (2) Choose a topic from the author-topic distribution. (3)Choose a word from the topic-word distribution. Compare to LDA [13],

the Author-Topic model is particularly useful when little content information is available for new documents. Besides, it can learn author topics and make author-specific predictions.

Although topic models have been widely adopted in paper recommendation, few researchers have applied them to movie recommendation due to the lacking of textual information. As a movie's actor information is often available, we incorporate it into movie recommendation. Let us first recall a user's movie rating process. When a new movie is released, a user might care most about who play in this movie, and then decide whether to watch it. Assume that actors keep their performance level consistent, we can argue that actors would affect users' ratings on the movie. Since the relationship between actors and movies resembles the one between authors and documents, the Author-Topic Model inspires us to propose an Actor-Topic Model for movie recommendation.

This model is based on two assumptions. First, we assume that a user's rating (either implicit or explicit) on a movie depends on the interaction between the user and the movie's topic distributions. Second, we assume that each movie is made of many "plots", and each "plot" is generated by an actor. This actor's preferences determine this plot's topic. All actors of a movie determine its topic distribution. We later propose the ATM+MF model, which combines the actor-topic model with the matrix factorization model.

The closest studies to our ATM+MF model are [12] [119] [3], in which they combined topic models with latent factor models. When combining topic model and MF, people tend to replace the item factors in MF with topic distributions learned from topic model when rating data is not available. In both [3] and [119], an LDA prior was added onto the item factors of a matrix factorization model. The difference is that in [119], user factors depended only on the ratings. In [3], user factors were learnt from a regression on user

features. Thus, user profiles were needed as input. The model in [119] was applied to paper recommendation. The model fLDA in [3] was applied to rating prediction on Movielens, Yahoo! Buzz[6] dataset and BookCrossing dataset[7]. Results showed that fLDA outperformed Collaborative filtering for cold start items.

The remainder of this chapter is organized as follows. Section 3.2 presents the Actor-topic model, and how it is applied to the movie recommendation problem. Section 3.3 introduces dataset and evaluation metrics. Section 3.4 reports experimental results. We conclude this chapter and propose future work in section 3.5. This chapter is based on our paper submitted to WI 2013.

3.2 Models

We introduce our models for movie recommendation below. In the first two sections, two variations of the LDA model are presented as baselines. In the third section, the actor-topic model is proposed. In the last section, we propose a hybrid model, which combines the actor-topic model with the matrix factorization model.

3.2.1Two LDA Models

Latent Dirichlet Allocation (LDA) is a generative model in which each document is modeled as a mixture of topic probabilities. LDA has been applied to various problems in text mining [13], tag recommendation [68] and paper recommendation [119]. However, LDA has not been used in recommendation problems with little text, such as movie recommendation. Thus, before introducing the Actor-Topic model, we first apply the LDA

[6] As of the time this thesis was written, the website does not exist.
[7] The BookCrossing dataset can be downloaded from http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

model to movie recommendation.

3.2.1.1 Model LDA-I

In LDA-I, users are modeled as documents. The movies each user has watched are modeled as her "bag of words (movies)". We assume that each movie is generated from the process shown in Figure 3.1. The outer plate represents users, and the inner plate represents movies of each user. Table 3.1 summarizes this notation.

(1) Choose each topic $\varphi_k \sim Dir(\beta)$, for $k \epsilon \{1, \dots, K\}$

(2) For each user u:

(1) Choose topic proportions $\theta_u \sim Dir(\alpha)$

(2) For each movie at position j in user u's collection, for $j \epsilon \{1, \dots, N_u\}$

(1) Choose a topic $z_{u,j} \sim Multinomial(\theta_u)$

(2) Choose a movie $m_{u,j} \sim Multinomial\left(\varphi_{z_{u,j}}\right)$

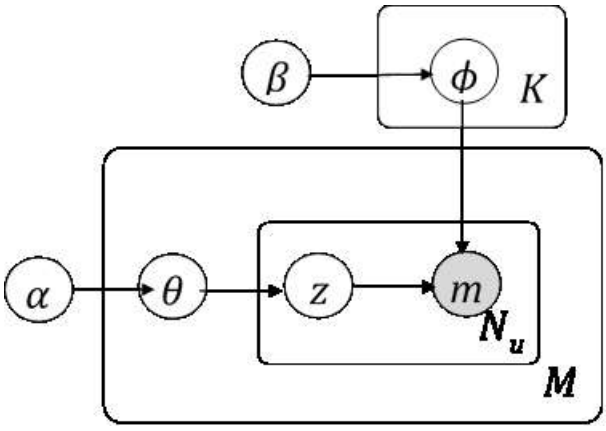


Figure 3.1 Plate notation for LDA-I [13]

Given one user's watching history, we can infer this user's topic distribution, and each

topic's movie distribution. Then, we get each movie's topic distribution by normalizing each topic's movie distribution. Finally, prediction is made by taking the inner product of the active user and movie's topic distributions. Note that this version of LDA model has been proposed in the "7.3 Collaborative Filtering" section of the original LDA paper [13].

Table 3.1Symbols associated with LDA-I

K	Number of topics	Scalar
M	Number of users	Scalar
N	Number of movies	Scalar
N_u	Number of movies u has watched	Scalar
$\boldsymbol{m}$	Movies in the corpus	N-dimensional vector
α	Dirichlet prior	Scalar
β	Dirichlet prior	Scalar
$\boldsymbol{z}$	Topic assignments	N-dimensional vector
θ	Probabilities of topics given users	$M \times K$ matrix
φ	Probabilities of movies given topics	$K \times N$ matrix

3.2.1.2 Model LDA-II

In LDA-II, movies are modeled as documents. Users who have watched each movie are modeled as the movie's "bag of words (users)". We assume that each movie is generated from the process shown in Figure 3.2. The outer plate represents movies, and the inner plate represents users who have watched this movie. Table 3.2 summarizes this notation.

(1) Choose each topic $\varphi_k \sim Dir(\beta)$, for $k \epsilon \{1, \dots, K\}$

(2) For each movie i:

(1) Choose topic proportions $\theta_i \sim Dir(\alpha)$

(2) For each user at position u in movie i's audiences, $u \epsilon \{1, \dots, M_i\}$

(1) Choose a topic $z_{i,u} \sim Multinomial(\theta_i)$

(2) Choose a movie $m_{i,u} \sim Multinomial(\varphi_{z_{i,u}})$

Given users who has watched each movie, we can infer each movie's topic distribution, and each topic's user distribution. Then, we get each user's topic distribution by normalizing each topic's user distribution. Finally, prediction is made by taking the inner product of the active user and movie's topic distributions.

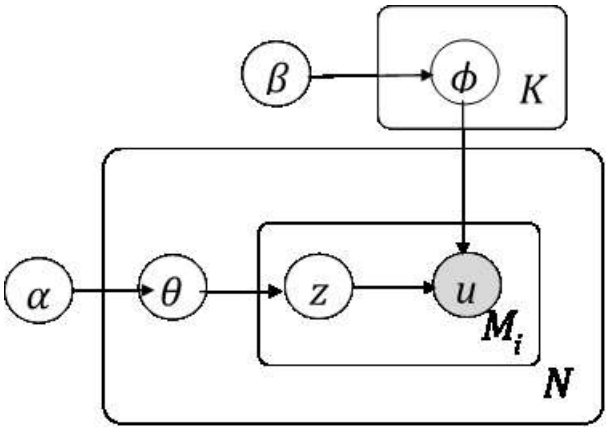


Figure 3.2 Plate notation for LDA-II

Table 3.2 Symbols associated with LDA-II

K	Number of topics	Scalar
M	Number of users	Scalar
N	Number of movies	Scalar
M_i	Number of users who have watched movie i	Scalar
$\boldsymbol{u}$	Users in the corpus	M-dimensional vector
α	Dirichlet prior	Scalar
β	Dirichlet prior	Scalar
$\boldsymbol{z}$	Topic assignments	M-dimensional vector
θ	Probabilities of topics given movies	$N \times K$ matrix
φ	Probabilities of users given topics	$K \times M$ matrix

3.2.2 Two Actor-Topic Models

Similar to LDA model, we also apply two variants of Author-Topic model (which we call Actor-Topic model) to movie recommendation. In the Actor-Topic models, besides implicit ratings, we also have movies' casting information as additional input.

3.2.2.1 ATM-I

In the first model (ATM-I), users are viewed as documents. All the movies watched by each user constitute "bag of movies" of this user. As a preliminary step, we chose top n_1 actors for each movie, and chose top n_2 actors in each user's watched movies as her "actor set". We assume that each movie is generated from the process shown in Figure 3.3.

(1) For each actor $a = 1, \dots, A$ choose $\theta_a \sim Dirichlet(\alpha)$

For each topic $t = 1, \dots, T$ choose $\varphi_t \sim Dirichlet(\beta)$

(2) For each user $u = 1, \dots, M$

Given the vector of actors a_u

For each movie $i = 1, \dots, N_u$

Conditioned on a_u choose an actor $x_{ui} \sim Uniform(a_u)$

Conditioned on x_{ui} choose a topic $z_{ui} \sim Discrete(\theta_{x_{ui}})$

Conditioned on z_{ui} choose a movie $m_{ui} \sim Discrete(\varphi_{z_{ui}})$

First, one actor x is selected from the user's actor seta_u. Then, a topic z is sampled from this actor's topic distributionθ. A movie m is then sampled from topic z's movie distribution φ. After learning, we obtain the topic distribution of each actor and each movie. We then compute each user's topic distribution by averaging the topic distributions of his "watched" actors. In the prediction stage, the rating is predicted as the similarity

between the topic distributions of the active user and the active movie. Table 3.3 summarizes this notation.

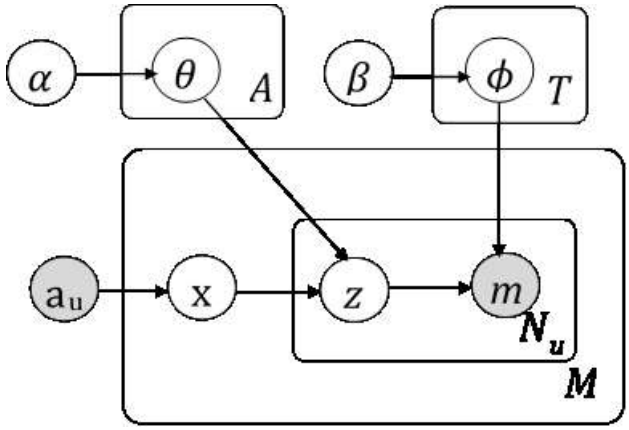


Figure 3.3 Plate notation for ATM-I

Table 3.3 Symbols associated with ATM-I

a_u	Actors in user u's watched movies	N_u-dimensional vector
A	Number of actors	Scalar
T	Number of topics	Scalar
M	Number of users	Scalar
N	Number of movies	Scalar
N_u	Number of movies user u has watched	Scalar
$\boldsymbol{m}$	Movies in the corpus	N-dimensional vector
α	Dirichlet prior	Scalar
β	Dirichlet prior	Scalar
$\boldsymbol{z}$	Topic assignments	N-dimensional vector
θ	Probabilities of topics given actors	$A \times T$ matrix
θ_a	Probabilities of topics given actor a	T-dimensional vector
φ	Probabilities of movies given topics	$T \times N$ matrix
φ_t	Probabilities of movies given topic t	N-dimensional vector

3.2.2.2ATM-II

In the second model (ATM-II), movies are viewed as documents. All the users who have watched this movie constitute the "bag of users". Each movie's actor set is modeled as its author set. We assume that each user is generated from the following process shown in Figure 3.4.

(1) For each actor $a = 1, \dots, A$ choose $\theta_a \sim Dirichlet(\alpha)$

For each topic $t = 1, \dots, T$ choose $\varphi_t \sim Dirichlet(\beta)$

(2) For each movie $i = 1, \dots, N$

Given the vector of actors a_i

For each user $u = 1, \dots, N_i$

Conditioned on a_i choose an actor $x_{iu} \sim Uniform(a_u)$

Conditioned on x_{iu} choose a topic $z_{iu} \sim Discrete(\theta_{x_{iu}})$

Conditioned on z_{iu} choose a movie $m_{iu} \sim Discrete(\varphi_{z_{iu}})$

First, one actor x is chosen from the movie's actor seta_i. Then, a topic z is sampled from this actor's topic distribution θ. A user is then sampled from topic z's user distributionφ. After learning, we obtain the topic distribution of each actor and each user. We then compute each movie's topic distribution by averaging the topic distributions of its actors. In the prediction stage, the rating is predicted as the similarity between the topic distributions of the active user and the active movie, the same as in ATM-I. Table 3.4 summarizes this notation.

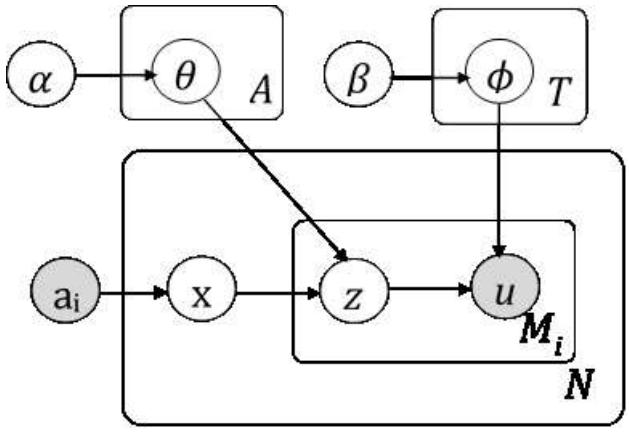


Figure 3.4 Plate notation for ATM-II

Table 3.4 Symbols associated with ATM-II

a_i	Actors of movie i	N_i-dimensional vector
A	Number of actors	Scalar
T	Number of topics	Scalar
M	Number of users	Scalar
N	Number of movies	Scalar
M_i	Number of users who have watched movie i	Scalar
$\boldsymbol{u}$	Users in the corpus	M-dimensional vector
α	Dirichlet prior	Scalar
β	Dirichlet prior	Scalar
$\boldsymbol{z}$	Topic assignments	M-dimensional vector
θ	Probabilities of topics given actors	$A \times T$ matrix
θ_a	Probabilities of topics given actor a	T-dimensional vector
φ	Probabilities of users given topics	$T \times M$ matrix
φ_t	Probabilities of users given topic t	M-dimensional vector

3.2.3 MF with Actor-Topic Regularization

Recall that the original objection function of Koren's Matrix Factorization (MF) model [65] is:

$$L = \min_{p_u, q_{i}, b_u, b_i} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda_1(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2) \tag{2.5}$$

In this study, we combined this MF model with the Author-topic model. Assuming that each user's latent factor should be close to her topic distribution. Similarly, each item's latent factor should also be close to its topic distribution, we present our objective function in equation(2.6). $L = \min_{p_u, q_{i}, b_u, b_i} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda_1(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2) + \lambda_2(\|p_u - t_u\|^2 + \|q_i - t_i\|^2)$

(2.6)

The regularization term

$$\lambda_2(\|p_u - t_u\|^2 + \|q_i - t_i\|^2) \tag{2.7}$$

is added to the matrix factorization model to represent our assumption that each user (item)'s latent factor should be close to the corresponding topic distribution.

In 3.1, p_u and q_i represent user factor and item factor learned from the MF model. t_u and t_i represent user topic vector and item topic vector learned from the Author-topic model.

Note that regularizer 3.2 inherits two additional assumptions. First, a user (item)'s latent factor has the same length with her (its) topic vector. Second, each element in a user (item)'s latent factor, should represent the same "topic" as the element in the topic vector on the corresponding position. For example, if $t_u = (0.1, 0.2, 0.7)$, with t_{u1} represents "science", t_{u2} represents "comedy", and t_{u3} represents "family", then p_u should also be a vector of length three, with p_{u1}represents "science", p_{u2} represents "comedy", and p_{u3} represents "family".

This algorithm goes through two steps. In the first step, we learnt t_u and t_i from the Actor-Topic model. In the second step, we plugged t_u and t_i into 3.1, and learnt model parameters p_u, q_i, $b_u$and $b_i$through stochastic gradient descent. The gradients of all the parameters and their learning functions are shown in equations (2.8) through(2.15).

$$\frac{\partial L}{\partial p_u} = 2e_{ui}(-q_i) + 2\lambda_1 p_u + 2\lambda_2(p_u - t_u) \tag{2.8}$$

$$\frac{\partial L}{\partial q_i} = 2e_{ui}(-p_u) + 2\lambda_1 q_i + 2\lambda_2(q_i - t_i) \tag{2.9}$$

$$\frac{\partial L}{\partial b_u} = 2e_{ui}(-1) + 2\lambda_1 b_u \tag{2.10}$$

$$\frac{\partial L}{\partial b_i} = 2e_{ui}(-1) + 2\lambda_1 b_i \tag{2.11}$$

$$p_u = p_u - \gamma\big((\lambda_1 + \lambda_2)p_u - e_{ui}q_i - \lambda_2 t_u\big) \tag{2.12}$$

$$q_i = q_i - \gamma\big((\lambda_1 + \lambda_2)q_i - e_{ui}p_u - \lambda_2 t_i\big) \tag{2.13}$$

$$b_u = b_u - \gamma(-e_{ui} + \lambda_1 b_u) \tag{2.14}$$

$$b_i = b_i - \gamma(-e_{ui} + \lambda_1 b_i) \tag{2.15}$$

Here,$e_{ui} = r_{ui} - \mu - b_u - b_i - p_u^T q_i$. We will call this algorithm ATM + MF in the remaining sections.

3.3 Experiments

We compare the performances of all four models on movie recommendation on the hetrec2011-Movielens dataset. In this section, we will introduce our dataset, baseline algorithms, evaluation tasks and metrics, and how we tune the parameters.

3.3.1 Dataset

The hetrec2011-Movielens is an extension of the MovieLens10M dataset, made

available by the GroupLens research group[8]. It links the movies of MovieLens dataset with their corresponding web pages at Internet Movie Database[9] and Rotten Tomatoes movie review systems[10]. Only those users with both rating and tagging information in the original dataset were kept. In our study, we only use the rating data and movies' actor information. Our dataset contains 855,598 ratings given by 2,113 users to 10,197 movies. These movies have 95,321 actors. Data statistics is shown in Table 3.5. Besides ratings and actors, the dataset also contains other movie metadata such as genre, director and locations. It also has tag assignments given by users to movies. Note that this dataset contains ratings less than one. In [14], the authors removed ratings less than one from a similar dataset of EachMovie since they considered these ratings "not real ratings but represented users' impressions on items". However, we have kept them, since we believe these ratings reflect users' negative opinions on the items.

Table 3.5 Data statistics of Hetrec 2011 dataset

# users	2,113
# movies	10,197
# actors	95,321
Avg # actors per movie	22.778
# ratings	855,598
Avg # ratings per user	404.921
Avg # ratings per movie	84.637

[8] http://www.grouplens.org
[9] IMDb http://www.imdb.com
[10] http://www.rottentomatoes.com

3.3.2Baselines

Matrix Factorization (MF) and Probabilistic Matrix Factorization (PMF) [98] are implemented as two baselines. We use the Matlab implementation of PMF from its author. This implementation used batch gradient descent, which kept a fixed number of (user, item, rating) triples in the training set. To make algorithms comparable, we use the same protocol to split the training and test set and apply our algorithms. Specifically, we kept 800,000 triples in the training set, and left the remaining 55,598 triples in the test set. Given a movie and a user in the test set, we require that both the movie and all the actors of this movie should present in the training set.

3.3.3Evaluation tasks and metrics

We evaluated the performances of algorithms on two tasks. The first task is implicit rating prediction. In this task, the model only predicts whether a user will rate a movie or not. The second task is rating prediction, in which both implicit rating and the actual rating are predicted. The reason that we perform the implicit rating prediction task, is that for Actor-Topic model (ATM), all the actual ratings are coded as ones. This is because in ATM is essentially a "bag-of-movie" (ATM-I) or "bag-of-users" (ATM-II) model, i.e. only binary information (whether a user has watched a movie), can be incorporated. For the ATM+MF model, we tried using either implicit or explicit ratings for the ATM component.

We also applied two approaches to utilizing the rating information. Take the LDA-I model for example. In the first approach, we treated each rating that a user gives to a movie as the pseudo "frequency" for this movie. That means, the higher a user rates a movie, the larger its pseudo frequency is. Under this approach, each algorithm is named as "Algorithm_explicit". In the second approach, we encoded each rating that a user gives to

a movie as one, i.e. each movie only appears once in a user's collection. Under this approach, each algorithm is named as "Algorithm_implicit".

Different metrics are used to evaluate these two tasks. For the implicit rating prediction task, we reported the precision, recall, F1, and NDCG (Normalized Discounted Cumulative Gain). For the rating prediction task, we reported both the root mean square error (RMSE) and mean average error (MAE).

3.3.4 Parameter tuning

This section introduces our parameter tuning for ATM+MF model for the implicit rating prediction. In the first step, we used grid search to tune n_1(number of top actors to represent each movie), and n_2 (number of top actors to represent each user). We found out that these two parameters did not affect the recommendation performance. Thus, we set both n_1 and $n_2$to ten, and tuned the topic number$T$. Result showed that the best MAE and RMSE were reached when T equaled to 20 and 50. We therefore fixed T to 20 in our following experiment. Then, we tuned the learning rate γ, the weight of the MF regularizationλ_1, and the weight of the ATM regularizationλ_2. We found that their optimal values were 0.003, 0.02, and 0.01. However, we also noticed that when setting the weight of the ATM component to zero, the performance was better than that when setting it to 0.01. This indicated that the ATM component is not helpful for implicit rating prediction.

For the two baselines MF and PMF, we also tuned their parameters to obtain their best performances. The optimal parameter setting for PMF is as follows. The optimal learning rate is 10, regularization weight is 0.01, momentum is 0.8, and number of factors is 2. The parameter tuning process for explicit rating prediction is similar.

3.4 Results

For the implicit rating (top-N recommendation) task, the precision and recall for algorithms are shown in Figure 3.5. For rating prediction task, the RMSE and MAE are shown in Figure 3.6.

3.4.1 Implicit Rating Prediction

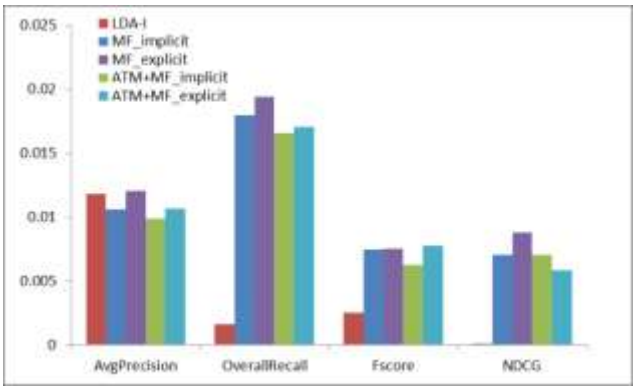


Figure 3.5 Result of implicit rating prediction

Results show that MF with explicit ratings performed best under all four metrics. This leads to the conclusion that users' actual ratings are helpful to predict their implicit preferences (i.e. whether they will watch a movie). LDA-I has good performance on average prediction, but performs worst on recall, F-score and NDCG. This indicates that LDA is able to find the most similar movies, but miss the majority. The two variations of ATM+MF models have comparable performance as MF. Moreover, ATM+MF with explicit ratings actually outperformed MF on F-score. Since F-score is a comprehensive score which considers both precision and recall, this shows the power of our ATM+MF model.

3.4.2 Explicit Rating Prediction

The performances of five algorithms in the explicit rating prediction task are shown in Figure 3.6. For MAE and RMSE, the lower the value is, the better it performs.

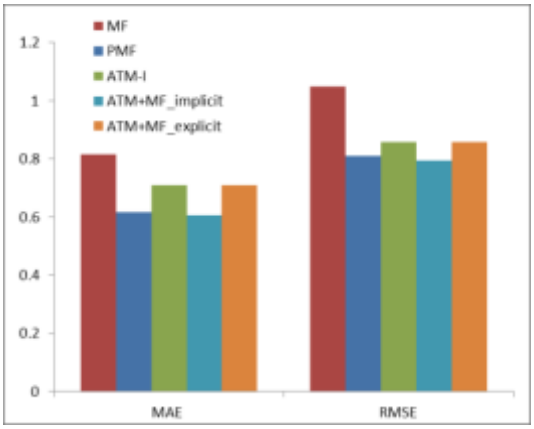


Figure 3.6 Result of explicit rating prediction

Results showed that, in this scenario, PMF and ATM+MF_implicit performed best, while MF perform worst. First, simple ATM outperform MF in this scenario. Recall that ATM algorithm uses actor information as input, but does not use the actual ratings. MF uses the ratings but does not have actor information. This result shows that movie casting information may be more valuable than the actual ratings in predicting the rating that a user will give to a movie. Second, comparing the two variations of ATM+MF, results show that ATM+MF_implicit outperforms ATM+MF_explicit. This means that ATM+MF which encodes ratings as binary information, outperforms the one uses actual ratings. This is contradictory to our expectation. We need further investigation on the reason of this result. One possible explanation is that implicit ratings contain sufficient information for rating prediction. The numeric values of ratings may incorporate noise and thus downgrade the

prediction performance.

3.5 Conclusion and Future Work

This study has shown that topic models can be used in recommendation tasks. In the implicit rating prediction task, they have comparable performances with the matrix factorization model. In explicit rating prediction task, actor-topic models outperforms MF. Since casting information is easily available for movies, these models can be helpful in real-world applications.

For future work, additional information, such as Actor co-star network can be incorporated into the actor-topic model. The co-star network can be constructed from the training dataset. If two actors act in the same movie, their co-star value will increase by one. Based on the assumption that co-star actors should have similar topic distributions, we can add another regularizer into the loss function of the actor-topic model.

Chapter 4. Content-based Hierarchical Music Recommendation

4.1Introduction

Nowadays, music is made available through a number of channels, including online and traditional ones. Music fans thus have access to an overwhelming amount of tracks in various types. Studies have shown that music artists' popularity follows the long tail [25]. Researchers have also found out that a large proportion of users have somewhat eccentric music tastes. They consume niche music and give high ratings to them [39]. This suggests that music recommenders should be developed, to help listeners discover niche music satisfying their unique tastes. This so called "Long tail" gained its fame in Anderson's popular book [6]. In this book, the long tail theory states that if numerous niche items are made available to the customers, retailers can make more profit.

However, the state-of-art music recommendation research focus on prediction accuracy, which is a common goal for item recommendations in other domains. Among them, collaborative filtering (CF) and metadata-based filtering can give the most accurate recommendations. Given a user, user-based CF recommends songs that her similar users have listened to. Item-based CF recommends songs similar to those she has listened to. In both CF approaches, user similarities or item similarities are computed based on the user feedback data (i.e. user-music implicit or explicit ratings). Metadata-based filtering also recommends music similar to those the user has listened to. Different from CF, such music similarity is based on music metadata, such as artist, genre, or age. Because of the way that music similarities are calculated, both CF and metadata-based recommendations tend to bias towards popular music or music from users' familiar artists. Besides academia, prediction accuracy is also a main objective in industry. For instance, the core component

of recommendation engine in Amazon is Collaborative filtering [74]. When a user searches for Beatles' White Album, results in the first six pages are their other albums[11] [24]. Although these recommendations are accurate, they may not be useful since users may have listened to them. It can be seen that accuracy and perceived quality (usefulness) are two contradictory goals. According to the aforementioned long tail phenomenon, we believe that users' perceived quality is a more meaningful objective than accuracy.

Since CF and metadata based content filtering generate accurate but familiar items, in this chapter, we try to propose a content-based recommendation algorithm by including acoustic features. The key of a content-based filtering algorithm is to measure the similarity between music. This is a challenging task. First, the simplest approach to measuring music similarity is to first represent each song into a feature vector, then calculate Euclidean distance between vectors. Such an approach is based on the assumption that music features are orthogonal and equally important. However, this assumption is unrealistic [108]. For instance, pitch, loudness and timbre all depend on the same signal structure [59]. The second challenge in measuring music similarity is that music similarity can be assessed by various criteria, including artist, genre, symbolic features such as tempo, melody, rhythm and timbre, harmony, perceptual aspects (energy, texture, beat), and even cognitive experience, to name a few [59]. Each user may have her unique criteria for similarity. Most existing content-based or hybrid music recommendation models only focus on one criteria. [82][35]. One additional challenge is that music has inherent hierarchical structures. Thus, music similarities should be measured on different scales/levels. Most previous research only look at one level [7]. One exception is [59]. In this thesis, Jehan proposed a method

[11] http://www.amazon.com/The-White-Album-Beatles/dp/B0025KVLU6/ref=dp_ob_title_music

to compute acoustic similarities from multiple aspects. Similarities on each aspect is computed recursively through dynamic programming. The music similarity measure of our content-based recommendation algorithm is mostly inspired by this study.

This chapter is based on our ECIR' 2014 paper and patent [32]. However, in that paper, our model is a two-level alignment model. The model was applied to the alignment of two subtitle files. In our patent, we only propose the hierarchical structure alignment model, with no application or experiment. In this chapter, we adapt hierarchical structure alignment model to computing music similarities, and use the result to music recommendation. We will focus on investigating how performances of music recommendation algorithms vary on songs with different popularities. We will first introduce our dataset and how we preprocess the data. Then, the music recommendation model is presented. In the section after, we show our Hierarchical Sequence Alignment (HSA) Model to measure music similarities. In the last two sections, we show the evaluation metric, baselines and recommendation results.

4.2 Dataset

We use the Million Song Dataset (MSD) to test our content-based music recommendation algorithm [10][12]. The main component of MSD is a collection of preprocessed audio features and metadata for one million popular tracks. These features and metadata are provided by The Echo Nest[13]. MSD also provide links to other types of datasets, including cover songs, lyrics, tags and user data. Moreover, it provides code to

[12] Million Song Dataset homepage. http://labrosa.ee.columbia.edu/millionsong/. Retrieved on Dec-19-2013.
[13] The Echo Nest. http://the.echonest.com/

fetch audio clips from 7digital.com[14]. A small subset of 10,000 songs is also provided.

In our experiment, our inputs include this 10,000-song MSD subset and the Taste Profile subset. The Taste Profile subset contains the play counts for 1,019,318 unique users on 384,546 unique songs. There are in total 48,373,586 user - song - play count triplets. All the songs in this dataset are contained in the large MSD dataset. A recent study pointed out that the Taste Profile subset was one of the largest publicly available collaborative filtering datasets [87].

4.2.1 Dataset preprocessing

Since not all songs in the Taste Profile subset is contained in the 10, 000 MSD subset, we first retrieve all the overlapping songs in the two datasets. We then extract the records of these common songs from the MSD subset, and the user-song-play count triplets for these common songs from the Taste Profile subset. The resulting dataset contains two components. The first component is the audio features of 3,666 songs. The other component is the user-song-playCount triplets given by 418,113 users to 3,666 songs. There are 772, 148 triplets in total. We use this dataset in our following experiment.

4.3 Music recommendation model

In traditional item recommendation, the task is to recommend items to each user. In our experiment, our main goal is to test how our algorithm performs on songs with different popularities, so we change the task to recommend users for each song.

Our content-based music recommendation model works as shown in Figure 4.1. Given songs in audio formats and user-song-play count matrix R, our goal is to recommend a list

[14] 7digital. http://www.7digital.com/

of users who are most likely to listen to i. The first step is to construct a similarity matrix S, where S_{ij} denotes the similarity between song i and song j. This is the core component of our algorithm. We will introduce it in detail in the sections below. Once S is constructed, the next step is to find from S the most similar songs with i. We denote this song set as I. In our experiment, we keep $|I| = N$. After that, we extract the user set U who listen to these songs in R. Finally, we predict the score that each potential user could have for the query song i. The score between song i and user u is calculated by the weighted average of the scores between each similar song i' with u. The weight of each song i' equals to its similarity with i. Once all the scores are computed, we recommend users who have highest scores with the query song but have not listened to it. Note that initially, for each user, we normalized her play count for each song by dividing it with her total play counts.

Input: Song i, all N song's acoustic features, user-song-playCount matrix M

Output: A list of users who are most likely to listen to i

Step 1: Construct a song similarity matrix S

Step 2: Find songs most similar with i, denote as I

Step 3: Find user set U who listen to these songs in R

Step 4: For each user u in U, predict the score u will have for song i

$$r_{i,u} = \sum_{i' \in I} sim(i, i') \cdot r_{i',u}$$

Step 5: Rank users in U by their scores and recommend top users with highest scores

Figure 4.1 Content-based Music recommendation algorithm

It can be seen that this recommendation algorithm is similar to user-based CF with user and item swapped. The difference lies in the normalization. In our approach, we first

normalize by dividing the total play count. In user-based CF, the predicted rating that u will give to item i is:

$$r_{u,i} = \bar{r}_u + k \sum_{u' \in U} sim(u, u') \cdot \left(r_{u,i} - \bar{r}_{u'}\right) \quad (4.1)$$

Where the normalization factor$k = \frac{1}{\sum_{u' \in U} sim(u,u')}$.

Our main contribution lies in how the music similarity matrix is constructed. In the next section, we propose a novel algorithm to measure music similarity, which considers both the hierarchical structure and variant aspects of music.

4.4 Model for computing Music Similarity

In this section, we will first introduce the hierarchical structure of music, and then propose our Hierarchical Sequence Alignment model (HSA) to compute the music similarity.

4.4.1 Hierarchical Music Structure

We use the hierarchical structure of music defined by Jehan and DesRoches in the documentation of the Echo Nest API [60]. Under this definition, each song is viewed as a four-level hierarchical structure according to its audio signal sequence. From top to bottom, these four levels are sections, bars, beats, and segments (Figure 4.2). On each level, a song is decomposed into a temporal sequence of audio pieces of that level. Each upper-level element consists of a sequence of lower-level elements.

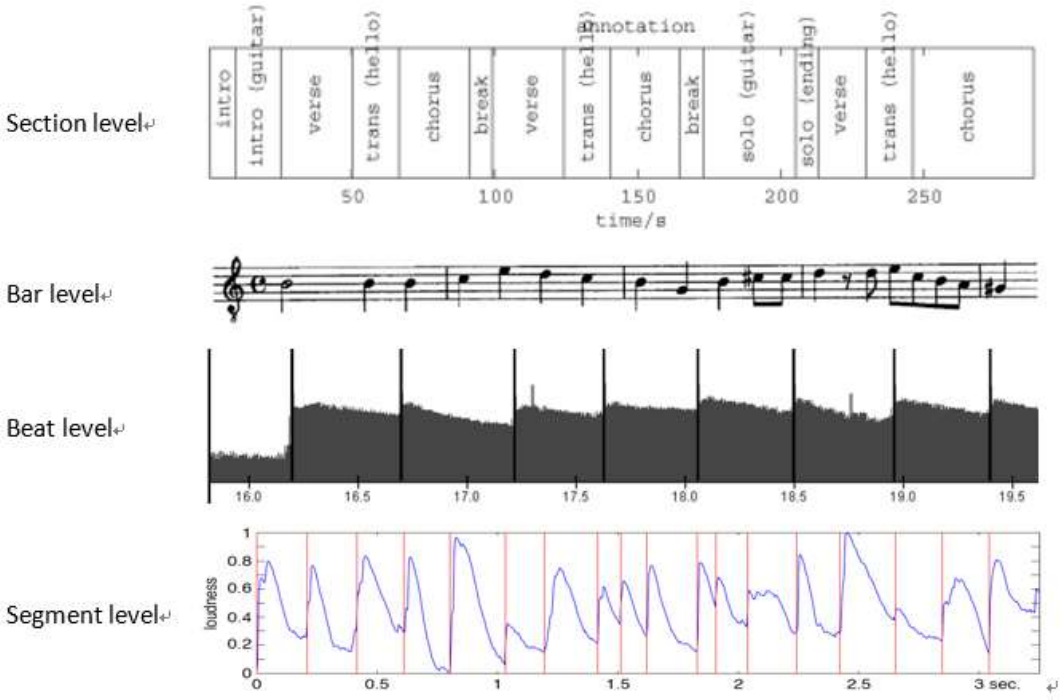


Figure 4.2 Music Hierarchical Structure (Section level: [21] Fig. 8 on P14; Bar level: Fig. 14 on P19; Beat Level: Fig. 4 on P7; Segment Level: [59] Fig. 3-12 on P53 middle figure)

As shown in Figure 4.2, on the Section level (fourth level), a song is a sequence of sections. Each section is defined by large variations in rhythm or timbre (chorus, verse, bridge, etc). On the Bar level (the third level), each bar is a subdivision of a section. On the Beat level, a beat is the subdivision of a bar and is the basic time unit of a song. On the lowest Segment level (First level), each segment is a subdivision of a beat, which represents a sound entity in terms of loudness, timbre and pitch [60].

In general, a K-level hierarchical structure Z is equivalent to a sequence of tree structures with depth K (we name it as a K-level tree structure). Each K-level tree structure is composed of two components: (1) its root element (denoted as$r(Z)$); (2) an ordered sequence of remaining $(K-1)$-level sub-tree belonging to the root (denoted

as$sub(Z)$), i.e.

$$Z = r(Z) + sub(Z) \tag{4.2}$$

It can be seen that, the sequence of remaining (K-1)-level sub-tree is also a (K-1)-level hierarchical structure. Such recursive property inspires us to apply dynamic programming to computing music similarities.

4.4.2Music Similarity via Hierarchical Sequence Alignment Model

Since each song is a sequence of K-level tree structures, the similarity between two music songs can be obtained by computing similarity between their corresponding sequences (Figure 4.3).

Sequence Alignment algorithm is a mainstream and efficient approach to obtaining sequence similarity. It originates from molecular biology, and arranges DNA, RNA, or protein sequences to identify similar regions between the sequences [85]. Needleman-Wunsch algorithm [86] and Smith-Waterman algorithm [110] [111] are the most widely used global and local alignment algorithms. Besides bioinformatics, sequence alignment has also been applied to approximate string matching and machine translation [20] [118].

We define the similarity between two music sequences, as their alignment similarity $SIM(K, X, Y)$ obtained from sequence alignment algorithm. Our objective is to find the optimal alignment to achieve maximum similarity between them. Let $\mathcal{L}(X, Y)$ denote an alignment between X and Y, which aligns the sequence of tree structures in X with the sequence of tree structures in Y. The optimal alignment is in the set of all feasible alignments $\{\mathcal{L}(X, Y)\}$. Therefore, the alignment similarity $SIM(K, X, Y)$ between music X and$Y$, given by sequence alignment algorithm, corresponds to the similarity between two K-level hierarchical structures X and Y with the optimal alignment. Hence, our

objective function is given as below:

$$SIM(K,X,Y) = \max_{\mathcal{L}\in\{\mathcal{L}(X,Y)\}} SIM_{\mathcal{L}}(K,X,Y) \tag{4.3}$$

where $SIM_{\mathcal{L}}(K,X,Y)$ denotes the similarity between X and Y according to the sequence alignment$\mathcal{L}$.

We will first introduce the Needleman-Wunsch algorithm [86] which is used to align two sequences of one level. Needleman-Wunsch algorithm can give optimal sequence alignment using dynamic programming. Given two sequences$X = x_1 \cdots x_M$, $Y = y_1 \cdots y_N$ with length M and N, this algorithm tries to find the optimal alignment between X and Y that maximizes alignment score by the equation below:

$$SIM(X_m, Y_n) = \max \begin{cases} SIM(X_{m-1}, Y_{n-1}) + sim(x_m, y_n), & x_m, y_n \text{ aligned} \\ SIM(X_{m-1}, Y_n) + g, & x_m \text{ aligned with a null} \\ SIM(X_m, Y_{n-1}) + g, & y_n \text{ aligned with a null} \end{cases} \tag{4.4}$$

$SIM(X_m, Y_n)$ is the maximum score for sequences $X_m = x_1 \cdots x_m$ and $Y_n = y_1 \cdots y_n$. $sim(x_m, y_n)$ represents the similarity between two elements x_m and y_n. g, the gap penalty, is designed to reduce the score when an element is aligned with a null.

In Needleman-Wunsch algorithm, one needs to know similarities $sim(x_m, y_n)$ between any two elements from two sequences respectively. In our case, each element in the sequence is a k-level tree structure ($1 \le k \le K$). Therefore, according to the composition of one hierarchical structure defined in (4.2), the similarity between a pair of two k-level tree structures x_m and y_n depends on two factors: the feature similarity between the root elements of two k-level structures, denoted as $sim_f(x_m, y_n)$, and the optimal alignment similarity between two $(k-1)$ -level sub-sequences $sub(x_m)$ and $sub(y_n)$ with the optimal alignment, that is, $SIM\big(k-1, sub(x_m), sub(y_n)\big)$. i.e.

$$sim(x_m, y_n) = sim_f(x_m, y_n) + SIM\big(k-1, sub(x_m), sub(y_n)\big) \tag{4.5}$$

For instance, the similarity between two beats, is defined as the similarity between the inherent feature vectors of these two beats, and the similarity between their segment sequences. Given that:

$$SIM(k, X_m, Y_n) = \max \begin{cases} SIM(k, X_{m-1}, Y_{n-1}) + sim(x_m, y_n), & x_m \text{ and } y_n \text{ aligned} \\ SIM(k, X_{m-1}, Y_n) + g, & x_m \text{ aligned with a null} \\ SIM(k, X_m, Y_{n-1}) + g, & y_n \text{ aligned with a null} \end{cases}$$

(4.6)

By plugging in (4.5) into(4.6), we have the final equation (4.7) below.

$$SIM(k, X_m, Y_n) =$$

$$\max \begin{cases} SIM(k, X_{m-1}, Y_{n-1}) + \left(sim_f(x_m, y_n) + SIM\left(k-1, sub(x_m), sub(y_n)\right)\right), x_m, y_n \text{ aligned} \\ SIM(k, X_{m-1}, Y_n) + g, & x_m \text{ aligned with a null} \\ SIM(k, X_m, Y_{n-1}) + g, & y_n \text{ aligned with a null} \end{cases}$$

(4.7)

Note that when k = 1, we do not have$SIM\left(k-1, sub(x_m), sub(y_n)\right)$. Since this is a recursive function, so we can apply Dynamic Programming to this problem.

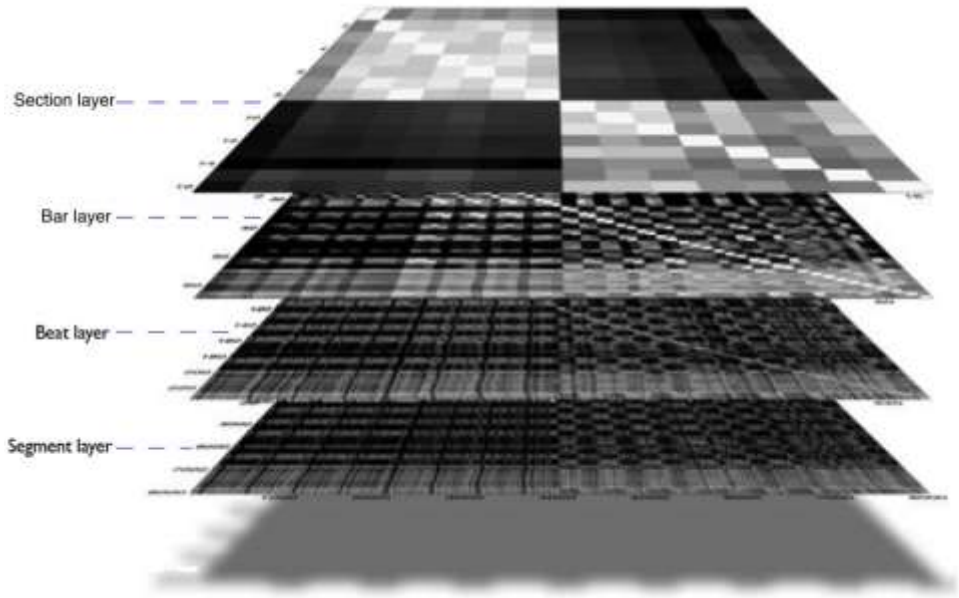


Figure 4.3 Hierarchical similarity representation (adapted from [59])

To compute the similarity $sim_f(x_m, y_n)$between the features vectors of root elements of two k[th]-level elements x_m and y_n, we first compute their Euclidean distance, and then transform the distance to similarity via an exponential function proposed by [105].

$$sim_f(x_m, y_n) = e^{-d(f_{x_m}, f_{y_n})} \tag{4.8}$$

$$\text{Where } d(f_{x_m}, f_{y_n}) = \|f_{x_m} - f_{y_n}\|_F$$

Our work is mainly inspired by Tristan's work [59]. The difference lies in that, in his work, for the segment level of music structure, he defines three classes of music similarity, including timbre similarity, pitch similarity and loudness similarity. In our work, we define a holistic similarity. Features from timbre, pitch and loudness are all viewed as a song's features. In the next section, we will introduce the features we use for each level.

4.4.3 Music feature on each level

Most features have been described in the Echo Nest Analyzer Documentation [60]. Features on the segment level include three types: loudness, timbre and pitch. Specifically, loudness information is given by three features: loudness at onset, loudness value at peak and time offset of peak loudness. Timbre is the quality of a sound that characterize certain types of musical instruments or voices. The Echo Nest Analyzer represents timbre into a 12-dimension vector as abstractions of the spectral surface, ordered by degree of importance. Similarly, pitch content is also represented as a 12-dimension vector. Each dimension corresponds to one of the 12 pitch classes from C, C#, D to B. The value on each dimension ranges from 0 to 1, describing the relative dominance of the corresponding pitch in the chromatic scale. On each level, we also consider two additional features, the

start time and confidence of elements. The features and feature numbers on each music level is shown in Table 4.1.

Table 4.1 Features and feature numbers on each music level

Section Level (2 features)	
sections_start	1
sections_confidence	1
Bar Level (2 features)	
bars_start	1
bars_confidence	1
Beat Level (2 features)	
beats_start	1
beats_confidence	1
Segment Level (29 features)	
segments_loudness_max	1
segments_loudness_max_time	1
segments_loudness_start	1
segments_pitches	12
segments_timbre	12
segments_start	1
segments_confidence	1

4.5 Experiments

We run experiments for our Hierarchical Sequence Alignment (HSA). We also use the Bottom level (segment level) alignment (denoted as BTMLYR) and CF as our baselines. We group songs by the number of listeners, and evaluate the recommendation performance for each song group. Since our content-based algorithm can potentially identify niche songs which are not popular, we assume that our algorithm (both HSA and BTMLYR) can outperform CF on “cold songs” with few play counts. Besides, since BTMLYR is a special case of HSA, we also assume that HSA should outperform BTMLYR. We will test these two hypotheses in our experiment.

H1: CF has better recommendation performance on hot songs than on cold songs.

H2: BTMLYR has better recommendation performance on cold songs than on hot songs.

H3: HSA has better recommendation performance on cold songs than on hot songs.

H4: HSA outperforms BTMLYR for all songs.

4.5.1 Evaluation metric

For each song, we use $precision@\,k$ and $recall\ @k$ to evaluate the performances of our HSA algorithm and baselines. k is the cut-off value at which we cut our result by considering only top-k recommendations. In our experiment, k is assigned to 1, 5, 10, 20, 50 and 100. The equation of precision and recall are given below.

$$precision = \frac{|\{recommended\ users\}| \cap |\{relevant\ users\}|}{|\{recommended\ users\}|} \tag{4.9}$$

$$recall = \frac{|\{recommended\ users\}| \cap |\{relevant\ users\}|}{|\{relevant\ users\}|} \tag{4.10}$$

where $|.|$ denotes the number of users in the set.

We group items by their popularities (i.e. represented by the number of listeners). Then we evaluate our algorithm and baselines for each group.

4.6 Results

We plot the precision of item-based CF, BTMLYR, and HSA for each song group. Since the song popularity follows the power-law (Figure 4.4), songs are grouped into six classes: (0-100], (100-200], (200-400], (400-800], (800-1600] and (1600-2817], denoting the rank of the song evaluated by the number of listeners. For instance, group "(0-100]" contains the top-100 songs with the highest number of listeners. In all the figures in this section, x axis is the "Item Rank", which is the rank of song popularity.

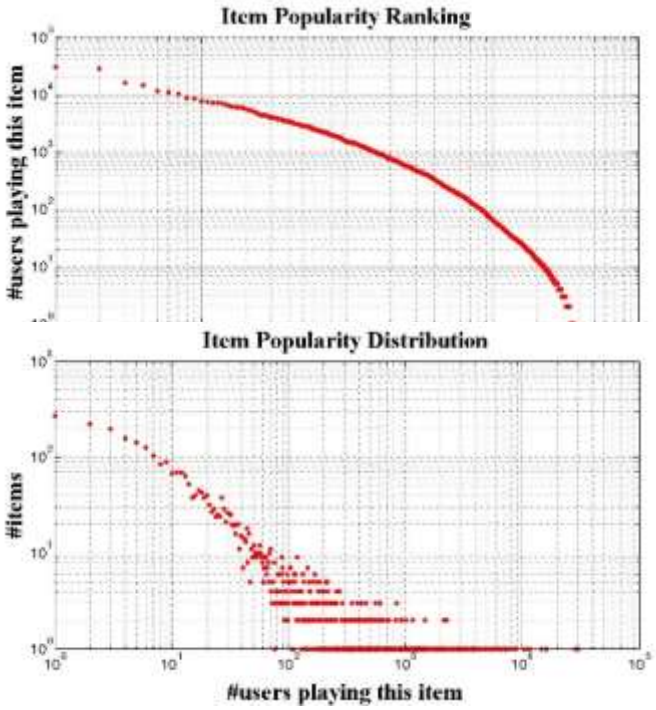


Figure 4.4 Power-law of item-popularity

4.6.1 Item-based CF

From Figure 4.5, we can see that for CF, except for $precision@1$, average $precision@k$ decreases as the item popularity decreases. In other words, CF has the best performance for hot songs. As the song popularity goes from "hot" to "cold", its performance degenerates quickly. This result is consistent with our hypothesis *H1*, proving that CF has bias towards popular items, and does not perform well on niche items.

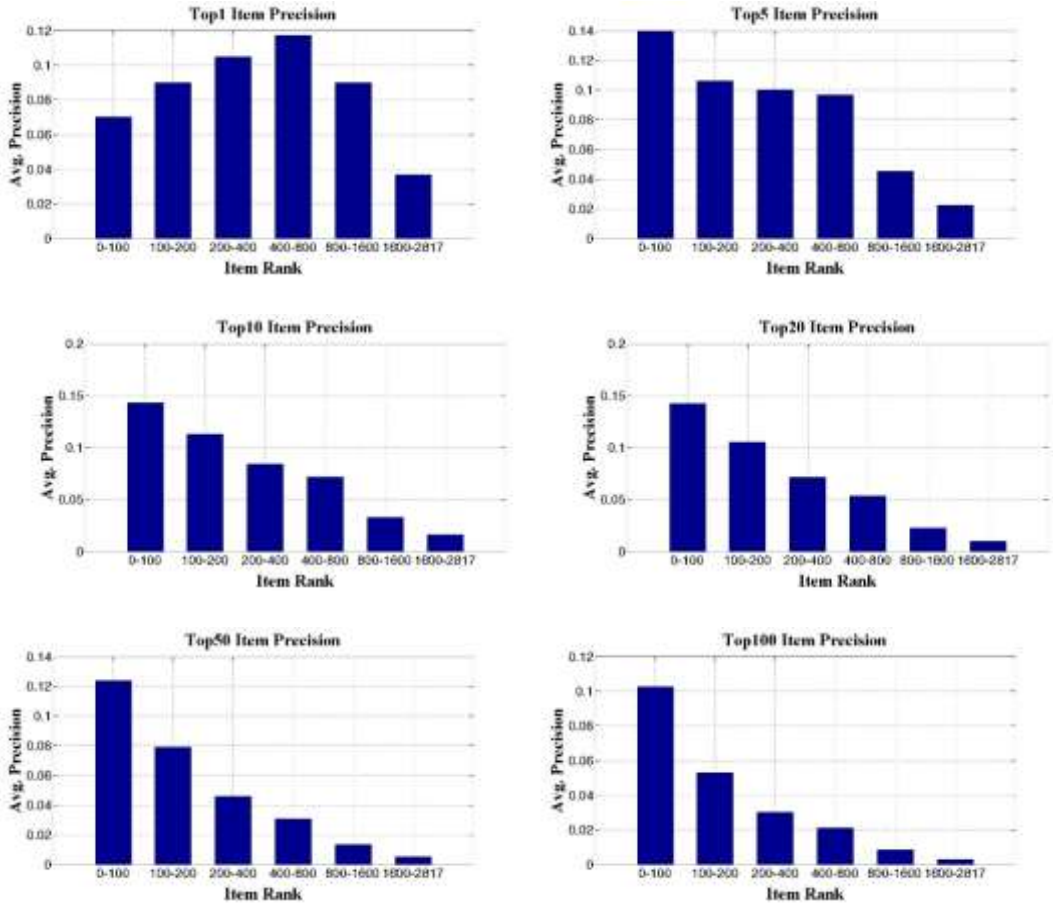


Figure 4.5 Precision of item-based CF by item Popularity

From Figure 4.6, we can see recall shows a reverse trend with precision for CF. This is consistent with the fact that when precision decreases, recall increases. Average $recall@k$ increases as the item popularity decreases. In other words, CF has the best recall for cold songs but worst recall for hot songs. The recall on popular songs is low because the most popular songs have large numbers of listeners, and it is difficult to find out them all.

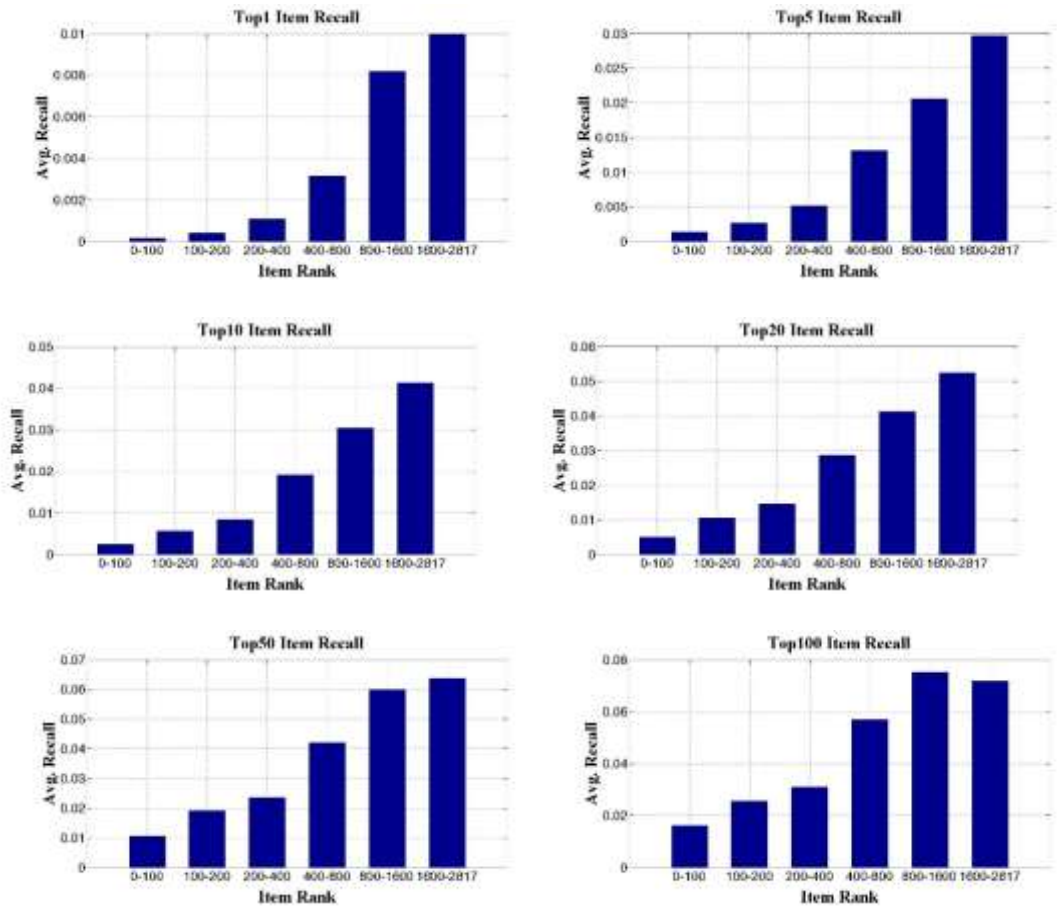


Figure 4.6 Recall of item-based CF by item Popularity

4.6.2 Bottom-layer alignment algorithm (BTMLYR)

From Figure 4.7, we can see that our single-layer alignment algorithm shows a different result from CF. The result is two-folds. First, similar with CF, Bottom layer algorithm generates the highest precision for the top 100 popular songs. However, for the other groups, average $precision@k$ does not decrease as the song popularity moves from hot to cold. When recommending the top-1, 5, and 10 users, BTMLYR has much higher precision for songs with rank between 200 and 400, than songs with higher rank between 100 and 200. When recommending the top-20, 50, and 100 users, BTMLYR algorithm has the second highest precision for songs with rank between 400 and 800, than songs with

higher rank. In other words, except for the most popular songs, our Bottom-layer alignment algorithm has better performance when recommending songs with medium popularity. Since these songs are not the most popular one, our algorithm is able to give recommendations with both novelty and relevance. Although *H2* does not hold, our content-based algorithm does have promising result on cold items.

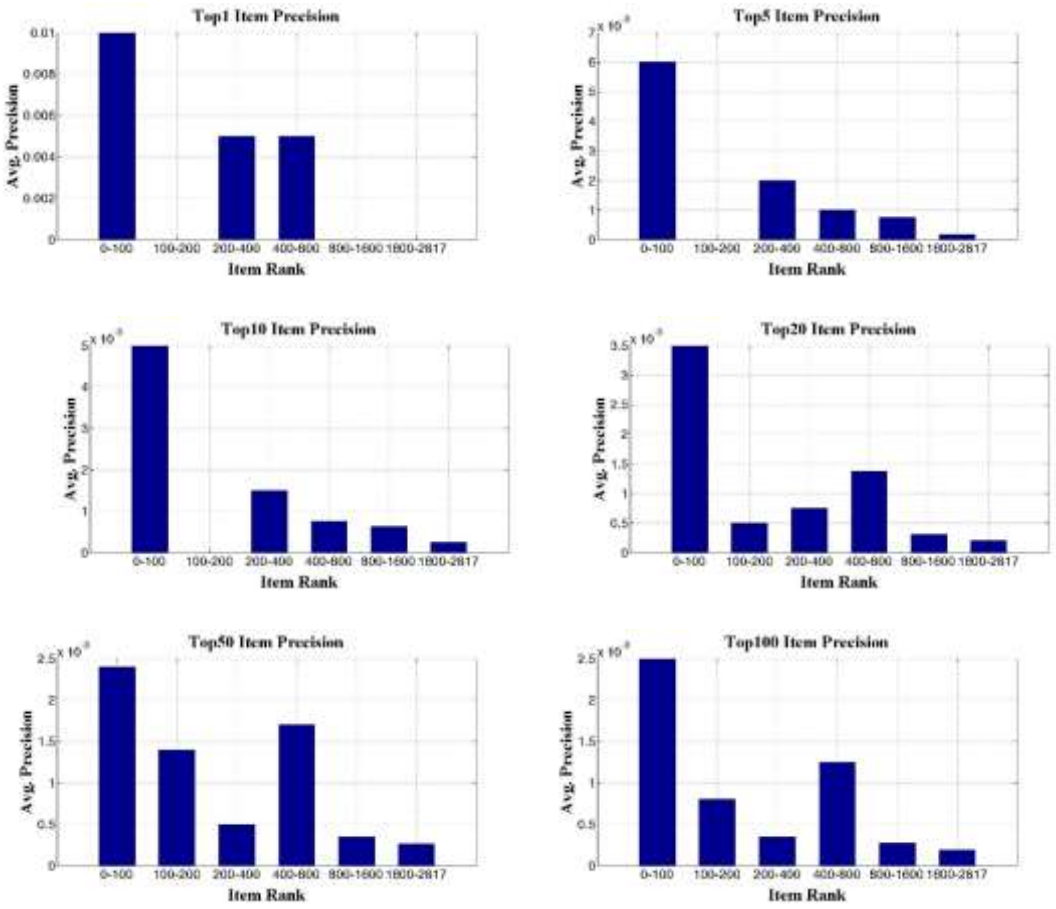


Figure 4.7 Precision of Bottom-Layer user Recommendation by item Popularity

From Figure 4.8, we can see that the recall of our single-layer alignment algorithm also shows a different pattern from CF. First, average $recall\ @k$ does not increase steadily as the song popularity moves from hot to cold. In particular, our single-layer algorithm

does not generate the worst recall for the most popular songs. Moreover, the best recall is not achieved on the coldest songs. Instead, songs with middle level of popularity often generates the best recall for different k values. Except for the scenario when k = 100, the best recall is obtained for songs with ranks between 400 and 1600. This finding is consistent with its precision performance. In a word, our BTMLYR algorithm has better performance in both precision and recall when recommending songs with medium popularity. Since these songs are not the most popular one, our algorithm is able to give recommendations with both novelty and relevance.

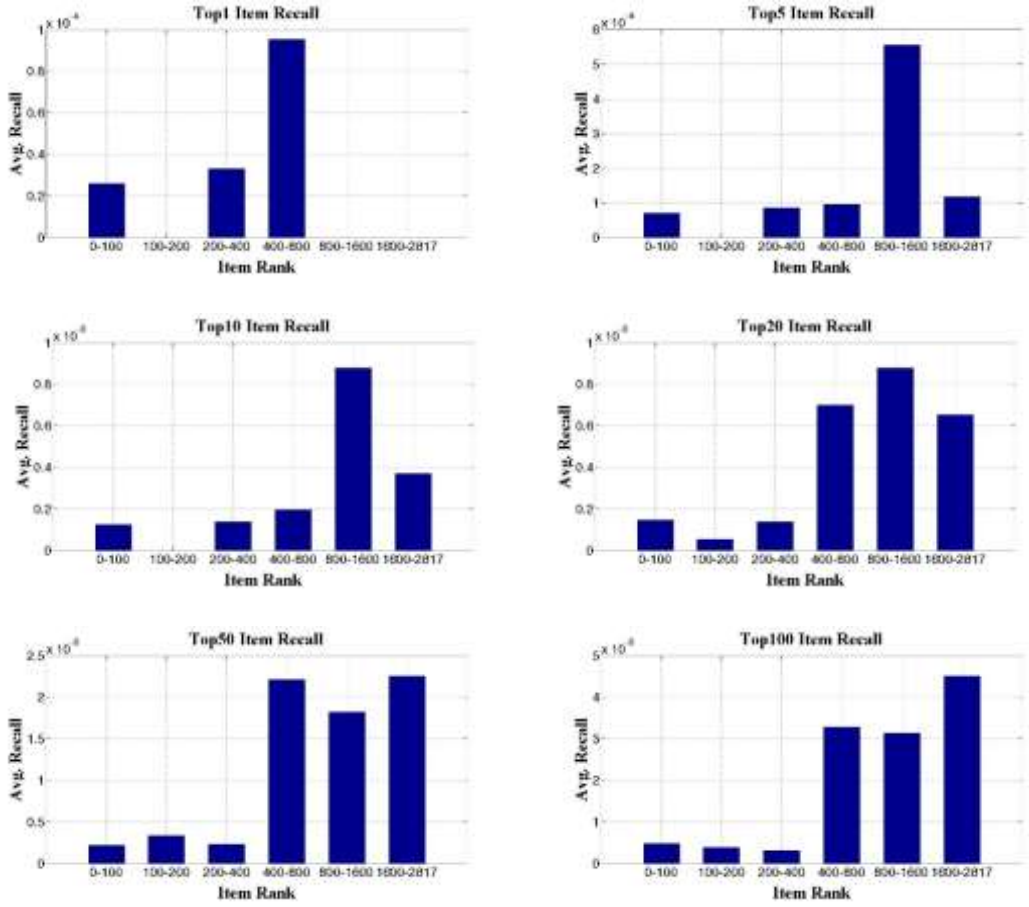


Figure 4.8 Recall of Bottom-Layer user Recommendation by item Popularity

4.6.3 Hierarchical Sequence Alignment (HSA)

Figure 4.9 shows that our Hierarchical Sequence Alignment (HSA) algorithm has a similar performance with CF on precision. Except for $precision\ @1$, average $precision\ @k$ decreases as the item popularity decreases. This indicates that HSA has the best performance for hot songs. As the song popularity goes from "hot" to "cold", its performance degenerates quickly. This result is contradictory to our hypothesis *H3*, where we assume HSA should have a similar result with the Single layer algorithm, since the latter is a special case of HSA. Further investigation will be done as our future work.

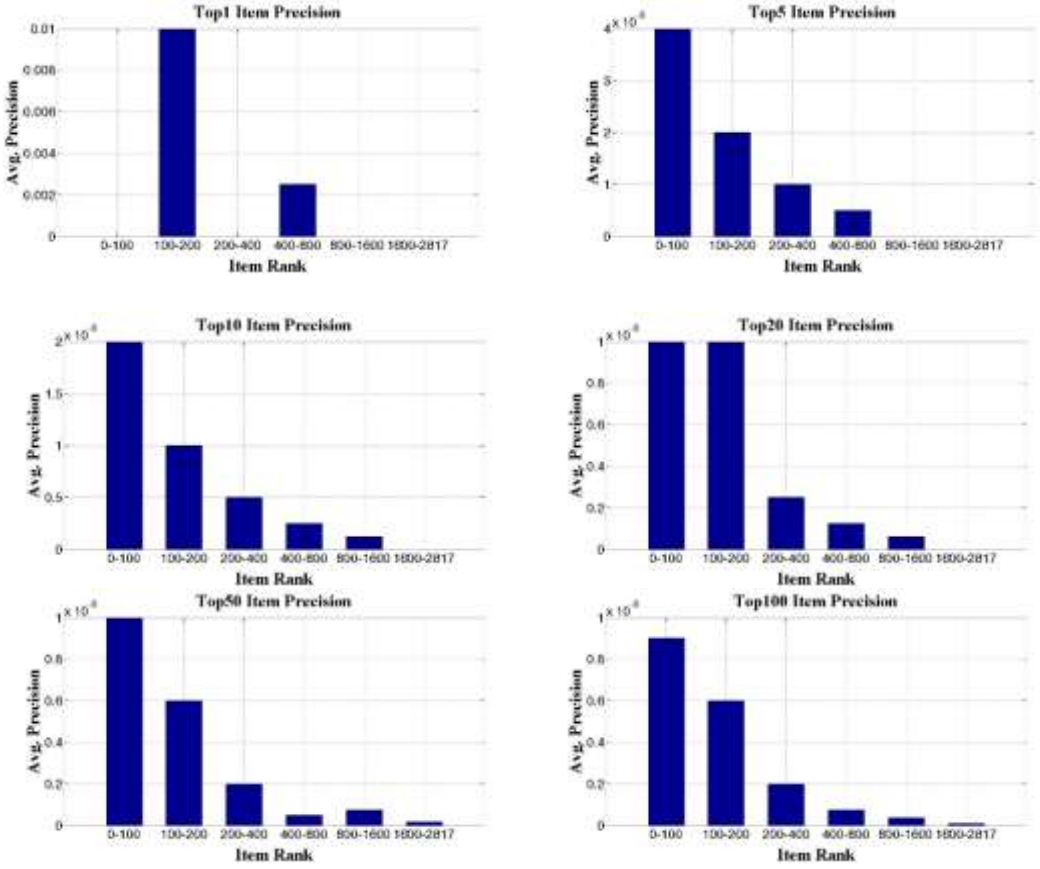


Figure 4.9 Precision of HSA by item Popularity

The recall of our Hierarchical Sequence Alignment (HSA) algorithm shows a different

pattern from both CF and the single layer algorithm (Figure 4.10). Compared to CF, there is no obvious trend that as the song popularity decreases, recall increases. Instead, recalls for songs with different popularity levels do not vary much. Compared to the single layer algorithm, HSA has the worst recall for songs with mid popularity while has better recall for songs with either highest popularity or lowest popularity. This is particularly true when recommending top 20, 50 and 100 users to the song. We also leave the explanation of this finding as our future work.

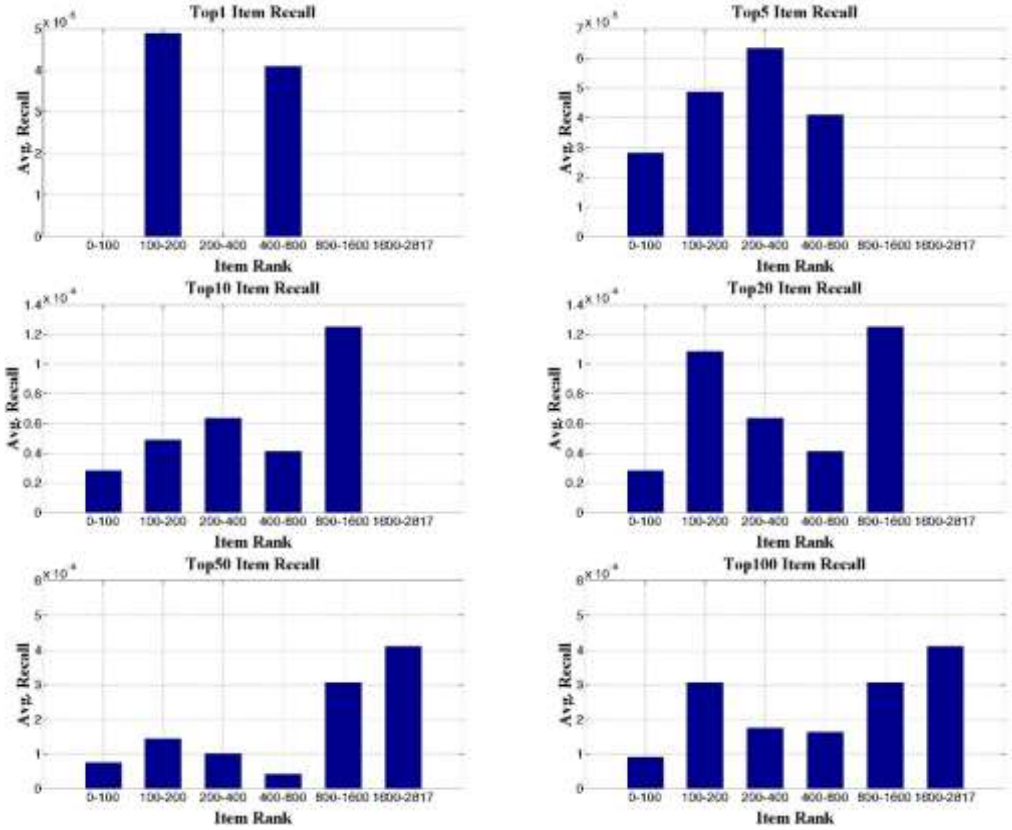


Figure 4.10 Recall of HSA by item Popularity

4.6.4 Comparison between BTMLYR and HSA

In this section, we further compare the recommendation performances of our single

layer algorithm BTMLYR and the multi-layer algorithm HSA.

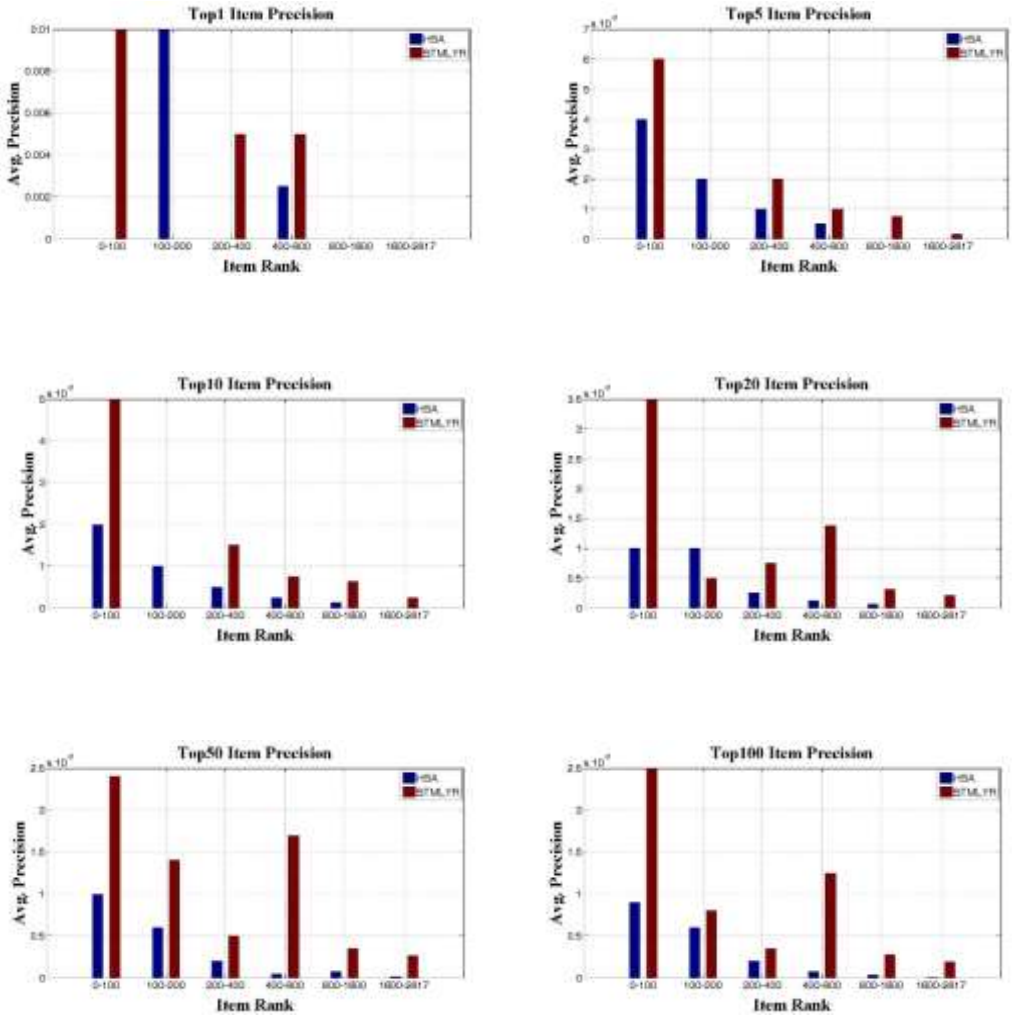


Figure 4.11 Comparison between BTMLYR and HSA on precision

It can be seen from Figure 4.11that BTMLYR outperforms HSA on almost all items with different popularity ranks. Such difference in performance is most obvious for cold items (with a popularity rank from 400 and higher). We believe the reason is that the parameters in HSA have not been set to its optimal.

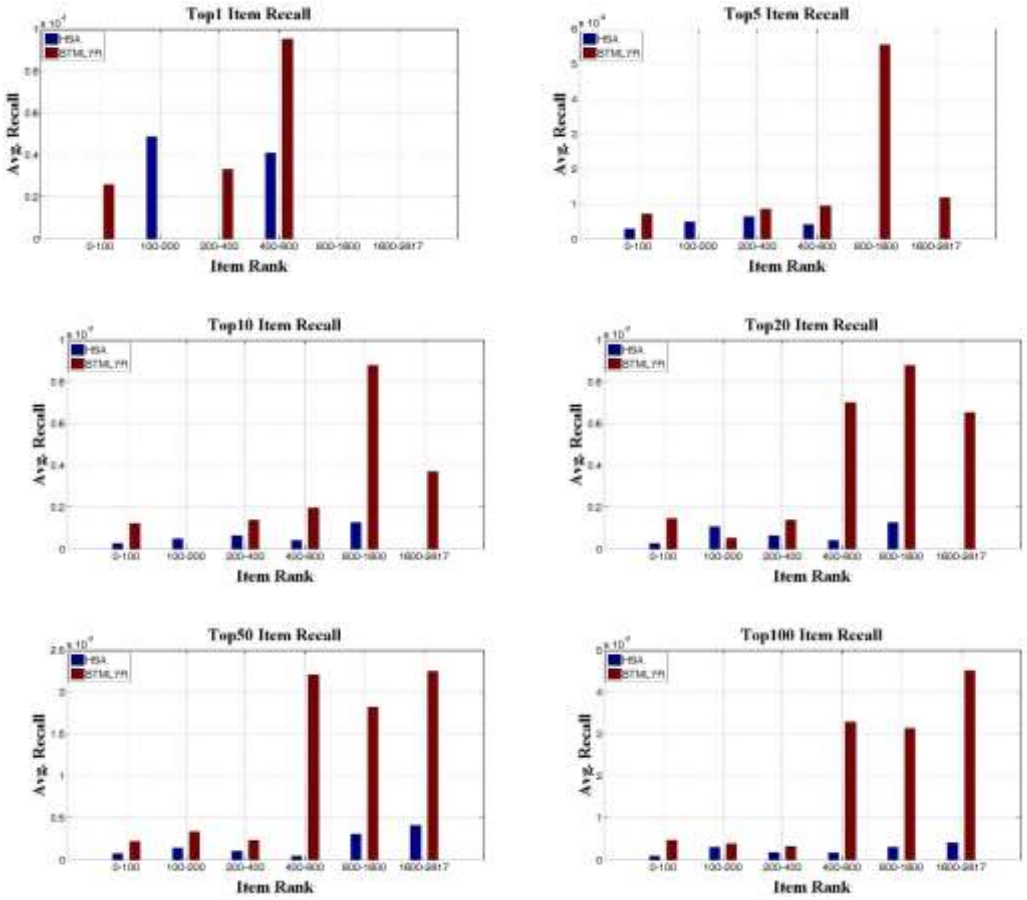


Figure 4.12 Comparison between BTMLYR and HSA on recall

For recall, the result is similar. As shown in Figure 4.12, BTMLYR still outperforms HSA on almost all items with different popularity ranks. Such difference in performance is most obvious for cold items (with a popularity rank from 400 and higher). We believe the reason is that the parameters in HSA have not been set to its optimal.

4.7 Conclusion and Future work

Results show that CF does perform poorly on cold songs as we expected. Besides, single layer alignment algorithm BTMLYR has comparable performance on cold songs, indicating that this algorithm is able to give recommendations with both novelty and relevance. However, to the contrary of our expectation, Hierarchical sequence alignment

algorithm (HSA) also performs better on popular songs than cold songs. Finally, single layer alignment algorithm BTMLYR outperforms HSA. Since BTMLYR is a special case of HSA, its performance should be bound to that of HSA. Hence, we believe that BTMLYR beats HSA because that the parameters in HSA have not been set to its optimal. Given that HSA has many parameters, the search space is so large that grid search would not work. In the future, we need to find an automatic algorithm to tune the parameters in HSA. Our result once again show that content-based cannot beat CF in music recommendation. Another direction is to develop a hybrid recommendation algorithm which combines our HSA model with memory-based CF or latent factor models. In our experiment, we also find that our algorithm has different recommendation performance on different music genres, so it is interesting to analyze how genre is related to the recommendation performance. Finally, since our algorithm could inherently perform well on niche items, precision and recall may not be the most appropriate evaluation metrics. User studies need to be conducted to evaluate the results.

Chapter 5. Comparison between familiarity network and similarity network

User-based collaborative filtering (CF) recommends items to a user based on those items similar users have consumed. User similarity can be defined either by their taste similarity or by their friendship. In this chapter, we compare a similarity-based network and a familiarity-based network. In the similarity-based network, if the similarity between two users' tastes pass a threshold, an undirected edge is constructed between them. The familiarity-based network is a social network. In our study, we use a "Twitter-like" follower-followee network. An edge from u_i to u_j is constructed if u_i follows u_j. Our motivation of this comparison is based on the assumption that if there are significant difference between the structure of the two types of networks, social network could provide additional information to user similarities. Furthermore, social network can be utilized to boost the user-based CF recommendation.

In this chapter, we make two contributions. First, we compare the structures of the "familiarity-based" and "similarity-based" networks in the same system. Second, we find out that users' followees and their interests sharers do not overlap. This finding is different from those of previous studies. Thus, it is possible to recommend friends to users based on similar reading patterns.

The remainder of this chapter is organized as follows. Section 5.1 introduces our motivation, and the problem of comparing familiarity and similarity networks. Section 5.2 presents our dataset. Section 5.3 introduces our method of comparison. In Section 5.4, we reports the comparison results on macroscopic, microscopic and local network structure

between two types of networks. We also present the multiplexity testing result between follower/followee relations and the co-reading relations. We conclude this chapter and propose future work in section 5.5. This chapter is based on our GrC 2011 paper [58].

5.1 Introduction

Automatic recommender systems have proliferated in recent years. They can recommend products (Amazon.com), metadata (Delicious), items, including books, music, and movies (Last.fm), or friends. The mechanism for most recommending systems is collaborative filtering.

Collaborative filtering can be divided into item-based CF and user-based CF. User-based CF recommends items to a user based on those items her similar users have purchased. The underlying assumption is that users prefer items that similar users prefer [76]. User similarity can be defined either by the similarity of two users' tastes or by their friendship. Researchers have shown contradictory opinions towards the problem that the similarity-based network and familiarity-based network are similar or different. Some researchers assumed that "the tastes of one's friends may vary significantly" [79]. If this assumption is true, then the similarity-based network will be different from the friendship network. However, the authors did not make a further step to test their assumption. [1] was one of the few studies which compared two relationships empirically. They found out that the friendships and the relations formed by interest sharing have correlations. However, [1] only focused on one particular online community, and thus their findings could be unique to their dataset. We aim at testing their conclusion in other systems.

In this chapter, we analyze the familiarity-based network and the similarity-based network of an online recommender system. Specifically, two research questions are

proposed. First, what is the difference between the familiarity and similarity network in terms of network structures, local structures and common users? Second, do one's followers and her "interest-sharers" overlap? We found out that the two networks were different in both macro and micro perspectives. Moreover, users' followers and their interest sharers did not overlap. We concluded that social networks could provide additional information to the user similarities, and thus could be helpful in recommendation. In this regard, this chapter can be viewed as a preliminary step towards the next Chapter, Recommendation with user networks.

5.2 Dataset

5.2.1 Data collection

Douban[15] is a Chinese web site founded in March, 2005. It is a platform for users to rate and review different movies, books and music. Recommendation services are provided based on users' behaviors. Different from IMDB[16] and Lastfm[17], Douban comprehensively comprise multiple types of entertaining items. Users can give ratings to movies, books and music from 1 to 5. It also supports several forms of user interactions. For instance, it allows users to follow other users, forming a "Twitter-like" social network.

In this study, data are collected from an "ego network" on Douban. Specifically, we first selected a seed user u_s. Then,u_s's followees and the followees of her followees are crawled. All these users constitute the nodes of the follower-followee social network (denoted as F). All the follower-followee relations between any of these two nodes

[15] Douban. http://www.douban.com
[16] Internet Movie Database(IMDB). http://www.imdb.com/
[17] Last.fm.http://www.last.fm/

constitute its edges. F is a directed network. After preprocessing, the final network in our dataset contains 1,080 users. In order to construct a similarity-based network, we also crawl each user's book collection. Since a user can mark a book into read, reading and wish, the collection entries which are marked as "read" are extracted. The user-book matrix contains 73, 573 books. In addition, we crawled users' location information from their profiles.

5.2.2 Network construction

As mentioned previously, familiarity network and similarity network are built and compared in this study. The familiarity network F is constructed as follows. If u_b follows user u_a (by adding u_a into her followee list), then an edge is added from u_b tou_a. The original dataset contains ones self-loop, indicating one user follows herself/himself. We removed this loop. The similarity network$C$, which represents co-reading relationship between users, is transformed from user-book matrix. We adopted two different methods to calculate user similarities for the network construction. First, two users' similarity equals to the co-occurrence count between their row vectors. In this method, and edge exists between two users if they have read at least one book in common (threshold = 1). We will denote this occurrence-based coreading network as $C_{cooccur}$. In the second method, two users' similarity equals to the cosine similarity between their row vectors. An edge exists between two users if their cosine similarity is larger than a threshold (we set it to 0.01), and 0 otherwise. We will denote this similarity-based coreading network as C_{sim}.

We find out that the average degree of the familiarity network F is 1.38, which is much lower than those of two similarity networks (both larger than 400). Thus, we tune the thresholds co-occurrence and the similarity in $C_{cooccur}$ and C_{sim} to make the average degree of the similarity networks and the familiarity network comparable. After tuning, the

optimal threshold for co-occurrence in $C_{cooccur}$ is 52. The optimal threshold for user similarity in C_{sim} is 0.169. We use these two thresholds to construct $C_{cooccur}$ and C_{sim}, and use them in our experiment.

5.3 Method

5.3.1 Network comparison (RQ1)

According to 5.2.2, three networks are constructed - one familiarity network denoted as F, and two similarity networks denoted as $C_{cooccur}$ and C_{sim} respectively. To solve the first research question, the three networks are analyzed and compared from three levels. On the macroscopic level, the indicators of the network structure, including small-world property, centralization, and correlations between different centrality measures, are analyzed. On the microscopic level, the top-20 users for each centrality measure in three networks are compared. Finally, blockmodeling is conducted in the local level analysis.

For the macroscopic level, we quantify the small-world property of each of our network by comparing its clustering coefficient and average path length to an equivalent random network with same degree distribution. A small-world network should have small average geodesic distance and large clustering coefficient [115]. Defined formally, to measure centralization, we first calculate the sum in differences in centrality between the most central node in a network and all other nodes; then divide it by the theoretically largest such sum of differences in any network of the same size. Give a centrality measure x, its corresponding centralization is:

$$C_x = \frac{\sum_{i=1}^{N} C_x(p_*) - C_x(p_i)}{max \sum_{i=1}^{N} C_x(p_*) - C_x(p_i)}$$

$C_x(p_i)$ is any centrality measure of node i. $C_x(p_*)$ is the largest value of $C_x(p_i)$.

$max \sum_{i=1}^{N} C_x(p_*) - C_x(p_i)$ is the largest sum of differences under centrality x for any graph of with the same number of nodes.

For the microscopic level, we measure three types of centrality, degree centrality, closeness centrality and betweenness centrality. Degree centrality is defined as the number of ties that a node has. A directed network has two types of degree centralities, indegree and outdegree. Indegree is the number of ties directed to the node while outdegree is the number of ties that a node directs to others[18]. The closeness of a node is defined as the inverse of the sum of its distances to all other nodes [97]. Betweenness centrality measures the number of times a node acts as a bridge along the shortest path between two other nodes [36]. The centralization of any network measures how central its most central node is in relation to how central all the other nodes are [37].

For the local level, we measure the structural equivalence and regular equivalence. Two nodes are defined as structurally equivalent if they have the same relationships to all other nodes. Two nodes are said to be regularly equivalent if they have the same profile of ties with members of other sets of nodes that are also regularly equivalent [49].

We perform several blockmodelings with different numbers of blocks are run to find the "best fit" model. First, structural equivalence blockmodel with random initial blocks is applied. Block modeling with two, three, four blocks are applied in sequence. The blockmodel with the lowest error score is selected. However, when a model with slightly larger error score is easier to interpret, the latter is chosen as the final model. Then, regular equivalence blockmodels with different number of blocks are applied. Finally, users' location information is used as initial "blocks" and blockmodelings are run again to

[18] http://en.wikipedia.org/wiki/Centrality

optimize it.

We use Pajek for structural equivalence-based blockmodeling, which implements an "exploratory" approach [30]. As "exploratory blockmodeling" is inappropriate for sparse networks, isolates are removed from two co-reading networks. After removal, the co-occurrence based co-reading network has 180 nodes. The similarity-based co-reading network has 305 nodes. Since Pajek does not allow blockmodeling be applied to networks larger than 256 nodes, for the co-reading network with more than 256 nodes, UCINET [16] is used instead. Finally, we use Fruchterman Reingold algorithm [38] for visualization.

5.3.2 Multiplexity of social relations (RQ2)

In this part, we try to answer the research question that if a user's friends and his "interest-sharers" overlap. 39 users who have outlinks in social network F are chosen as the target group. The multiplicity is computed as the cosine similarity between one user's vectors in two networks (familiarity network F, and one of the similarity network C). Each user's vector in a network equals to its row vector, where the values are the edge weights.

5.4 Results

5.4.1 Macroscopic Network structure

5.4.1.1 Small-world property

In this section, we compare the network properties of the similarity networks $C_{cooccur}$ and C_{sim} with the familiarity network F. The results are shown in Table 5.1. Lrepresents average geodesic distance. CCrepresents clustering coefficient. Recall that a small-world network should have small average geodesic distance and large clustering coefficient [115]. From Table 1 we can see that, the average geodesic distance of three

networks are much smaller than those of the corresponding random networks. Their clustering coefficient is more than 10 times larger than those of the random networks. Thus, it can be concluded that all three networks have the small-world property.

Table 5.1 Macroscopic Network Structures for Three Networks

	F	$C_{cooccur}$	C_{sim}
N	974	974	974
Arcs/Edges	1340	684	669
degree	1.38	1.41	1.37
Density	0.001	0.001	0.001
Diameter	6	6	12
L	2.981	2.517	4.284
L of random	11.93	15.66	12.54
CC	0.061	0.088	0.077
CC of random	0.0018	0.0012	0.0014

5.4.1.2 Centralizations

Table 5.2 Centralizations of Three Networks

	F	$C_{cooccur}$	C_{sim}
degree	0.083	0.112	0.063
indegree	0.018	NA	NA
outdegree	0.160	NA	NA
closeness	0.456	*	*
inclose	*	NA	NA
outclose	*	NA	NA
betweenness	0.021	0.038	0.022

* For the social network$F$, both in-closeness and out-closeness centralities cannot be computed since this network is not strongly connected. For both co-reading networks$C$, closeness centrality cannot be computed since the network is not weakly connected.

It can be seen from Table 5.2 that the closeness centralization of familiarity network F is quite high. This means that on average, the users in this network are close to each other. This aligns with our expectation, because F is an ego network for a seed user. Besides, outdegree centrality is also high. This means that only a few users follow many others, whereas most users only follow a few. The indegree centrality is low, which indicates that users have few followers on average. The standard deviation of indegree centrality is also low, which means that users' indegrees are more uniformly distributed. They have similar number of followers.

It is also worth noticing that, $C_{cooccur}$ has much larger degree centrality thanC_{sim}. The reason lies in different similarity measures. When the co-reading relationship is based on the number of co-read books, users who have read thousands of books can have many "co-readers". However, once the co-reading relationship is computed by cosine similarity, these users cannot stand out because their total number of books is used as a normalization factor in the denominator.

5.4.1.3 Correlation between centralization measures

For all three networks, we find that all centralization measures do not follow normal distribution, so Spearman's correlation coefficient is used.

Table 5.3 Correlation between centralizations for familiarity network F

	indegree	outdegree	incloseness	outcloseness	betweenness
Indegree CC	1.000	.286**	.462**	.281**	.324**
Sig. (2-tailed)	.	.000	.000	.000	.000
N	974	974	974	974	974
Outdegree CC	.286**	1.000	.299**	.999**	.923**
Sig. (2-tailed)	.000	.	.000	.000	.000
N	974	974	974	974	974
Incloseness CC	.462**	.299**	1.000	.298**	.283**
Sig. (2-tailed)	.000	.000	.	.000	.000
N	974	974	974	974	974
Outcloseness CC	.281**	.999**	.298**	1.000	.916**
Sig. (2-tailed)	.000	.000	.000	.	.000
N	974	974	974	974	974

CC: Correlation Coefficient

As shown in

Table 5.3, For the familiarity network F, outdegree, outcloseness and betweenness have the strongest correlation. The correlation is significant at the 0.01 level (two-tailed). The correlation lies between outdegree and outcloseness is as strong as 0.999. This indicates that users who follow many others, are also at the “center” of the network made of out links. The correlation between outdegree and betweenness is 0.923. This indicates that users who follow many others, are also the ones who are on the shortest paths between many user pairs. The correlation between outcloseness and betweenness is 0.916. This indicates that users who are at the “center” of the network made of out links, are also on other users’ shortest paths.

Table 5.4 Correlation between centralities for similarity network $C_{cooccur}$

			degree	closeness	betweenness
Spearman's rho	degree	Correlation Coefficient	1.000	**.997****	.662**
		Sig. (2-tailed)	.	.000	.000
		N	974	974	974
	closeness	Correlation Coefficient	.997**	1.000	.647**
		Sig. (2-tailed)	.000	.	.000
		N	974	974	974
	betweennes s	Correlation Coefficient	.662**	.647**	1.000
		Sig. (2-tailed)	.000	.000	.
		N	974	974	974

**. Correlation is significant at the 0.01 level (2-tailed).

For the similarity network based on co-occurrence $C_{cooccur}$, all three centralities measures have significant correlations (at the 0.01 level). The degree and closeness have the strongest correlation (0.997). It means that users, who have many "interest sharers" (if they have read more than 52 common books), are also at the center of this network. It is easy for them to reach other users with different reading interests (users who do not read common books).

Table 5.5 Correlation between centralities for similarity network C_{sim}

			degree	closeness	betweenness
Spearman's rho	degree	Correlation Coefficient	1.000	.988**	.764**
		Sig. (2-tailed)	.	.000	.000
		N	974	974	974
	closeness	Correlation Coefficient	.988**	1.000	.718**
		Sig. (2-tailed)	.000	.	.000
		N	974	974	974
	betweenness	Correlation Coefficient	.764**	.718**	1.000
		Sig. (2-tailed)	.000	.000	.
		N	974	974	974

**. Correlation is significant at the 0.01 level (2-tailed).

For the similarity network based on cosine similarity C_{sim}, all three centralities measures have significant correlations. These correlations are stronger than $C_{cooccur}$. The strongest correlation is still between degree and closeness (0.988). It means that users who have many "interest sharers" (based on if they have a similar reading pattern), are also at the center of this network. It is easy for them to reach other users with different reading patterns.

5.4.2 Microscopic Network structure

Figure 5.1 shows the network structure of social network F where node size indicates indegree. In this network, the user with the largest indegree is the seed user. It is interesting to see that users with large indegree and those with large outdegree seldom overlap. The users who have large indegree but small outdegree, may be users with high attractiveness. For example, one user has a large indegree in the studied network, and his or her outdegree is not in the top-20. Through further examining this user network metrics, one can see that

this user is followed by thousands of people. On the other hand, users who have large outdegree but small indegree may be those who hang out a lot on Douban.

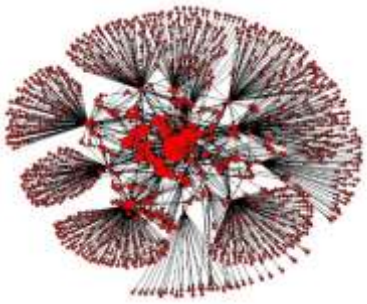


Figure 5.1 Familarity network F (node size indicates indegree)

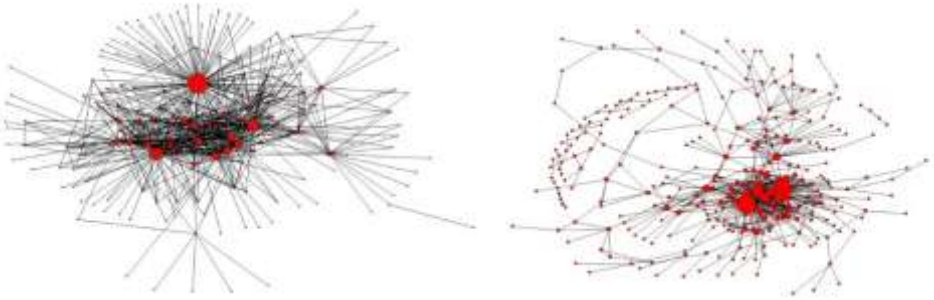


Figure 5.2 Similarity network based on Co-occurrence (left) and cosine similarity (right)

Figure 5.2 shows that the two similarity networks have very different structures. All the nodes in $C_{cooccur}$ are connected, while there are isolated groups in C_{sim}. As is mentioned in the "centralization" section, this is because that the cosine similarity measure also considers the number of books a user has read. In C_{sim}, the upper right 4-node-group consists of users with similar reading interests. Two of the users show a high concentration in Chinese traditional literatures and architecture. Furthermore, one user's homepage has no followee lists but only book collections. It indicates that users who share reading

interests are not yet friends. This suggests a possible future direction, which is to recommend friends to users based on similar reading patterns.

Different from social network$F$, for both co-reading networks, the top uses in three centrality measures are quite consistent. There is only one exception. For cosine-similarity based co-reading network C_{sim}, users with large betweenness do not have large degree or closeness. For example, one user has read only eight books. Two books discuss the topic of social network and complex networks, while the remaining are all Harry Porter series. Since Harry Porter series are quite popular and have thousands of readers in Douban, this user becomes the bridge node on the shortest paths of many users who have read Harry Porter series.

5.4.3 Local Network Structure: Blockmodeling

5.4.3.1 Overall blockmodel

According to the discussion in the method section, structural equivalence is first applied on the networks. The numbers of blocks are set to two, three and four to find the smallest number of errors. In the image matrices in Table 5.6 and Table 5.7, "-"denotes 0 block, "reg" denotes regular block where each row and column contains at least one "1", "com" denotes complete block.

For $C_{cooccur}$, first, structural equivalence block model with random initial assignments is applied. The 2-block, 3-block, and 4-block model have 655, 585 and 574 errors. Thus, regular equivalence is applied, which has much better result as is shown below. The 2-block, 3-block, and 4-block model have 655, 243 and 243 errors. We only show the performances of the 3-block (Table 5.6) and 4-block (Table 5.7) models since they outperform their 2-block counterpart by far few errors.

Table 5.6 Image Matrix and Error Matrix of RE with 3 blocks

Image Matrix				Error Matrix			
	1	2	3		1	2	3
1	-	reg	-	1	20	0	10
2	-	reg	-	2	58	75	73
3	-	-	-	3	0	0	7

Table 5.7 Image Matrix and Error Matrix of RE with 4 blocks

Image Matrix					Error Matrix				
	1	2	3	4		1	2	3	4
1	-	-	reg	-	1	21	2	0	7
2	-	-	-	-	2	0	0	0	0
3	-	-	reg	-	3	53	13	79	61
4	-	-	-	-	4	0	0	0	7

For social network F, by comparing the two images in Figure 5.3, we can see that the only difference between the 3-block model and the 4-block model lies in the newly added cluster which consists of only two users. Thus, the 3-block model is easier to interpret. The blue block consists of nodes with a small betweenness (close to zero), while the green block contains users with larger betweenness.

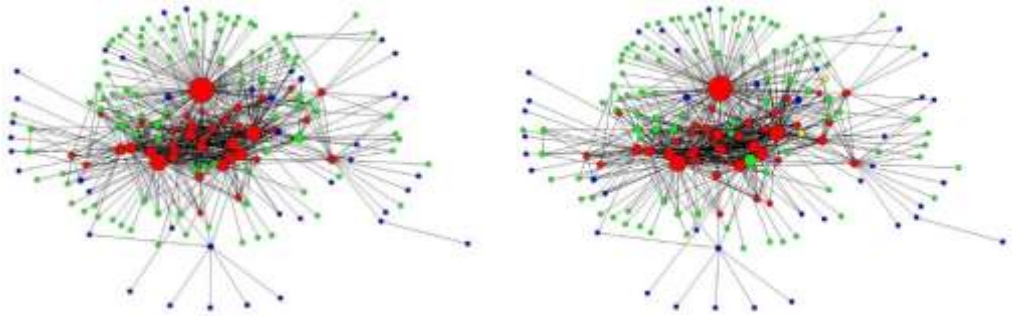


Figure 5.3 3-block and 4-block model for F (color denotes the partitions by RE)

For C_{sim}, since it has more than 256 nodes, block modeling is done by UCINET instead of Pajek. The 3-block Regular Equivalence model is very interesting. Two blocks contain only one user. One contains user 676, the other contains user 10. However, no obvious differences are found between these two users from others on any attributes.

5.4.3.2 Location and Location-based Blockmodeling

In this section, we use chi-square test[19] to find out if a user's location is correlated with her partitions in the block model. Result shows that continent information is too broad to partition the users into separate clusters since most douban users come from China. Thus, users are coded by their city areas instead of continents. For users outside China, we code them as "Overseas". Since Beijing and Shanghai have the most users, they are coded separately. Users without location information are coded as "Null". Red, Green, Yellow, Pink and Blue represent users from Beijing, Shanghai, China-other city, overseas, and users without location information.

[19] http://en.wikipedia.org/wiki/Chi-squared_test

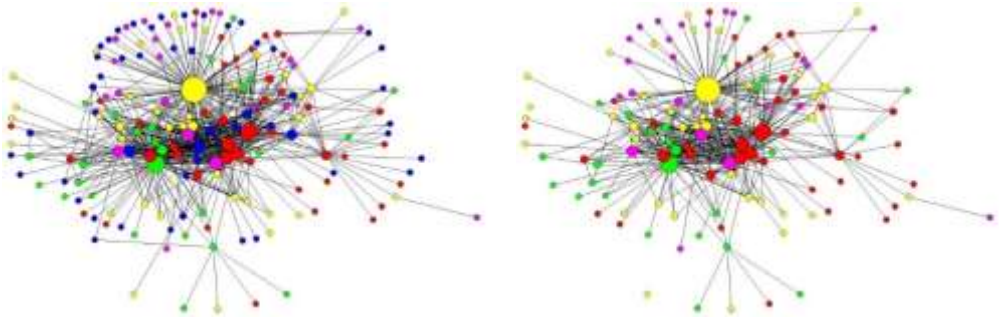


Figure 5.4 Similarity network based on Co-occurrence (left) and cosine similarity (right)(color denotes users' city_area, size denotes degree)

Table 5.8 Color coding of user locations

Location	Color
Beijing	
Shanghai	
China-other city	
Overseas	
No location information	

From Figure 5.4, we can see that after coding locations into major city areas (Table 5.8), each user's degree and location does not have clear relation. The user with the largest degree comes from neither Beijing nor Shanghai. However, except for this user, users from Beijing still have a larger degree in general.

We then run regular Equivalence blockmodeing again by using location as the initial partition. However, for both structural equivalence and regular equivalence models, the final error number is 684, much larger than that of the blockmodels with random initial partitions. Thus, it can be concluded that there is no correlation between users' location information and their positions in the co-reading network.

5.4.4 Test the multiplexity of relations

To see if the social network F and co-reading networks C overlap, the cosine similarities between each user's vectors in two networks are computed. We compute this cosine similarity for a user set of 39 users in total. Results show that these cosine similarities are quite small. They are within the range between [-0.018, 0.044]. More than half users have this value close to zero. Thus, we can conclude that for the small user set examined, there is no correlation between their follower/followee relationship and the co-reading relations. In other words, users' followees and their interest sharers do not overlap.

5.5 Conclusion and Future work

In this chapter, we make a comparison study on the follower-followee network and the co-reading network of an online recommendation system. We are motivated by the assumption that if two networks have similar properties, social network may not provide additional information to user similarities. Hence, a user-based Collaborative Filtering algorithm would be sufficient for recommendation. We compared the structures of the "familiarity-based" and "similarity-based" networks in the same system from macroscopic and microscopic perspectives. We also used blockmodeling to find out if a user's location is correlated with his or her network partitions. In addition, we computed the cosine similarities between the vectors of 39 overlapping users in two networks. Results show that these two types of networks were different. Moreover, users' followers and their interest sharers did not overlap. We concluded that social networks could provide additional information to user similarities based on preferences. It would be a helpful data source for recommendation tasks. In this regard, this chapter prepare us for developing social network-based recommendation algorithms. This topic will be discussed in the next

Chapter.

We also recognize several limitations of this study. For example, the social network F and the book collections were crawled at different time points. At the time of crawling collections, users' followee list could have changed. A more rigorous dataset need to be obtained, where the social network and book collections are crawled at the same time points.

Future studies can focus on the following issues. First, other similarity measures, such as Jaccard similarity, can be used to construct the co-reading network. Besides, ratings can be used to build user vectors. Furthermore, this study can be extended to other domains such as music and movie. One can construct the "co-listening" and "co-watching" networks. These networks can then be compared with the Follower-Followee network to see if different item types lead to different results. Last but not the least, since we find out that similarity-based network is different from social network, it is possible to recommend interest sharers to users as their friends.

Chapter 6. Recommendation with user network

In last chapter, we compared a social network and interest-sharer network in a recommender system, Douban. We find that the cosine similarities between the same user's vectors in the two types of networks were very small, indicating that users' followers and their interest sharers did not overlap. This leads to the conclusion that social networks could be a valuable data source for recommendation tasks. Based on this finding, in this chapter, we will develop recommendation algorithms with user networks. This can also alleviate the user-side cold start problem.

6.1 Introduction

Recommendation has become a buzzword nowadays. Although recommendation systems can help people find their favorite products automatically, "they cannot completely substitute personalized recommendations that people receive from friends" [70]. A survey in 2003 found out that 68% of individuals consulted friends and relatives before purchasing home electronics [19].

Previous studies have incorporated social networks into recommender algorithms in two approaches. In the first approach, the social network is fused with the rating matrix. Both the user social network and the rating matrix are decomposed by the Probabilistic Matrix Factorization (PMF) model by sharing the same latent user factors [77]. In the second approach, only the rating matrix is decomposed by the PMF model. The social network is used to define regularization terms of the objective function of the PMF model [79]. In this chapter, we will apply these two models to our douban dataset. We will denote them as Social-Fusion and Social-Regularization respectively.

In [77], only one type of social network is incorporated. Their goal is to examine if social relations can help improve recommendation performance. In this chapter, we first construct two types of user networks, a social network and a similarity network based on users' profiles. We then apply the Social-Fusion and Social-Regularization models on these two types of networks. Similar to [79], our first goal is also to compare the performances of two models when the same user network in incorporated. However, our main goal is to find out which type network can help improve the recommendation performance more under the same recommendation algorithm. As shown in Table 6.1, we compare the results between I and II, III and IV, to see which social-based recommendation algorithm performs better. We also compare them with the baselines. In addition, we compare the results between I and III, II and IV, to see which user network the performances of the same algorithm by incorporating different user networks rand this will be our primary focus.

Table 6.1 General Framework of Chapter 6

	Social-Fusion	Social-Regularization
Social network	I	II
Similarity network	III	IV

6.1.1 Problem definition

In this chapter, we use book recommendation as an example. In a recommender system, the input are a user-rating matrix R, containing users' ratings on books they have read, and a user network C. The task of rating prediction task, we need to predict the rating that users will give to a book they have not yet read. The task of item recommendation

moves one step further. After all the books get their predicted ratings, they are ranked by the predictions and a ranked list is returned as recommendation.

6.2 Models

In order to incorporate networks into recommendation, we first construct two types of user networks. A Social network F is constructed based on follower-followee relationships. A user similarity network C is constructed based on users' demographic profiles. F_{ij} is one if user i follows user j, and zero otherwise. C_{ij} equals to the cosine similarity of two users' feature vectors. In our experiment, a user's features can include her gender, age, job and location. Note that the follower-followee network is directed, while the similarity network is undirected. Then, both types of networks are incorporated into the two recommendation models originally proposed in [77] [79]. They are presented in the following subsections.

6.2.1 Social-Fusion Model

The key idea behind the Social-Fusion model (Figure 6.1) is that both the rating matrix and the user network are factorized through PMF model [99]. The two decompositions are connected by sharing the same user latent space. The user-item rating matrix R is factored into $U^T V$. The social network F (or similarity network C) is factored into $U^T Z$. U is the common latent user feature matrix, V is the latent item feature matrix, and Z is the factor matrix of the user network.

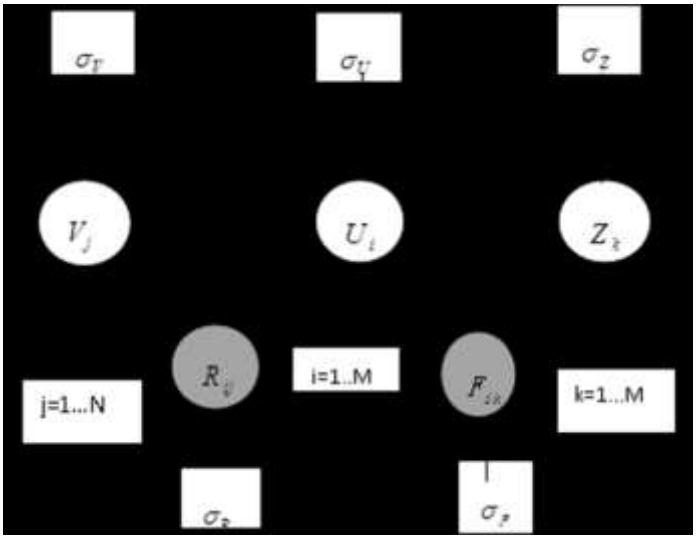


Figure 6.1 Graphical Model for Social-Fusion Model [77]

Take the factorization of the social network F for example. They gave the probability model as below.

$$P(F|U,Z,\sigma_F^2) = \prod_{i=1}^{m}\prod_{k=1}^{m}\mathcal{N}[(f_{ik}|g(U_i^T Z_k),\sigma_F^2)]^{I_{ik}^F} \quad (6.1)$$

Each column of user feature matrix,U_i, represents each user's latent feature vector. Each columns of Z, Z_k, represents factor-specific latent feature vectors. $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2. $g(x)$ is the logistic function, which transforms $U_i^T Z_k$ into (0, 1). I_{ik}^F is 1 if user i has a social relationship with user k and 0 otherwise. In addition, zero-mean spherical Gaussian priors are put on user and factor features vectors as shown in the equations below.

$$P(U|\sigma_U^2) = \prod_{i=1}^{m}\mathcal{N}[(U_i|0,\sigma_U^2 I)] \quad (6.2)$$

$$P(Z|\sigma_Z^2) = \prod_{k=1}^{m}\mathcal{N}[(Z_k|0,\sigma_Z^2 I)] \quad (6.3)$$

Finally, put all these three equations together, the probabilistic model of the social network F is given below. U and Z can be computed using Bayesian rule.

$$P(U,Z|F,\sigma_F^2,\sigma_U^2,\sigma_Z^2) \propto P(F|U,Z,\sigma_F^2)P(U|\sigma_U^2)P(Z|\sigma_Z^2) =$$

$$\prod_{i=1}^{m}\prod_{k=1}^{m}\mathcal{N}[(f_{ik}|g(U_i^T Z_k),\sigma_F^2)]^{I_{ik}^C}\times\prod_{i=1}^{m}\mathcal{N}[(U_i|0,\sigma_U^2 I)]\times\prod_{k=1}^{m}\mathcal{N}[(Z_k|0,\sigma_Z^2 I)] \quad (6.4)$$

In [70], they normalize the weight of f_{ik}. Their assumption is that the social relationship from $u_i$to $u_k$is weaker if u_i has out-links to many users, and is stronger if $u_k$has in-links from many users. The normalization function is given as:

$$f_{ik}^* = \sqrt{\frac{indegree(k)}{outdegree(i)+indegree(k)}}\times f_{ik} \quad (6.5)$$

It is not difficult to see that such normalization is appropriate for directed networks. Hence, In our experiment, besides this normalization function, we use another approach. If F is a directed network, we dividing each f_{ik} by the outdegree of the ith node. If F is an undirected network, we dividing each f_{ik} by the degree of the ith node. This is in fact similar to the transition matrix used in PageRank. f_{ik}^* is a row normalized version of the original social matrix, so that each row of F sum up to one.

$$f_{ik}^* = \frac{f_{ik}}{outdegree(i)} \quad (6.6)$$

Similarity, the probabilistic model of the user-item matrix R factorization is as follows.

$$P(U,V|R,\sigma_R^2,\sigma_U^2,\sigma_V^2)\propto P(R|U,Z,\sigma_R^2)P(U|\sigma_U^2)P(V|\sigma_V^2)=$$

$$\prod_{i=1}^{m}\prod_{j=1}^{n}\mathcal{N}\left[\left(r_{ij}|g\left(U_i^T V_j\right),\sigma_R^2\right)\right]^{I_{ij}^R}\times\prod_{i=1}^{m}\mathcal{N}[(U_i|0,\sigma_U^2 I)]\times\prod_{j=1}^{n}\mathcal{N}\left[\left(V_j|0,\sigma_V^2 I\right)\right] \quad (6.7)$$

Put the probabilistic model of social network and rating matrix together, the final objective function of their social-fusion model is:

$$\min_{U,V,Z}\mathcal{L}(R,F,U,V,Z)=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^R\left(R_{ij}-g\left(U_i^T V_j\right)\right)^2+\frac{\lambda_F}{2}\sum_{i=1}^{m}\sum_{k=1}^{m}\left(F_{ik}^*-\right.$$

$$\left.g(U_i^T Z_k)\right)^2+\frac{\lambda_U}{2}\|U\|_F^2+\frac{\lambda_V}{2}\|V\|_F^2+\frac{\lambda_Z}{2}\|Z\|_F^2 \quad (6.8)$$

Where we have$\lambda_F=\sigma_R^2/\sigma_C^2$,$\lambda_U=\sigma_R^2/\sigma_U^2$,$\lambda_V=\sigma_R^2/\sigma_V^2$, $\lambda_Z=\sigma_R^2/\sigma_Z^2$ $\|\cdot\|_F^2$ represents

the Frobenius norm. Hence, we can do gradient descent to find the optimal parameters U, V and Z which give a local minimum of this objective function as shown below.

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j=1}^{n} I_{ij}^{R}\, g'\left(U_i^T V_j\right)\left(g\left(U_i^T V_j\right) - r_{ij}\right)V_j + \lambda_F \sum_{k=1}^{m} I_{ik}^{F} g'(U_i^T Z_k)(g(U_i^T Z_k) - f_{ik}^{*})Z_k + \lambda_U U_i \quad (6.9)$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^{R}\, g'\left(U_i^T V_j\right)\left(g\left(U_i^T V_j\right) - r_{ij}\right)U_i + \lambda_V V_j \quad (6.10)$$

$$\frac{\partial \mathcal{L}}{\partial Z_k} = \lambda_C \sum_{i=1}^{m} I_{ik}^{F} g'(U_i^T Z_k)(g(U_i^T Z_k) - f_{ik}^{*})U_i + \lambda_z Z_k \quad (6.11)$$

In our approach, besides using the social network$F$, we also try to construct a user similarity network and use that as additional information. We still use Ma et al's social fusion model [77], but replacing F with the user similarity network$C$.

6.2.2 Social-regularization model

The basic matrix factorization method puts no constraint on either the latent user matrix or the item matrix. Intuitively, this is not correct. For example, if two movies are directed by the same director, their feature vectors should be more similar than movies directed by different directors. Similarity, if two users live in the same location, their preference vectors could be more similar than users who live in different locations. Besides, a user's preference vector should be similar to her friends' preference vectors. This suggest us to add constraints onto the user and item feature vectors. Constraints can be added by incorporating item metadata, user profile or user social networks. In [79], they included user social network and proposed four models by varying the way that constraints are added. They divided them into two models, "Average-based Regularization" and "Individual-based Regularization". The objective functions of these two models are given below. Similar with the social fusion model, in our approach, besides using the social network$F$,

we also use the constructed user similarity network C as additional information. We still use [79] social regularization model, but replacing F with the user similarity network C.

6.2.2.1 Average-based Regularization

In the average-based regularization model, the assumption is that a user's preferences (represented in her feature vector) should be close to the average of her friends' preferences. The objective function is shown in equation (6.12):

$$\min_{U,V} \mathcal{L}_1(R,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\alpha}{2}\sum_{i=1}^{m}\left\|U_i - \frac{1}{|\mathcal{F}^+(i)|}\sum_{f\in\mathcal{F}^+(i)} U_f\right\|_F^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 \quad (6.12)$$

where $\mathcal{F}^+(i)$ denotes u_i's out-link friends. For instance, in Douban, they are u_i's followees. However, the authors pointed out that different friends should have different levels of impacts on one's preference. Such impact strength can be reflected on the similarity between the friend and the user. As a result, the objective function is modified by adding the user similarities which is presented in equation (6.13). The assumption is that, a user's preferences should be close to the weighted average of her friends' preferences. The more similar a friend is with the active user, the more impact her preferences have on the active user's preferences.

$$\min_{U,V} \mathcal{L}_1(R,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\alpha}{2}\sum_{i=1}^{m}\left\|U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i.f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i.f)}\right\|_F^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 \quad (6.13)$$

The local minimum of this objective function is found by gradient descent on user and item feature vectors. The gradients of the user and item features are:

$$\frac{\partial \mathcal{L}_1}{\partial U_1} = \sum_{j=1}^{n} I_{ij}\left(U_i^T V_j - R_{ij}\right)V_j + \lambda_1 U_i + \alpha\left(U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i.f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i.f)}\right) +$$

$$\alpha \sum_{g \in \mathcal{F}^{-}(i)} \frac{-Sim(i.g)\left(U_g - \frac{\sum_{f \in \mathcal{F}^{+}(g)} Sim(g.f) \times U_f}{\sum_{f \in \mathcal{F}^{+}(g)} Sim(g.f)}\right)}{\sum_{f \in \mathcal{F}^{+}(g)} Sim(g.f)} \tag{6.14}$$

The similarity between two users is calculated by the Vector Space Similarity [17] of their rating vectors.

In our experiment, due to the data sparseness, we apply the model in the first equation (6.15). We first apply this model by using social network F. Besides, for MovieLens dataset, we replace the social network by a user similarity network. The assumption here is that a user's preference should be close to the preferences of users with similar demographics. Since the user similarity matrix is a symmetric matrix, we use $S(i)$ to denote user i's similar users.

$$\min_{U,V} \mathcal{L}_1(R, U, V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\alpha}{2}\sum_{i=1}^{m}\left\|U_i - \frac{1}{|S(i)|}\sum_{s \in S(i)} U_s\right\|_F^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 \tag{6.15}$$

6.3 Experiments

In this section, we will first present our datasets used in the experiments. We then show the preliminary steps to prepare our datasets.

6.3.1 Dataset

Three datasets are used in our experiments. Among them, one is the MovieLens dataset which is directly downloaded online. The other two are Douban datasets. One is provided by the authors in [79]. We use Douban's API to crawl the site and obtain the other dataset. The detailed descriptions for each dataset are shown below.

(1) MovieLens 100k. This MovieLens dataset contains 100,000 ratings by 943 users on 1682 movies. Each user has rated at least 20 movies. In addition, it contains

demographic information for users and metadata for movies. User demographic information include age, gender, occupation and zip code. Movie metadata includes title, release date, video release data, IMDb URL, and genres.

(2) Ma et al.'s Douban dataset [79]. The authors first crawled all the users in more than 700 groups under "Movie" subcategory in Douban. These users' social networks and movie ratings were further crawled. The dataset consists of two parts. The user-item rating data contains 16,830,839 ratings on 58,541 unique movie items given by 29,490 unique users. The social friend network contains 1,692,952 social relationships. The social network here is a "Facebook-like" network where an edge between two users is one if they find each other via email and add each other friends. The authors claimed that users in these network actually know each other offline. This network is an undirected network.

(3) Our douban book dataset. We obtain our douban dataset through crawling douban via its API[20], the largest recommender system in China. As of April, 2013, douban has more than 68 million registered users[21]. We first crawled users' followees from one seed user[22]. All users within three steps of the seed user were crawled by breadth-first-search. In this way, we obtained a "who-follows-whom" social network which contains 4,779 users and 104,799 edges. We then crawled the book collections of all these users, and obtained a user preference network which contains 4,779 users, 434,005 books, and 2,549,523 edges. To construct the user similarity network, we also crawl users profiles, which include their ids, userIds, names, times when they created their accounts, locationId and locationName.

---

[20] Douban API V2. http://developers.douban.com/wiki/?title=api_v2

[21] http://www.alexa.com/siteinfo/douban.com

[22] We started crawling Douban from May, 2011.

6.3.2 Preliminary step

In this section, we present the preliminary steps to prepare our datasets for the experiment. Since the MovieLens datasets have no social network data, we build models by incorporating a user similarity matrix. The user similarity matrix is constructed by computing the similarities of users' profile vectors. Then, the user similarity matrix is incorporated into the Social-Fusion and Social-Regularization models introduced earlier.

6.3.2.1 MovieLens 100k dataset

For MovieLens 100k dataset, we represent each user as a vector in which gender, age, occupation, and zip code are features. Each unique value of the demographic information is coded as one feature. First, a user's gender is coded into two features, isMale and isFemale. A male user is represented as (1, 0), and a female user is represented as (0, 1). Second, a user's age information is coded into 11 features, which are 11 age ranges including [0, 9], [10, 14], [15, 19], [20, 24], [25, 29], [30, 34], [35, 39], [40, 44], [45, 49], [50, 54], [55, 59], [60, 64] and [65, infinity). Negative ages are removed. A user whose age falls into one group is coded as one on that age feature, and zero for the remaining features. Third, a user's location information is coded into 10 features based on the first digit of their zip code since this digit normally denotes the states. Finally, we manually go through users' occupations and classify them into eight categories. Each category is coded as one feature. Our categorization of users' jobs is shown in Table 6.2.

Table 6.2 Categorization of users' occupations of MovieLens 100k dataset

Category	Occupations
1	Administrator, Executive
2	Artist, Entertainment, Writer

3	Educator, Librarian, Scientist
4	Doctor, Engineer, Healthcare, Lawyer, Programmer, Technician
5	Marketing, Salesman
6	Homemaker, None, Retired
7	Other
8	Student

6.3.2.2 Two douban datasets

For Ma et al's douban dataset [79], in the original dataset provided, there are missing edges between two users. Specifically, for edge from u_i to u_j, some of the reverse edges do not present. This is not correct since the edges in this social network should be reciprocal. Thus, we fill in all these missing edges. We did not do any data preprocessing for our douban dataset.

For both douban datasets, we use three different strategies to normalize the social network, including normalizing it by both followee's indegree and follower's outdegree (equation (4.5)), normalizing it only by the follower's outdegree (equation (4.6)), and no normalization. We will call them "Double-Norm", "Single-Norm" and "No-norm" in the following sections.

6.3.3 Baselines and Metrics

We compare the Social-fusion algorithm, Social-regularization algorithm, with the state-of-the-art algorithm, Probabilistic Matrix Factorization (PMF) [99]. For MovieLens, we use their five-fold cross validation dataset provided with the original dataset to evaluate. For Ma's douban dataset and our douban dataset, we randomly select 80% of the user-item-rating triplets as training data, and use the remaining 20% as test set. To tune the parameters

and prevent overfitting, we further split the training set into a train set, which takes 80% random triplets in training, and a validation set, which takes the remaining 20% triplets.

Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are implemented as our evaluation metrics to measure the prediction performances of the social-based recommendation algorithms and baselines. Specifically, MAE and RMSE are defined as follows. MAE is the average of the absolute errors between predicted ratings and the true ratings. RMSE equals to the square root of the average sum squared error between predictions and true ratings.

$$MAE = \frac{\sum_{i,j} |r_{ij} - \hat{r}_{ij}|}{N} \tag{6.16}$$

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{ij} - \hat{r}_{ij})^2}{N}} \tag{6.17}$$

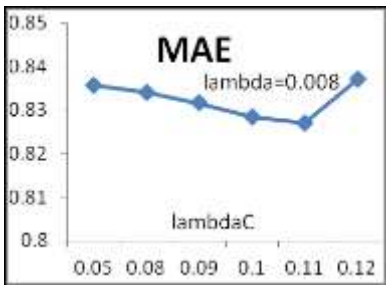
where r_{ij} is the rating that user i gave to item j, $\hat{r}_{ij}$ is the predicted rating that user i would give to item j from the algorithms. N is the total number of ratings in the test set.

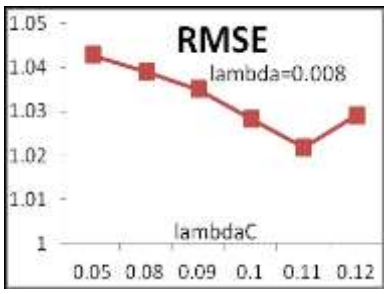
6.3.4 Parameter tuning

6.3.4.1 MovieLens 100k dataset

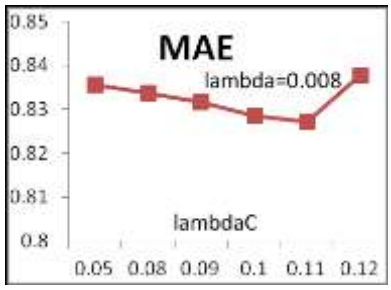
In order to make comparison fair, we first tune the parameters by grid search. Then, the best performance of each algorithm is reported. We first report the parameter tuning for the Social-Fusion model [77], then we report the parameter tuning for the Social-regularization model [79].

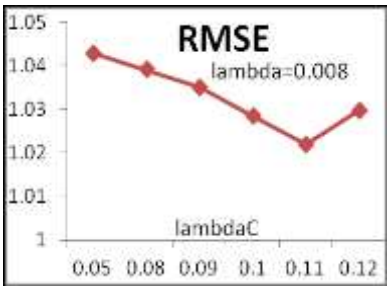
(1) Parameter tuning for the Social-fusion model [77] on MovieLens 100k





(a) Dimensionality = 2

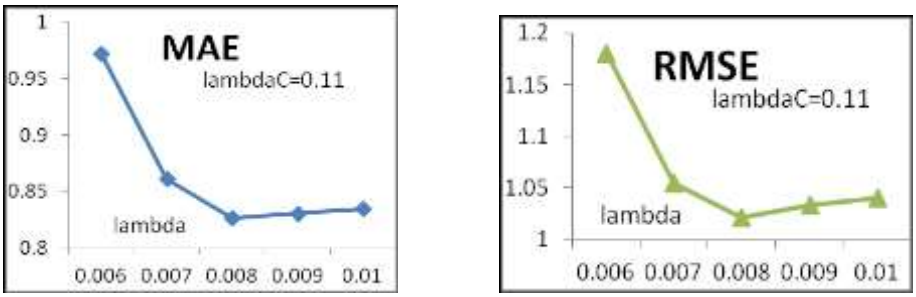




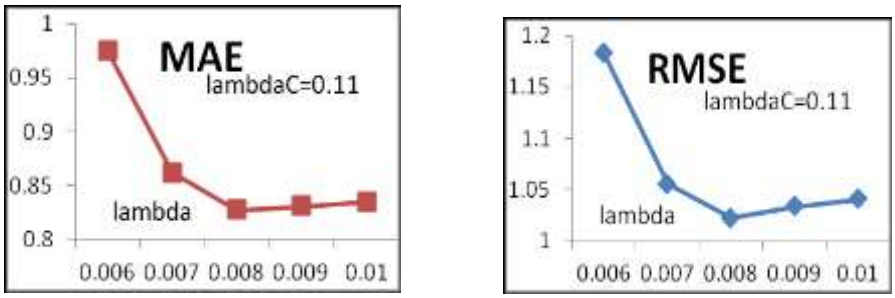
(b) Dimensionality = 3

Figure 6.2 Impact of λ_C (the weight of user similarity matrix)

As is shown in Figure 6.2, λ_C is relatively sensitive. If it is set to infinity, then we only extract the user similarity network to predict users' preferences. If it is set to zero, then we only learn users' preferences from the user-item rating matrix. When we set the regularization parameter λ to 0.008, we found the optimal value for λ_C is 0.11. This number is much smaller than its optimal value in the Epinion dataset [77], which ranges from 10 to 20. This indicates that the parameters are dependent on the dataset. Besides, douban's social network may have less help than the trust network in the Epinion dataset.



(a) Dimensionality = 2



(b) Dimensionality = 3

Figure 6.3 Impact of λ (the regularization parameter) on MovieLens 100k

We also tune the regularization parameter lambda, by fixing the weight λ_C at its optimal value. Figure 6.3 shows that the optimal regularization value equals 0.008, which is also different from its optimal value in Epinion dataset [77], which equals 0.001. This again indicates that the parameters are dependent on the dataset.

(2) Parameter tuning for the Social-regularization model

Before tuning the parameters, we first tune the learning rate alpha by examining the RMSE on the training data only. Result showed that the RMSE steadily decreases when alpha is within the range of [0.004, 0.012]. The RMSE reaches smallest when alpha equals 0.009 (or 0.01).

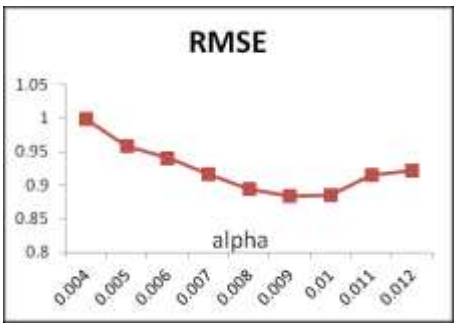


Figure 6.4 Learning rate for Social-regularized model

We then tune the weight of social-regularization component c, by setting the learning rate to 0.009. A small c means only use the rating matrix to learn user preference. A large c means a user's similar users have large impact on his preference.

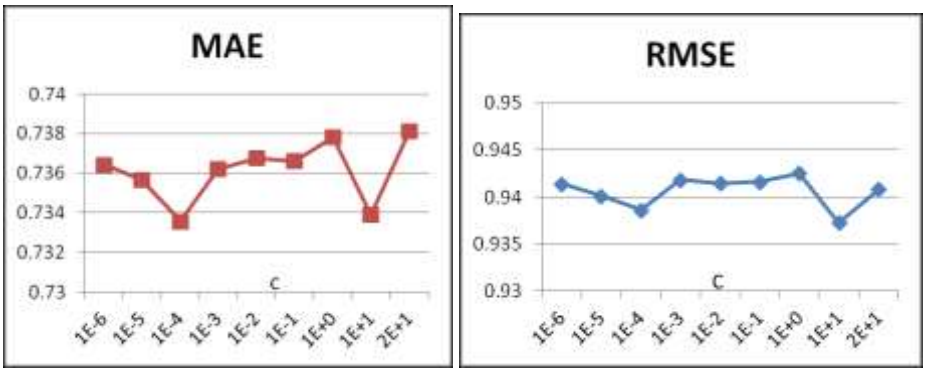


Figure 6.5 Impact of c(the weight of social regularization)

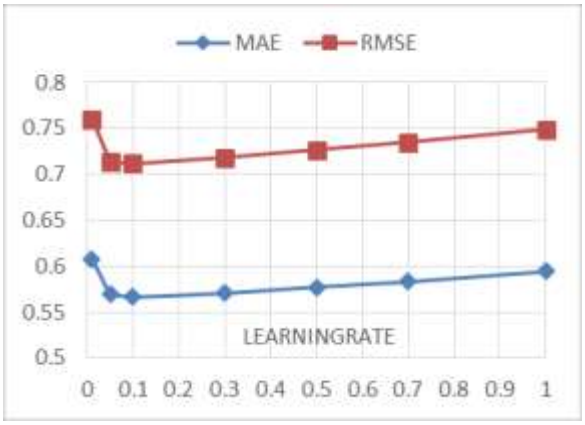
The impact of c is interesting. From Figure 6.5, we can see that there exist two optimal values. One is e10-4, the other is 1. This result is different from [79], and is difficult to understand from intuition. In their study on Douban and Epinion datasets, the same parameter has a single optimal value. Their finding is consistent with the intuition that integrating both rating matrix and social network can generate better recommendation

performance than using only one of them. We assume that such difference may be due to the characteristics of the datasets. We will find out if the "two-optimal-value" phenomenon also appears in other MovieLens datasets.

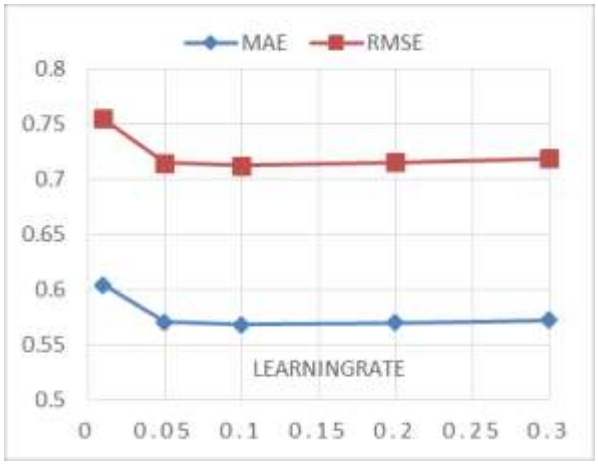
6.3.4.2 Ma's douban dataset

(1) Parameter tuning for the Social-fusion model

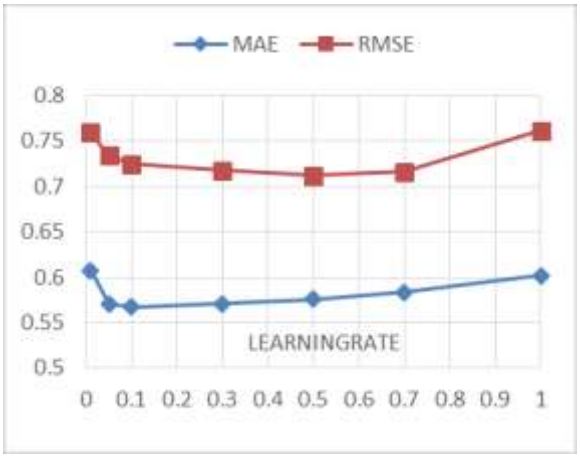
We use the same parameter settings reported in [77] for the Social-fusion model on Ma's douban dataset. The parameters are $\lambda = 0.001$ and$\lambda_C = 10$. We set dimensionality = 5 and tune the learning rate given these settings. We show the impact of learning rate under three strategies of social network normalization, "Double-Norm", "Single-Norm" and "No-norm".



(a) No-Norm (Normalize by follower outdegree)



(b) Double-Norm (Normalize by both followee indegree and follower outdegree)



(c) Single-Norm (Normalize by follower outdegree)

Figure 6.6 Impact of learning rate for different Social network normalizations

From Figure 6.6, we can observe that under both No-norm and Double-Norm strategies, the optimal learning rate is 0.1. However, under our Single-Norm strategy, there is no optimal learning rate which can generate both smallest MAE and RMSE. When learning rate equals to 0.1, we reach the best the MAE; when learning rate equals to 0.5, we reach the best the RMSE. We believe this is because when we normalize the social

network via dividing the edge weights by the follower's outdegree, the normalized edge weights become much smaller. Thus, a larger learning rate can work well without overfitting.

(2) Parameter tuning for PMF

We also tune parameters for Probabilistic Matrix Factorization (PMF) model. We set dimensionality $k = 5$, and tune the learning rate α and regularization λ. The optimal parameter settings for this dataset are: $k = 5$,$\lambda = 0.03$,$\alpha = 0.01$. The number of iterations is set to 50.

6.3.4.3 Our douban book dataset

(1) Parameter tuning for the Social-fusion model

We first tune the learning rate by setting dimensionality $k = 5$, $\lambda = 0.001$, $\lambda_C = 1$. As shown in Figure 6.7, the optimal learning rate for the Social-fusion algorithm under three normalization strategies are the same, all equal to 0.1. We can also see that beyond 0.05, the impact of learning rate on the algorithm performance is small, indicating the robustness of this algorithm.

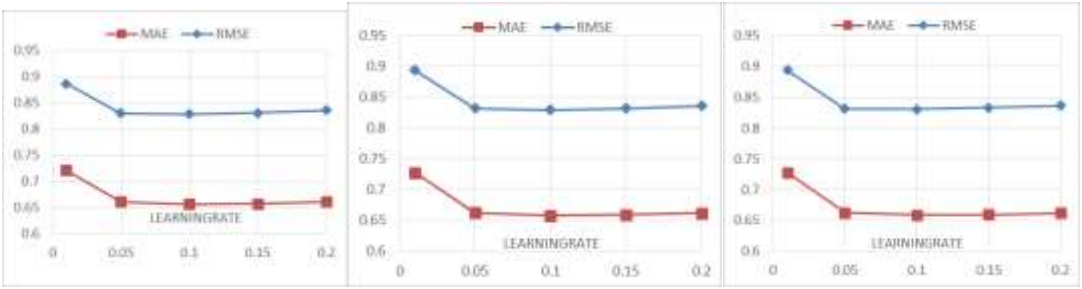


Figure 6.7 Impact of learning rate (left: No-norm SNS, middle: Single-norm, right: Double-norm)

We then tune regularization weight λ by setting dimensionality $k = 5$, $\lambda_C = 1$, $learningRate = 0.1$. As shown in Figure 6.8, the optimal λ for the Social-fusion algorithm under three normalization strategies are the same, all equal to 0.0005. Such a small regularization value tells us that the learned values in latent factors U, V, and Z are in a reasonable range and there is little risk to overfit. Compared to learning rate, the regularization weight has even smaller impact on the algorithm performance, indicating the robustness of this algorithm.

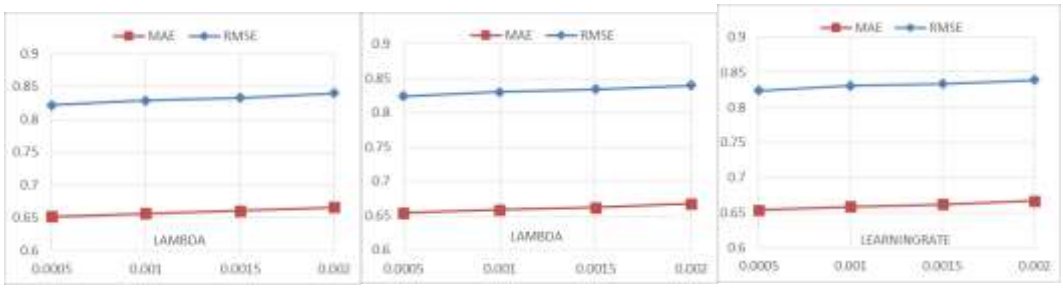


Figure 6.8 Impact of λ (left: No-norm SNS, middle: Single-norm, right: Double-norm)

Finally, we tune λ_C by setting dimensionality $k = 5$, $\lambda = 0.0005$, $learningRate = 0.1$. λ_C is a parameter to balance the information from the social network and from the rating matrix. If $\lambda_C = Infinity$, only social network is used to predict user ratings. If $\lambda_C = 0$, only the rating matrix is used to learn user ratings. For any values between these two extremes, both the social network and rating matrix are used in the matrix factorization to predict user preferences. As shown in Figure 6.9, the optimal λ_C for the Social-fusion algorithm under three normalization strategies are different. They are 1, 0.4 and 0.2 respectively.

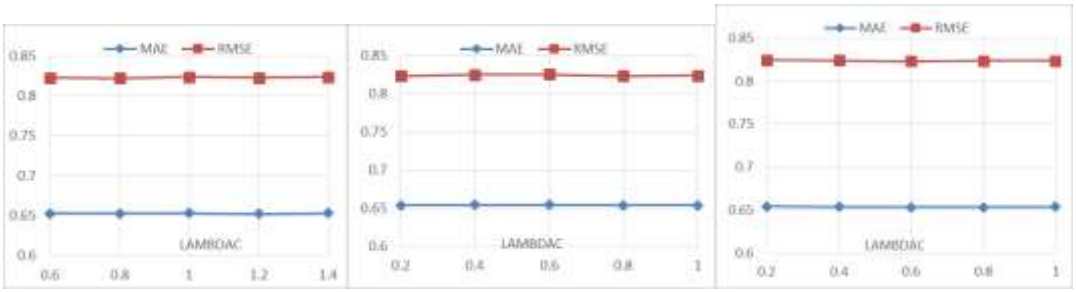


Figure 6.9 Impact of λ_C (left: No-norm SNS, middle: Single-norm, right: Double-norm)

(3) Parameter tuning for PMF

For our douban dataset, we also tune parameters for Probabilistic Matrix Factorization (PMF) model. We set dimensionality $k = 5$, and tune the learning rate α and regularization λ. The optimal parameter settings for this dataset are: $k = 5$, $\lambda = 0.05$, $\alpha = 0.07$. The number of iterations is set to 50.

6.4 Results

In this section, we conducted experiments to compare two variants of social-based recommender, Social-Fusion [77] and Social-Regularization model [79] and the baseline. The experiments are conducted on the aforementioned three datasets.

6.4.1 MovieLens 100k dataset

As shown in Table 6.3, for the movielens 100k dataset, once we add a user similarity network as the social component, we found out that the Social-regularization Model [79] performs best. In particular, it outperforms the Social-fusion model [77]. The reason may be that in Social-fusion model [77], the factorization of both the rating matrix and the social network are constrained by sharing the same user feature matrix. Such constraint makes it

difficult to learning the parameters. Moreover, the Social-fusion model [77] is much sensitive to parameter tunings than the Social-regularization model [79].

Table 6.3 MAE and RMSE for MovieLens 100k (dimension = 2)

Models	MAE	RMSE
PLSA	0.7597	0.9724
PMF (gdpmf)	0.7543	0.9614
Social-fusion	0.8404	1.0459
Social-regularization	0.7336	0.9386

6.4.2 Ma's douban dataset

We also compare the performance of Social-fusion model [77] under different social network normalization strategies. As can be seen from Table 6.4, it first seems surprisingly that this model performs best when no normalization is placed on the edge weights of the social network. However, after examining the social network of this dataset, it can be seen that this is a reasonable result. In this dataset, all the edges in the social network are reciprocal, which means that if there is an edge from u_i to u_k, then there is an edge from u_k to u_i. However, the Double-norm only applies for directional networks since it differentiates between a user's indegree and outdegree. Hence, originally, if there is an edge between u_i and u_k, then we have $c_{ik} = c_{ki} = 1$. After this normalization,

$$f_{ik}^* = \sqrt{\frac{indegree(k)}{outdegree(i)+indegree(k)}} \times f_{ik}$$

we have $f_{ik}^* \neq f_{jk}^*$. Although the single-norm strategy also destroys the symmetry of the network and have $f_{ik}^* \neq f_{jk}^*$, the normalized weights becomes much smaller and have little effect on the final performance. Hence, in the following section, we will only use the

performance of Social-fusion model without normalizing the social network, with other models.

Table 6.4 Impact of Social network normalizations

	MAE	**RMSE**
No-norm	0.5641	0.7080
Single-norm	0.5645	0.7089
Double-norm	0.5672	0.7109

As one can see from Table 6.5, different from what is reported in [77], the social-fusion model performs worse than PMF. We assume the reason could be that we have a training and test different from theirs.

Table 6.5 Comparsion of performance on Ma et al.'s douban dataset

Models	**MAE**	**RMSE**
PMF (gdpmf)	0.5578	0.7047
Social-fusion	0.5641	0.7080

6.4.3 Our douban book dataset

For our douban book dataset, we also compare the performance of Social-fusion under different social network normalizations. As can be seen from Table 6.6, Social-fusion with no normalization still performs best. However, for our douban dataset, Double-norm and Single-norm have almost identical performance. This is different from our initiation. In this dataset, edges in the social network are directional. A user's indegree and outdegree can be different. Since the Double-norm considers both indegree and outdegree of a node,

it should generate better result than Single-norm which only consider a node's outdegree. We will investigate further on reason why they perform the same.

Table 6.6 Impact of Social network normalizations

	MAE	RMSE
non-norm	0.6521	0.8222
Single-norm	0.6534	0.8235
Double-norm	0.6535	0.8232

As one can see from Table 6.7, for our douban dataset, the social-fusion model still performs worse than PMF.

Table 6.7 MAE and RMSE comparison

Models	**MAE**	**RMSE**
PMF (gdpmf)	0.6423	0.8164
Social-fusion	0.6521	0.8222

6.5 Conclusion and Future work

In this chapter, we focus on the social recommendation problem. We apply two existing social recommendation algorithms onto three datasets, one from Movielens, the other two from Douban. In our experiment, for the Movielens dataset, since it does not have real social network, we construct a user similarity network based on user profiles as a surrogate for the social network. For the other two douban datasets, the social network is either made of reciprocal friendship links, or directed follower-followee links. Our intuition is that both social network and user similarity network can help improve recommendation

accuracy. For the social network, this is based on the assumption that users tend to share their friends' preferences. For the similarity network, this is based on the assumption that users tend to share preferences with others having similar demographics. Besides, social recommendation can also solve the user-side cold-start problem.

However, experimental results are contradictory to our expectations. We find out that social recommenders do not always outperform the matrix factorization baseline. For the Movielens dataset, results show that the social regularization algorithm [79] outperforms PMF, while Social-fusion model [77] does not. For the other two douban datasets, results show that Social-fusion model [77] still does not outperform PMF. In addition, results from different normalization strategies on user network show that, normalizing the user network does not help the prediction accuracy, instead, it has a negative effect.

In this chapter, we use all the connections of each user to help improve the recommendation. In reality, people may consult different social connections on different topics. Take book recommendation for example. A user may consult some friends on historic books, while affected by other friends' preference on science fictions. Hence, in the future, we will develop a topic-based social recommendation. Given a user and a topic, this algorithm would first identify the group of social connectors who have largest impact on this topic and then recommend items based on their preferences. Since we have already obtained the tags for douban items, it is not difficult to find out the main topics which connect two users in a user network.

Chapter 7. Social Influence in Recommendation Networks

In this chapter, we analyze the social influence in an online recommendation system. In the last chapter, we present the performance of two user network-based recommendation algorithms on several datasets. The problem with these models is that they are predefined based on heuristics. The heuristic underlying the Social Fusion model [77] is that user (social) network and rating matrix should share the same user latent factors. Based on this heuristic, user network and the rating matrix were simultaneously decomposed. The heuristic underlying the Social regularization model [79] is that a user's preference should be close to her friends or similar users' preferences. Based on this heuristic, user network was employed as a regularization term to the objective function of the matrix factorization. Although these heuristics were reasonable and the predefined models outperformed the baselines, it was unclear why these models worked. We believe that only after understanding the contexts in which social networks work in recommender systems, we can develop more appropriate social recommendation algorithms. The objective of this chapter is to study social influence in recommender systems.

Our dataset consists of a user-book "read" network and a user-follower network. We first investigate in general, if one's recommendations have impacts on her followers' decisions. We then analyze the correlation between one's influence and her network centrality. Finally, we investigate how recommendation acceptance rate changes with the number of recommendations. Such investigation is taken from both senders' and receivers' perspectives. Results show that a user does have influence over her followers' decisions, but such influence is not correlated with her centrality. From the receiver's perspective, the

more recommendations one receives for a book, the more likely that she will accept it. However, there is a saturate point beyond which additional recommendations will have no more impact. From the sender's perspective, the more recommendations one sends out, the more likely that her recommendations will be accepted. This trend has no saturate point. This chapter is based on our paper published in the first workshop on User Engagement Optimization at CIKM 2013.

7.1 Introduction

Social influence denotes that "one person performing an action causes people connected to her to do the same [22]. There are two lines of work in social influence studies. The first line of work applied social influence in various applications such as recommender systems [5] [91] [104] [112] [113], viral marketing [33] [63] [94], information diffusion [8] [26] [83] [94] [119] [122] [126], and etc. For instance, viral marketing gains its success in traditional marketing [8] [41] [62]. People can take better advantage of recommendation after they understand the contexts in which viral marketing works and the characteristics of products [116]. We can also use viral marketing to market niche products since studies have shown that these products contribute to overall sales significantly [18].

The other line of research in social influence is to validate its existence. Initially, most studies on social influence assumed that social influence was the only reason for people to take similar actions. However, according to Anagnostopoulos's study [4], social influence is just one source of social correlation. Before employing social influence into recommendation algorithm, we need to first validate if social influence was the only source of information diffusion.

Albeit the fact that social influence models have been widely applied in

aforementioned areas, most of them focus on a well-defined problem called influence maximization. In this problem, the goal is to find a set of seed users to target so as to maximize the influence spread. To our knowledge, we have seen few studies [127] which apply social influence analysis to developing recommendation algorithms. In this study, the authors developed a probabilistic generative model, the social influenced selection (SIS) model, which integrated social influence, user behavior and item content. They applied this model to item recommendation and group recommendation. Result showed that the SIS model was effective for both recommendation tasks. They also showed that users rarely followed their friends' uncommon preferences. Moreover, the strength of influence between a user and one friend was not correlated with their similarity. In a recommender system, we say that a user is activated if he reads a book or listens to a song. Given a user and her social connections' actions on an item in the past, we can build a model of social influence, and apply the model to predicting whether a user will adopt the item in the future. If the prediction is that the user will adopt it, we can recommend the item to the user.

In this chapter, we study the pattern of implicit social influence in a recommendation system, and try to find out the recommendation effectiveness based on such implicit influence. To achieve this goal, we analyze the information passing process, and propose three research questions. First, on average, when an individual reads a book, how likely is that her follower will read this book later? Second, for each individual user, what factors can affect her information pass rate? Third, will more recommendations lead to higher acceptance rate? Our study is different from the previous studies in two aspects. First, the social influence in our dataset is implicit instead of explicit. Users do not recommend items directly to their followers; instead, the followers themselves check the updates of their

followees. Our study also differs from [42]. In their problem, the only input is the infection times of each node. They have no knowledge of each node's neighbors, which we have.

The remainder of this chapter is organized as follows. The next section introduces related work and research questions. Section 7.2 defines core concepts, and formulizes our problem. In Section 7.3, we introduce our approaches. Section 7.4 present our dataset and results. Section 7.5 summarizes this paper and discusses future work.

7.2 Problem Formulation

7.2.1 Definitions of core concepts

Before formulizing our problem, we first differentiate between explicit and implicit social influences. Explicit social influence exists in the process that a user first reads a book and recommends it to a follower. The follower then decides whether or not to read the book. Implicit social influence exists in the process that a user first reads a book, and then one of her followers sees this activity, and makes decisions later. Figure 7.1 shows an example of an explicit or implicit feedback. For both types of feedbacks, we have $t_1 < t_2 < t_3$.

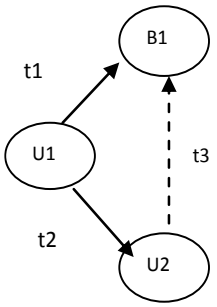


Figure 7.1 An Information transmission triangle

7.2.2 Model information pass rate

In [45], the information pass rate of an entire network was calculated, and compared with that of a random network. We define this information pass rate as global information pass rate, and also calculate it. However, it is of more interest to model the individual information pass rate. Recognizing that the transmission probability between two nodes depended on the properties of both the nodes and edges, the authors in [42] built a simple model where the transmission probability depended only on the difference between the node hit times$\Delta(u,v) = t_v - t_u$. We do not directly apply predefined models (power-law, exponential, etc.). Instead, we first propose several hypotheses of the possible factors on the information pass rate. Then, we calculate the individual information pass rate to evaluate our hypotheses.

In [47], the information pass rate was defined as the likelihood that one's friend purchases a product given that one first purchases a product, then told the friend. Formally, it equals to:

$Prob(b_2\, buys\, from\ s_1\, at\, t_2 + \Delta | b_1\, buys\, from\ s_1\, at\, t_1\ \cap$

$b_1 messages\, b_2\, at\, t_2, t_2\ > t_1)$ Δ denotes the lag between the time the friend buys it and the time one tells her. They required Δ <=2 days to decrease the effects of regular purchases that were irrelevant to messaging behavior. Here, we define the Information pass rate as:

$$Prob(u_2\, reads\, b_1\, at\, t_3 | u_1 reads\, b_1\, at\, t_1\ \cap\ u_2 follows\, u_1\, at\, t_2)$$

where $t_3 = \max\{t_1, t_2\} + \Delta$, and t_1 and t_2 has no order. Similar to [47], we also require that Δ less than a threshold. In our experiment, we choose the threshold to be two days. Thus, only if u_2 reads a book within two days after her followee u_1 reads it, or soon

after she follows u_1, we count it as an successful information pass. As shown in Figure 7-1, we define lag as$t_3 - t_1$. This is different from the definition in [47], which is$t_3 - t_2$. This is because in our douban dataset, there is no information regarding when a follower sees a followee's activity, i.e. we have no information about t_2.

7.3 Methods

7.3.1 Model to predict the information pass rate

In this section, we try to answer the question that "When an individual reads a book, how likely that is her friend will read this book later". We will first calculate the average information pass rate for all the users. Then, we calculate the individual information pass rate for each user. Finally, we try to find out if one's information pass rate is correlated with one's centrality. We make hypotheses in this section and will answer them in the results section.

7.3.1.1 The Existence of Social influence in Global information pass rate

To validate the existence of social influence, we follow the approach in [4] by taking the shuffle test and edge-reversal test. These two tests can help us fin dour if social influence is the only reason for social correlation. We will briefly introduce these two tests here. Define G as follower-followee network, $U = \{u_1, u_2, \dots, u_m\}$as the user set. Each user u_i has an activation time t_i within the period$[0, T]$. The shuffle test is performed by first selecting a random permutation π of$\{1, \dots, m\}$. Then it sets the activation time $t_i$of each user to $t_i' \coloneqq t_{\pi(i)}$. The edge reversal test is performed by reversing all the edges in G.

We compute the global information pass rate in both the original graph G and the

newly created graph G' for two tests. Our assumption is that if the information pass rate in G and G' are close, then social influence does not exist; otherwise, social influence does exist. The global information pass rate (IPR) is calculated in three scenarios.

(1) IPR based on the whole collection. This is the probability that a user reads a book in her followee's collection, given that the followee has read, is reading, or just wishes to read the book.

(2) IPR based on the read books. This is the probability that a user reads a book in her followee's collection, given that her followee has read it.

(3) IPR based on the rated books. This is the probability that a user reads a book in her followee's collection, given that her followee has explicitly rated it.

IPR under these three scenarios measure on average, how much a user is influenced by a followee's tastes given the followee's whole collection, read books and rated books, respectively. Since both "read" and "rated" can be signals of a book's quality, we assume that the IPR based on the followee's read or rated books should be higher than that based on her whole collection. Accordingly, we propose two hypotheses below.

H1: The IPR based on the followee's read books is larger than that based on all books.

H2: The IPR based on the followee's rated books is larger than that based on all books.

7.3.1.2 Individual information pass rate

In previous sections, we present our approach to compute the global Information Pass Rate (IPR) by taking into account all the triangles in the network. In this section, we introduce our approach to compute the IPR of each user. Then, we can analyze the correlation between each user's IPR and her network centralities, including indegree,

outdegree, and total book number. We make two hypotheses based on the role of the user in the recommendation.

H3: Each receiver's information pass rate, is correlated with her indegree centrality. The higher the indegree of the receiver, the larger the information pass rate.

H4: Each sender's information pass rate, is correlated with her outdegree centrality. The higher the outdegree of the sender, the larger the information pass rate.

7.3.2 Probability of Acceptance versus Number of Recommendations

In this section, we introduce our approach to analyzing the relationship between the probability of reading a book versus the number of recommendations for this book. From the receiver's perspective, we ask the research question that, do users who receive more recommendations, have a higher probability to accept them? From the sender's perspective, we ask the research question that, do users who make more recommendations, have a higher probability of get her recommendations accepted? We then make two hypotheses accordingly.

H5: Given a book, the more recommendations each user receives for this book, the higher the probability that she will read it.

H6: Given a book, the more recommendations each user sends out for this book, the higher the probability that her recommendations will be accepted.

7.4 Experiments and Results

7.4.1 Dataset

We use the same douban book dataset as in the last Chapter. The douban book dataset consists of a follower-followee social network with 4,779 users and 104,799 edges. It also contains a user preference network with 2,549,523 edges between 4,779 users and 434,005

books. According to our previous definitions of implicit and explicit feedbacks, feedbacks are implicit in douban.

7.4.2 Results

In this section, we present our results on global and individual Information Pass Rate. Then, we show result on the relationship between the number of recommendations with the probability of acceptance.

7.4.2.1 Model information pass rate

(1) The Existence of Social influence in Global IPR

We first compare the IPR in the original dataset with the dataset after the shuffle test. As shown in Table 7.1, in the first scenario (whole collection), the information pass rate is 0.00125, much larger than that of the shuffle test, which is 3.99 E-5. In the second scenario ("read" books only), the information pass rate is 5.415E-4, ten times larger than that of the shuffle test, which is 5.02E-5. For the third scenario (rated books), the information pass rate is 6.65E-4, ten times larger than that of the shuffle test, which is 5.20E-5. However, in all three scenarios, we found little differences between IPR of the original dataset and IPR of the dataset after the edge reverse test. Consequently, we conclude that the social influence does exist, but is not significant.

We also observed that the IPR based on the whole collection, is larger than both the IPR based on read books, and the IPR based on rated books. Thus, neither *H1* nor *H2* hold. This is contradictory to our intuition. One explanation is that when a user see her followers recent reading activates, he or she does not care about whether the follower has finished reading the book or has rated it.

Table 7.1 Global IPR in Original, Shuffle Test, and Edge Reversal Test

Scenario	Original	Shuffle	Edge Reversal
whole collection	0.00125	3.99 E-5	0.00108
read books	5.415E-4	5.02E-5	4.05E-4
rated books	6.65E-4	5.20E-5	5.27E-4

(2) Individual-based information pass rate

We then analyze if there exists correlations between one user's information pass rate and her network centralities. Our results are shown in Table 7.2. The asterisk (*) means that the p value is lower than 0.05. It can be seen that each user's IPR is not correlated with his followee number (outdegree), or the total number of books she has read. However, one's IPR is correlated with her follower number (indegree). This indicates that the more followers one has, the more influential one becomes. Hence, *H3* holds, but *H4* does not hold. We also found out that one's follower number and followee number is significantly correlated, showing that a user with many followers also follows many others. One's book number is correlated with both his follower number and followee number, showing that a user who reads many books has many followers and follows many others.

Table 7.2 Correlation coefficient between individual IPR and other indices

	IPR	Outdegree	Indegree	Book#
IPR	1	--	--	--
Outdegree	0.0110	1	--	--
Indegree	0.0346*	0.3536*	1	--
Book#	0.0083	0.2550*	0.1657*	1

7.4.2.2 Probability of Acceptance versus Number of Recommendations

We plot the probability of reading a book versus the number of incoming and outgoing recommendations in Figure 7-2. Following Leskovec et al. [70], we also assume that each point in the plot follows a binomial distribution. Given the number of observations$n$, let m and $k(k = n - m)$be the number of successful and failed recommendations. In our case, m is the number of users who read a book after receiving r recommendations on it. Then the estimated probability of reading is$\hat{p} = m/n$. The standard error $s_{\hat{p}}$ of estimate$\hat{p}$ is$s_{\hat{p}} = \sqrt{p(1-p)/n}$. We use this to draw an error bar on each point. Given useru_i, for each book in her followeeu_j's collection, if u_i did not read it or read it later thanu_j, we consider it as a recommendation. If the time u_i read it is later than u_j and is within our predefined threshold, we consider this recommendation as accepted. If one received a recommendation but did not read it or read it after the threshold time, we consider it as not accepted. We show the results in Figure 7.2.

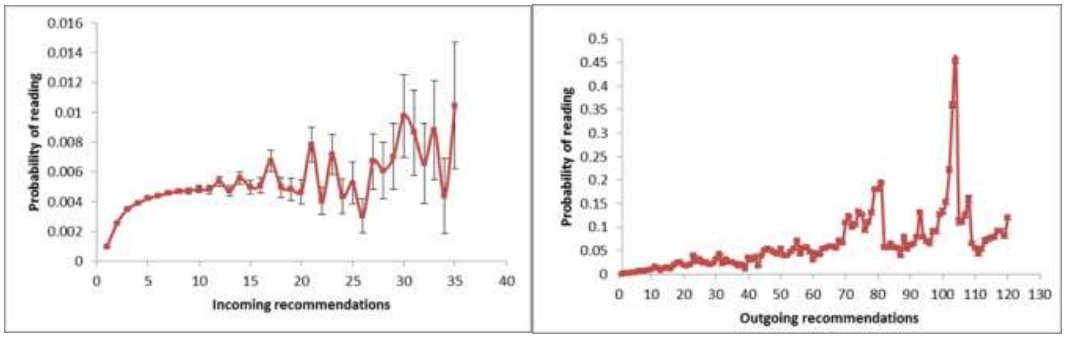


Figure 7.2 Probability of reading a book given number of incoming (left) and outgoing (right) recommendations

(1) Success of Incoming Recommendations

The left plot of Figure 7.2 shows the relationship between the probability of reading a book and the number of incoming recommendations. We see a saturation point appears at around ten recommendations. This is similar with the pattern of DVD recommendations in previous studies [70]. However, their result on book recommendation showed that from the second recommendation and up, the more recommendations one received for a book, the less likely that she would read it. Nevertheless, in our dataset, we see that the more recommendations one receives for a book, the more likely that she will read it. The probability that she reads it stops increasing until she gets ten recommendations. Thus, *H5* does not hold. This finding is consistent with the linear threshold model [43]. This model considers the social influence at the individual level. It states that each user has a threshold. Once the number of recommendations passes one's threshold, he or she would accept the recommendations.

The authors in [70] explained that their DVD recommendation pattern comes from a result of "someone signed up to receive DVD recommendations for a product they intended to purchase". Hence, a larger number of received recommendations correspond to a higher likelihood of purchase. Since our book recommendation result share similar pattern with their DVD recommendation, we believe one possible reason for our plot is that douban users also signed up douban to receive book recommendations. However, this is unknown till in-depth user studies are conducted.

(2) Success of Outgoing Recommendations

In [70], books, music, and videos present similar patterns, where the number of purchases grows fast up to around 10 outgoing recommendations and then the trend either slows or starts to drop. However, their DVDs exhibit a different pattern. The expected

number of purchases increased throughout.

The right plot of Figure 7-2 shows that, the book recommendation in our douban dataset shares similar patterns with the DVD recommendations in [70], again. When one sends out more recommendations, the probability that her recommendations are accepted by others becomes higher. Thus, H6 holds. This result indicates that sending out more recommendations to one's followers, can be an effective strategy for viral marketing in online recommender systems.

7.5 Conclusion and Future work

This chapter showed that recommendations from one's social relations do have impact over one's decisions. The more recommendation one receives for a book, the higher the probability that one will read it. However, once the threshold is passed, the receiver will no longer be influenced in the recommended book. This resembles the linear threshold model proposed by Granovette [43]. Thus, one of our future studies is to test if the information pass in the recommendation network fits this model. Moreover, we can compare the recommendation success rate between different book categories in our dataset, and see if the result is similar to that of [70]. Finally, "edge-reversal test" showed that social influence is not significant. More experiments need to be done to test if social influence is the only source of the social correlations between a user and her followers.

Chapter 8. Conclusion and Future work

8.1 Conclusion

Recommender system has become a hot topic in academia and has been widely adopted in industry. Among various recommendation approaches, collaborative filtering is the most popular one since it is domain independent and easy to implement. However, CF suffers from the item-side and user-side cold-start problems. Item-side cold-start denotes that CF cannot recommend new items to users. User-side cold-start denotes that CF is not capable to give recommendation to new users. Besides, most model-based CF performs poorly on users with few ratings.

In general, this thesis comprise of two parts. In the first part, we deal with the item-side cold-start problem. In the second part, we try to solve the user-side cold start problem. The approaches we proposed or implemented are based on the goal of helping solving the cold-start problem.

In the first part (Chapter 3 and 4), we incorporate item content to alleviate the item-side cold start problem. In Chapter 3, we incorporate casting information and proposed a topic model adapted from the Author-topic model, and apply it to movie recommendation. Results show that in implicit rating prediction task, topic models have comparable performances with Matrix Factorization. In explicit rating prediction task, they outperform MF. In Chapter 4, we include music acoustic features and propose a hierarchical sequence alignment model to computing music similarities. These similarities are then utilized for music recommendation. Results show that our single layer alignment algorithm has comparable performance on cold songs, indicating that this content-based algorithm is able to give recommendations with both novelty and relevance.

In the second part (Chapter 5, 6, and 7), we incorporate user profiles and social networks to alleviate the user-side cold start problem. In Chapter 5, we compare the structures of the "familiarity-based" and "similarity-based" networks in the same system. We also find out that users' followees and their interests sharers do not overlap. This leads to the conclusion that social networks could be a valuable data source for recommendation tasks. Based on this finding, in Chapter 6, we apply two social recommendation algorithms on MovieLens and two douban datasets. Besides social networks, we also construct a similarity network based on user demographics. Results show that social recommenders do not always outperform the matrix factorization baseline.

8.2 Future work

This thesis has made progress in developing content-based and social-based recommendation approaches to solve the cold-start problem. Several open problems still need us to explore.

For the item-side cold start problem, more work need to be done to develop content-based algorithms. In the task of movie recommendation, we can add additional information as regularization terms. Based on the assumption that co-star actors should have similar topic distributions, we can construct an actor co-star network and add it as a regularization term into the loss function of the actor-topic model. In music recommendation, single layer alignment algorithm BTMLYR outperforms its multi-layer counterpart HSA. We believe the reason is that parameters in HSA have not been set to its optimal. Given that HSA has many parameters and hence the grid search would not work, we need to find an automatic algorithm to tune its parameters. Another direction is to develop a hybrid recommendation algorithm via combining our HSA model with memory-based CF or latent factor models.

It is also interesting to analyze how genre affects the recommendation performance. User studies also need to be done to evaluate our algorithms on metrics besides precision and recall.

The comparison between similarity network and familiarity network can be extended to other domains such as music and movie. Other similarity measures, such as Jaccard similarity, can also be utilized to construct the similarity network. Since these two networks do not overlap, one application is friend recommendation, where we can recommend interest sharers to users as their friends.

For the user side cold start problem, since the two existing social recommenders do not always outperform the matrix factorization baseline, we need to design a more realistic social recommendation algorithm. Considering that people may consult different social connections on different topics, a topic-based social recommendation is necessary. Given a user and a topic, this algorithm would first identify the group of social connectors who have largest impact on this topic and then recommend items based on their preferences. Moreover, studies have shown that different forms of social networks have different characteristics [125]. Thus, it is also interesting to see what types of social relations can help improve recommendation performance most. Given that our social influence study generates results consistent with the linear threshold model, we can further test if the information pass in the recommendation network fits this model. Moreover, we can compare the recommendation success rate between different book categories on our dataset.

List of References

[1] L. A. Adamic, O. Buyukkokten, and E. Adar. A social network caught in the Web. First Monday, 8(6), 2003.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of the-art and possible extensions. IEEE Trans. Knowl. Data Eng., 17(6), pages 734-749, 2005.

[3] D. Agarwal and B-C. Chen. flda: matrix factorization through latent dirichlet allocation. Proceedings of the Third ACM International Conference on Web Search and Data Mining, pages 91-100, 2010.

[4] A. Anagnostopoulos, R. Kumar, M. Mahdian. Influence and correlation in social networks. In KDD, pages 7-15, 2008.

[5] S.A. Sarabjot and N. Griffiths. A market-based approach to address the new item problem. In Proceedings of the fifth ACM conference on Recommender systems, RecSys '11, pages 205–212, New York, NY, USA, 2011.

[6] C. Anderson. The Long Tail: Why The Future of Business Is Selling Less of More. New York, NY: Hyperion. 2006.

[7] J.-J. Autouturier and F. Pachet. Finding songs that sound the same. In Proceedings of IEEE Workshop on Model based Processing and Coding of Audio. University of Leuven, Belgium, November 2002. Invited Talk.

[8] B. Eyta, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In Proceedings of the fourth ACM international conference on Web search and data mining, WSDM '11, pages 65–74, New York, NY, USA, 2011.

[9] F. Bass, A new product growth for model consumer durables. Manage. Sci. 15, 5, pages 215-227, 1969.

[10]T. Bertin-Mahieux, D. P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), 2011.

[11]D. Billsus and M. J. Pazzani. Learning collaborative information filters. In ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, pages 46-54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[12]D. Blei and J. McAuliffe. Supervised topic models. In NIPS, 2007.

[13]D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 3, pages 993-1022, January 2003.

[14]T.Bogers. Movie Recommendation using Random Walks over the Contextual Graph. In CARS'2010. 2010.

[15]P. Bonhard & M. A. Sasse. Knowing me, knowing you --Using profiles and social networking to improve recommender systems. BT Technology Journal 24, 3 (Jul. 2006), pages 84-98. 2006

[16]S.P. Borgatti, M.G. Everett, and L.C. Freeman. Ucinet for Windows: Software for Social Network Analysis. Harvard, MA: Analytic Technologies. 2002.

[17]J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In UAI'98: Proceedings of Uncertainty in Artificial Intelligence, 1998.

[18]E. Brynjolfsson, Y. Hu, and M. D. Smith. Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. Manage. Sci.49,11, pages 1580-1596, 2003.

[19]K. Burke. As consumer attitudes shift, so must marketing strategies. study suggests rethinking the customer experience. http://www.audioholics.com/contact/ConsumerAttitudesMarketingStrategiesStudy.html, 2003.

[20]L. Caroline, S. Kamel, and L. David. Building Parallel Corpora from Movies. In The 4th International Workshop on Natural Language Processing and Cognitive Science – NLPCS 2007. 2007.

[21] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. Proceedings of the IEEE, 96(4): pages 668–696, 2008.

[22] C. Castillo, W. Chen, L. V. S. Lakshmanan. KDD'2012 Tutorial: Information and Influence Spread in Social Networks http://research.microsoft.com/en-us/people/weic/kdd12tutorial_inf.aspx. Retrieved in May 2013.

[23] Z. Cataltepe, B. Altinel. Music recommendation based on adaptive feature and user grouping. In 22nd International Symposium on Computer and Information Sciences, Ankara, Turkey. 2007.

[24] O. Celma. Music Recommendation and Discovery in the Long Tail. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[25] O. Celma and P. Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. In 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, 2008.

[26] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummad. Measuring user influence on twitter: The million follower fallacy. In 4th Int'l AAAI Conference on Weblogs and Social Media, Washington, DC, 2010.

[27] X. Chen, X. Hu, Z. Zhou, C. Lu, G. Rosen, T. He, and E. K. Park. A probabilistic topic-connection model for automatic image annotation. In CIKM, pages 899–908, 2010.

[28] Y. H. Chien and E. I. George. A bayesian model for collaborative filtering. In Proceedings of the Seventh International Workshop Artificial Intelligence and Statistics, 1999.

[29] M. Claypool, A. Gokhale, and T. Miranda. Combining content-based and collaborative filters in an online newspaper. In Proceedings of the ACM SIGIR Workshop on Recommender Systems -Implementation and Evaluation, 1999.

[30] de Nooy, W., A. Mrvar, and V. Batagelj, Exploratory Social Network Analysis with Pajek, Cambridge University Press, 2005.

[31] S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In Proceeding of the 17th international conference on World Wide Web, WWW'08, pages 1041–1042, New York, NY, USA, 2008. ACM.

[32] H. Ding, J. Huang and H. Wang. US Patent. HAM- Hierarchical Alignment in Media applications. Patent No: 00155.0019.00US.

[33] Pedro Domingos and Matt Richardson. Mining the network value of customers. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[34] C. Eckart, G.Young, The approximation of one matrix by another of lower rank. Psychometrika 1 (3): 211–8. doi:10.1007/BF02288367. 1936.

[35] D. P. Ellis and J. Arroyo. Eigenrhythms: Drum pattern basis sets for classification and generation. In Proceedings of the International Symposium on Music Information Retrieval (ISMIR), Barcelona, Spain, October 2004.

[36] L. Freeman. A set of measures of centrality based upon betweenness. Sociometry 40, pages 35-41. 1977.

[37] L. C. Freeman. Centrality in social networks: Conceptual clarification. Social Networks, 1(3), pages 215–239. 1979

[38] T. Fruchterman & E. Reingold. Graph drawing by force-directed placement. Software - Pract. Exp. 21, pages 1129–1164. 1991.

[39] S. Goel, A. Broder, E. Gabrilovich, and B. Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In WSDM, 2010.

[40] K. Y. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. Inf. Retr., 4(2), pages133–151, 2001.

[41] J.Goldenberg, B. Libai, and E.Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. Market. Lett. 3, 12, pages 211-223. 2001.

[42] M. Gomez-Rodriguez, J. Leskovec, and A.Krause, Inferring Networks of Diffusion and Influence. In Proc.of the 16th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pages 1019–1028, 2010.

[43] M. Granovetter. Threshold Models of Collective Behavior. The American Journal of Sociology, Vol. 83, No. 6, pages 1420-1443. 1978

[44] G.Groh, C. Ehmig, 2007. Recommendations in Taste Related Domains: Collaborative Filtering vs. Social Filtering. In Proc. ACM Group'07, pages127-136. 2007.

[45] A. Gunawardana and C. Meek. Tied Boltzmann machines for cold start recommendations. In RecSys '08.2008.

[46] L. Guo. Semantically Enhanced Topic Modeling and Its Applications into Social Media. PhD thesis, Drexel University. Philadelphia, USA, 2013.

[47] S. Guo, M. Wang, and J. Leskovec. The role of social networks in online shopping information passing, price of trust and consumer choice, In EC '11, 2011.

[48] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E.Uziel, S. Yogev, S. Ofek-Koifman. Personalized Recommendation of Social Software Items Based on Social Relations. In RecSys'09, pages 53-60. 2009.

[49] R. Hanneman and M. Riddle. Introduction to social network methods. Chapter 12: Network positions and social roles: The idea of equivalence. http://faculty.ucr.edu/~hanneman/nettext/C12_Equivalence.html#define, 2005.

[50] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 230–237, New York, NY, USA, 1999.

[51] Thomas Hofmann. Probabilistic latent semantic analysis. In Proceedings of the 15th Conference on Uncertainty in AI, pages 289–296, San Fransisco, California, Morgan Kaufmann. 1999a.

[52]T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pages 688–693, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. 1999b.

[53]T. Hofmann, J. Puzicha, and M. I. Jordan. Learning from dyadic data. In Advances in Neural Information Processing Systems 11, 1999c.

[54]T. 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 information retrieval, pages 259–266, New York, NY, USA, 2003a.

[55]T. Hofmann. Gaussian Latent Semantic Models for Collaborative Filtering. In 26th Annual International ACM SIGIR Conference, 2003b.

[56]T. Hofmann. Latent semantic models for collaborative filtering. ACM Transactions on Information Systems, 22(1): pages 89-115, 2004.

[57]Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, 2008.

[58]J. Huang, X. Hu, C. Lu. A Comparison Study on Familiarity-Based and Similarity-Based Social Networks. In Proceedings of the 2011 IEEE Conference on Granular Computing, 2011.

[59]T. Jehan. Creating Music by Listening. PhD thesis, Massachusetts Institute of Technology, 2005.

[60]T. Jehan, D. DesRoches. The Echo Nest Analyzer Documentation. Prepared by: September 2, 2011. Analyzer Version: 3.08. 2011.

[61]R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. Inf. Retr., 9(3), pages 357–382, 2006.

[62]S.Jurvetson. What exactly is viral marketing? Red Herring 78, pages 110–112. 2000.

[63] D. Kempe, J. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, pages 137–146, New York, NY, USA, 2003.

[64] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09, pages 195–202, New York, NY, USA, 2009.

[65] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 426-434, New York, NY, USA, 2008.

[66] Y. 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.

[67] Y.Koren, R. Bell, C.Volinsky. Matrix factorization techniques for recommender systems. IEEE Computer, vol. 42, pages 30–37, 2009.

[68] R. Krestel, P. Fankhauser, and W. Nejdl. Latent dirichlet allocation for tag recommendation. In Proceedings of the third ACM conference on Recommender systems, pages 61-68. ACM, 2009

[69] K.Lerman. Social networks and social information filtering on Digg. In Proc. ICWSM'07, 2007. Bonhard, P. & Sasse, M. A. 2006.

[70] J. Leskovec, L.A. Adamic, and B.A. Huberman. The dynamics of viral marketing, ACM Transactions on the Web, 2007.

[71] H. Li, G. Wu, X. Hu, J. Zhang, L. Li and X. Wu. K-means clustering with bagging and MapReduce. In Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), pages 1-8, 2011.

[72] H. Li, X. Wu, Z. Li, W. Ding. Online Group Feature Selection from Feature Streams. In Proceedings of Twenty-Seventh AAAI Conference on Artificial Intelligence, pages 1627-1628, 2013.

[73]H. Li, X. Wu, Z. Li, W. Ding. Group feature selection with feature streams. In Proceedings of IEEE 13th International Conference on Data Mining (ICDM), pages 1109 – 1114, 2013.

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

[75]C. Lu. Exploiting Social Tagging Network for Web Mining and Search. PhD thesis, Drexel University, Philadelphia, USA, 2012.

[76]H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, 2007, pages 39-46.

[77]H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management, pages 931-940, New York, NY, USA, 2008.

[78]H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In SIGIR 2009, pages 203-210. 2009.

[79]H. Ma, D. Zhou, C.Liu, M. R. Lyu and I. King. Recommender Systems with Social regularization. In WSDM '11.pages 287-296. 2011.

[80]B. Marlin. Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul, and B. Scholkopf, editors, Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA, 2004.

[81]B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In Proc. of ICML '04, page 73, Banff, Alberta, Canada, 2004.

[82]K. D. Martin. Sound-Source Recognition: A Theory and Computational Model. PhD thesis, MIT Media Lab, 1999.

[83]Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '12, pages 6–14, New York, NY, USA, 2012.

[84] T. Meng and M. Shyu, Leveraging Concept Association Network for Multimedia Rare Concept Mining and Retrieval. The 2012 IEEE International Conference on Multimedia & Expo (ICME 2012), pages 860-865, Melbourne, Australia, July 9-13, 2012.

[85] D. Mount. Bioinformatics: Sequence and Genome Analysis (2nd ed.). Cold Spring Harbor Laboratory Press: Cold Spring Harbor, NY. ISBN 0---87969---608---7. 2004.

[86] S. B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48 (3): pages 443–53.1970.

[87] A. Van den Oord, S. Dieleman, B. Schrauwen. Deep content-based music recommendation. Proc. NIPS, 2013.

[88] S-T. Park, D. Pennock, O. Madani, N.Good, and D. DeCoste. Naive filterbots for robust cold-start recommendations. In KDD ‘06, pages 699-705, 2006.

[89] S-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In RecSys ‘09, pages 21-28. 2009.

[90] A. Popescul, L.H. Ungar, D.M. Pennock. S. Lawrence. Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. In UAI ‘2001, pages 437-444, 2001.

[91] A. Rashid, G. Karypis, and J. Riedl. Influence in ratings-based recommender systems: An algorithm-independent approach. In SDM, 2005.

[92] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In ICML05: Proceedings of the 22th International Conference on Machine Learning, 2005.

[93] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In Proceedings of ACM Conference on Computer Supported Cooperative Work, 1994.

[94] M. Richardson and P. 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, New York, NY, USA, 2002.

[95] D. M. Romero, W. Galuba, S. Asur, and B. A. Huberman. Influence and passivity in social media. In Proceedings of the 20th international conference companion on World wide web, WWW '11, pages 113–114, New York, NY, USA, 2011.

[96] M. Rosen-Zvi, T.Griffiths, M. Steyvers, and P. Smith. The author-topic model for authors and documents. In AUAI'04: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence 487–494. AUAI Press, Arlington, VA. 2004.

[97] G. Sabidussi. The centrality index of a graph. Psychometrika 31, pages 581- 603. 1966.

[98] R. Salakhutdinov, A. Mnih. Probabilistic matrix factorization. Advances in Neural Information Processing Systems 20. Cambridge, MA: MIT Press. 2008a.

[99] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In ICML '08: Proceedings of the 25th International Conference on Machine Learning, 2008b.

[100] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In WWW '01: Proceedings of the 10th international conference on World Wide Web, pages 285–295, New York, NY, USA, 2001.

[101] J. Satter. N. Antonopoulos. CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering. IEEE Intelligent Systems, pages 35-41. 2006.

[102] AI. Schein, A. Popescul, LH. Ungar, and D. M. Pennock. Generate models for coldstart recommendations. Proceedings of the 2001 ACMSIGIR Workshop on Recommender Systems. ACM, New York. 2001.

[103] AI. Schein, A. Popescul, LH. Ungar. Methods and metrics for cold-start recommendations. In SIGIR '2002, pages 253-260, 2002.

[104] S. Shang, P. Hui, S. R. Kulkarni, and P. W. Cuff. Wisdom of the crowd: Incorporating social influence in recommendation models. In Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems, ICPADS '11, pages 835–840, Washington, DC, USA, 2011. IEEE Computer Society. 2011.

[105] R. N. Shepard. Towards a universal law of generalization for psychological science. Science, 237, pages 1317-1323. 1987.

[106] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In ICML ’03: Proceedings of the 20th International Conference on Machine Learning, 2003.

[107] R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries, 2001.

[108] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. In International Symposium on Music Information Retrieval (ISMIR2008), pages 313–318, September 2008.

[109] M. Slaney. Web-scale multimedia analysis: Does content matter? MultiMedia, IEEE, 18(2), pages12–15, 2011.

[110] T. F. Smith and M. S. Waterman. Comparison of biosequences.Advances in Applied Mathematics 2.4, pages 482-489. 1981.

[111] T. F. Smith and M. S. Waterman. Identification of Common Molecular Subsequences. Journal of Molecular Biology 147, pages 195–197. 1981.

[112] X. Song, Y. Chi, K. Hino, and B. L. Tseng. Information flow modeling based on diffusion rate for prediction and ranking. In Proceedings of the 16th international conference on World Wide Web, WWW ’07, pages 191–200, New York, NY, USA, 2007.

[113] X. Song, B. L. Tseng, C. Lin, and M. Sun. Personalized recommendation driven by information flow. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ‘06, pages 509–516, New York, NY, USA, 2006.

[114] N. Srebro, J. D. M. Rennie, and T. Jaakkola. Maximum margin matrix factorization. In NIPS, 2004.

[115] M. Stanley. The small world problem. Psychology Today.1(1), pages 60-67. 1967.

[116] M. R.Subramani, and B.Rajagopalan. Knowledge-sharing and influence in online social networks via viral marketing. Comm. ACM 46, 12, pages 300-307. 2003.

[117] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. ISMIR, 2008.

[118] R. Turetsky, N. Dimitrova. Screenplay alignment for closed system speaker identification and analysis of feature films. 2004.

[119] O. Tsur and A. Rappoport. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In Proceedings of the fifth ACM international conference on Web search and data mining, WSDM '12, pages 643–652, New York, NY, USA, 2012.

[120] C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In SIGKDD, pages 448-456, 2011.

[121] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 501-508, New York, NY, USA, 2006. ACM.

[122] J. Weng, E. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In Proceedings of the third ACM international conference on Web search and data mining, WSDM '10, pages 261–270, New York, NY, USA, 2010.

[123] L. Xiang, PhD thesis. University of Chinese Academy of Sciences. China. 2011.

[124] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster based smoothing. In SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pages 114–121, New York, NY, USA, 2005.

[125] C. C. Yang, X. Tang, J. Huang, and J. Unger, "A Study of Social Interactions in Online Health Communities," Proceedings of IEEE International Conference on Social Computing, Boston, MA, October 9 – 11, 2011

[126] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In Proceedings of the 2010 IEEE International Conference on Data Mining, ICDM '10, pages 599–608, Washington, DC, USA, 2010.

[127] M. Ye, X. Liu, and W. Lee. Exploring social influence for recommendation: a generative model approach. In SIGIR 2012, 2012.

[128] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In Proc. of SIGIR ‘07, pages 47–54, New York, NY, USA, 2007.

VITA

CONTACT
Jia Huang, cucumberguagua@gmail.com, 215-827-9012

EDUCATION

Ph.D. in Information Studies **2009 - 2014**
College of Computing and Informatics, Drexel University, Philadelphia, PA
Advisor: Xiaohua (Tony) Hu, GPA: 3.88/4.00

M.S. in Information Resources Management **2007- 2009**
School of Information Management, Wuhan University, Wuhan, China
Advisor: Feicheng Ma, GPA: 3.86/4.00

B.S. in Electronic Commerce **2003 - 2007**
School of Information Management, Wuhan University, Wuhan, China
GPA: 85/100, Ranking: 5/62

RESEARCH INTERESTS

Machine Learning, Information Retrieval and Social Network Analysis. Particularly interested in Recommender Systems and Social influence.

RESEARCH EXPERIENCES

Research Assistant, College of Computing and Informatics, Drexel University 2009-2014

SELECTED PUBLICATIONS

- J. Huang, H. Ding, X. Hu and Y. Liu. A Two-level Approach for Subtitle Alignment. Advances in Information Retrieval Lecture Notes in Computer Science Volume 8416, 2014, pp 468-473.
- J. Huang, X. Hu. Information Passing in Online Recommendation. In The First Workshop on User Engagement Optimization at ACM CIKM 2013, San Francisco, CA.
- H. Ding, J. Huang and H. Wang. US Patent. HAM- Hierarchical Alignment in Media applications. Patent No: 00155.0019.00US. 2013.
- L. Guo, J. Huang, Y. Ling and X. Hu. Probabilistic Item Social Reputation Model for Recommender System. In the CrowdRec 2013 Workshop at ACM RecSys 2013, Hong Kong.
- J. Huang, X. Hu, C. Lu. A Comparison Study on Familiarity-Based and Similarity-Based Social Networks. In Proceedings of the 2011 IEEE Conference on Granular Computing, 2011, Kaohsiung.
- C. C. Yang, X. Tang, J. Huang, and J. Unger, A Study of Social Interactions in Online Health Communities. In Proceedings of IEEE International Conference on Social Computing, Boston, MA, October 9 - 11, 2011.
- C. Lu, X. Hu, J. Park, J. Huang. Post-based Collaborative Filtering for Personalized Tag Recommendation. In Proceedings of the iConference 2011, Seattle WA.